Diplomarbeit

Konfiguration adaptiver Interaktion mittels Deep-Learning

eingereicht von

Tim Gröger
geboren am 11.12.1994 in Spremberg

Technische Universität Dresden

Fakultät Informatik
Institut für Angewandte Informatik
Lehrstuhl Mensch-Computer-Interaktion

Betreuer:
David Gollasch, M. Sc.

Hochschullehrer:
Prof. Dr. rer. nat. habil. Gerhard Weber

Eingereicht am 14. Dezember 2022

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Angewandte Informatik Professur für Mensch-Computer-Interaktion

Aufgabenstellung für die Diplomarbeit

Name des Studenten: **Tim Gröger**
Studiengang: **Diplom Informatik**
Immatrikulationsnummer: **3952995**

Thema:

Konfiguration adaptiver Interaktion mittels Deep-Learning

Configuration of Adaptive Interaction by Means of Deep Learning

Zielstellung

Kontext. Entlang der aktuell stark bearbeiteten Forschungsfelder der Künstlichen Intelligenz und der natürlich-sprachlichen Interaktion entwickelt sich der Bereich der Robotik fernab von Industrierobotern in schnellem Tempo weiter. Insbesondere die Bereiche der Sozialen Assistenzroboter (SARs) und Mensch-Roboter-Interaktion bieten allerdings noch zahlreiche offene technische und auch anwendungsbezogene Fragen.

Ein zunehmender Bedarf an SARs erschließt sich im Umfeld von Menschen mit Beeinträchtigungen, Senioren sowie im Altenpflegebereich. SARs könnten hier eingesetzt werden, um die Selbständigkeit und Selbstbestimmtheit der Nutzenden zu fördern. Gleichzeitig können solche Assistenzsysteme dazu beitragen, den personell angespannten Pflegebereich zu entlasten.

Bisherige SARs verfügen noch nicht über einen breiten Funktionsumfang, um verschiedentliche Anwendungsfälle abzudecken und die Entwicklung von Software für SARs ist mangels plattformübergreifender Lösungen teuer. Eine besondere Herausforderung ist außerdem die Umsetzung einer gebrauchstauglichen Mensch-Roboter-Interaktion. Idealerweise wird diese benutzendenzentriert entwickelt und erlaubt eine automatische Adaption an die Anforderungen der Nutzenden, den Anwendungskontext und sich dynamisch ändernde Faktoren.

Ein Ansatz zur Lösung der beschriebenen Probleme kann die Entwicklung einer gemeinsamen Anwendungsarchitektur, basierend auf den Methoden aus der Softwarevariabilität, sein. Diese umschließt neben den implementierten Anwendungsfällen auch die Interaktion und ermöglicht daher die erforderliche Adaptivität.

(Fortsetzung Rückseite)

Fachbetreuer: David Gollasch, M.Sc.
Beginn am: 01.04.2022
Einzureichen am: 02.09.2022 (22 Wochen)

Dresden, 03.03.2022

Prof. Dr. rer. nat. habil. Gerhard Weber
verantwortlicher Hochschullehrer

Technische Universität Dresden
Fakultät Informatik
Institut für Angewandte Informatik
Professur Mensch-Computer-Interaktion

Projektziel. Ziel dieser Diplomarbeit ist die Beantwortung der folgenden Fragestellung: Welche Schritte sind nötig, damit ein Assistenzroboter selbst lernt, eine optimale Interaktion für einen Benutzenden während der Laufzeit zu wählen, welche die Einschränkungen des Benutzenden berücksichtigt? Zu beachten sind in diesem Zusammenhang die folgenden Punkte: Gesucht wird in erster Linie ein KI-basiertes Konfigurationsverfahren, das sich auf Feature-Modelle anwenden lässt, die zur Anpassung von Benutzungsoberflächen eingesetzt werden. Dabei handelt es sich um einen Konfigurationsprozess zur Laufzeit. Das Feature-Modell bildet ein Mapping zwischen den Anforderungen der Nutzenden und verfügbaren Designvarianten der Benutzungsoberfläche. Der Bereich der Assistenzroboter (SARs) bildet den Anwendungskontext für die zu entwickelnde Lösung, sodass die Konfiguration der Sprachinteraktion (VUI/CUI) im Fokus der Arbeit steht. Zur Datenerhebung bzgl. der Nutzendenanforderungen soll auf die Ergebnisse bestehender Arbeiten zurückgegriffen werden. Die Umsetzung eines Prototyps umfasst einen Sprachassistenten, der mittels des entwickelten Verfahrens seine Dialogführung an die Bedürfnisse der Nutzenden anpasst. Die Evaluation des entwickelten Verfahrens besteht aus einem Funktionstest des KI-basierten Verfahrens sowie aus einer Untersuchung der Technologieakzeptanz (TAM) mittels Probanden mit unterschiedlichen Anforderungsprofilen.

Schwerpunkte:

- Einarbeitung in die folgenden Themengebiete inklusive Analyse des aktuellen Forschungsstandes:
    - Softwarevariabilität, insbesondere Modellierung mittels Feature-Modellen und Runtime-Konfiguration von Feature-Modellen
    - Deep-Learning-Verfahren inkl. Kaltstart-Problematik, Lernfähigkeit und Ressourcenverbrauch zum Netz-Training; Einarbeitung in TensorFlow
    - Modellierung von Einschränkungen von Menschen mit Behinderung (ISO Guide 71 und ergänzende Dokumente)
    - Entwicklung eines Voice User Interface mit MyCroft.ai und RASA
    - Umsetzung von Sprachinteraktion für einen SAR (Loomo, Android-basiett)
    - Technologieakzeptanz von Sprachassistenten und Sprachinteraktion von SARs bei Senioren
- Analyse der Anforderungen und Entwicklung eines systematischen Konzepts zur Konfiguration von Feature-Modellen zur Anpassung von Sprachinteraktionsoberflächen
- Umsetzung des Konzepts unter Verwendung des Android-basierten Roboters Loomo, ggf. auch weiterer Komponenten, wie bspw. eines Raspberry Pi zur Auslagerung von Berechnungen
- Evaluation und Auswertung des erarbeiteten Verfahrens auf angemessene, wissenschaftliche Weise mittels Funktionstest und Nutzendenevaluation.
- Dokumentation der Ergebnisse in geeigneter, wissenschaftlicher Form

Relevante Vorarbeiten:

[1] Augustat, Leon; Liefke, Robert; Pattoka, Thomas. Enable the Elderly to use Voice Assistants within Social Assistance Robots. Complex Practical Task. 2021.
[2] Fuchs, Julian. Development of User Profiles for Adaptive Assistance Robots. Master's Thesis. 2021.

[3] Gröger, Tim. Designing a Configuration Method for Adaptive Assistance Robot Interaction. Term Paper. 2021.
[4] Ziemann, Sophie. Construction of Variable App-Level Process Chains in Android. Term Paper. 2020.
[5] Haustein, Yasmin. Best Practises for Conversational User Interfaces of Assistance Robots for the Elderly. Bachelor's Thesis. 2020.
[6] Eisoldt, Martin. Construction of an Independent Natural-Language Human-Robot Interaction. Master's Thesis. 2019.

Erklärung

Ich erkläre, dass ich die vorliegende Arbeit mit dem Titel *Konfiguration adaptiver Interaktion mittels Deep-Learning* selbständig unter Angabe aller Zitate angefertigt und dabei ausschließlich die aufgeführte Literatur und genannten Hilfsmittel verwendet habe.

Dresden, 14. Dezember 2022

Tim Gröger

Kurzfassung

Es werden immer mehr Pflegekräfte benötigt, um in unserer alternden Bevölkerung Bedürftigen entsprechende Unterstützung zu geben. Vorraussichtlich werden in Zukunft nicht genügend Pflegekräfte vorhanden sein, um diese Aufgabe zu bewerkstelligen. Eine Abhilfe können hier sogenannte soziale Assistenzroboter sein, die Bedürftigen im Haushalt oder bei sonstigen Aktivitäten unterstützen. Eine zentrale Rolle zwischen sozialen Assistenzrobotern und Bedürftigen spielt die Kommunikation. Insbesondere muss sich ein sozialer Assistenzroboter auf gegebene (altersbedingte) Einschränkungen einstellen können und entsprechend abgestimmte Interaktionsmöglichkeiten für die Kommunikation anbieten.

Ziel dieser Arbeit ist es ein Konzept zu erarbeiten, wie mithilfe von Deep-Learning für einen Benutzer mit bestimmten Einschränkungen und Eigenschaften eine passende Interaktion gewählt werden kann. Hierfür wurde ein Feature-Modell erarbeitet, welches valide Konfigurationen für eine sprachliche Interaktion mit älteren Personen zeigt. Weiterhin wurden mehrere Deep-Learning-Modelle verglichen und ein Neuronal-Collaborativ-Filtering-Modell gewählt. Dieses Modell wurde erweitert, sodass neben dem kollaborativen Filtern auch die Features einer Interaktion und die Eigenschaften sowie Einschränkungen des Benutzers bei der Bewertung einer Interaktion mit einbezogen werden.

Um dieses Modell zu testen und zu trainieren, wurden mithilfe von Studien und Statistiken Daten generiert und es wurden an das Konzept Anforderungen definiert. Die Tests zeigen, dass solch ein Modell in der Theorie genutzt werden kann, jedoch verbessert werden muss.

Abstract

While society continues to age, more and more caregivers are needed to take care for the elderly population. However, in the future there will not be enough caregivers to accomplish this task. A remedy can be social assistance robots, which can support elderly people in the activities of daily living. Because of users can have different constraints and properties the social assistance robot needs to know which interaction can be used to establish optimal human robot communication.

The aim of this work is to develop a concept of how a suitable interaction can be selected for a user with certain constraints and properties using deep learning. A feature model was developed that shows valid configurations for verbal interactions with elderly people. Furthermore, several deep learning models were compared and a neuronal colaborativ filtering model was chosen. This model was extended to include, in addition to collaborative filtering, the features of an interaction and the users's properties and constraints when evaluating an interaction. To test and train this model, data was genereted using studies and statistics.

The model requirements have to have a defined concept for the testing phase.

Inhaltsverzeichnis

1 Motivation

Derzeit leben ca. 7,9 Milliarden Menschen auf der Welt. Laut United Nations [UDP22] befindet sich die Weltbevölkerung im ständigen Wachstum. So prognostiziert United Nations, dass 2100 ca. 10,8 Milliarden Menschen auf der Erde leben.

Neben dem stetigen Wachstum der Weltbevölkerung steigt auch das Alter. Somit zeigt United Nations et al. [UDP20] ebenfalls, dass die Anzahl der Menschen über 64 Jahren steigen wird. In den nächsten 30 Jahren wird die Anzahl dieser von 727 Millionen auf 1,5 Milliarden anwachsen. Im Jahr 2020 lag der Anteil der älteren Bevölkerung bei 9,3 % und steigt im Jahr 2050 auf ca. 16 %. Hier wird deutlich, dass immer mehr Menschen pflegebedürftig werden.

Allein in Deutschland wird es laut Müller und Rothgang [MR16] nicht genügend Ausbildungsplätze geben. Dies hat zur Folge, dass nicht genügend Pflegepersonal existieren wird, um pflegebedürftige Menschen zu pflegen. Somit werden Pflegefachkräfte durch den Mangel in extreme Belastungen gelangen. Cichy [Cic20] zeigt, dass schon 2020 Pflegekräfte unter starker Belastung stehen. Diese findet ihre Ursprünge in schlechter Entlohnung, starkem Zeitmangel und Unterbesetzung.

Abhilfe durch Roboter Eine Abhilfe können hier sogenannte Sozialer Assistenzroboter (SAR) schaffen. Diese können die Pflegekräfte entlasten, indem sie dem Pflegepersonal Arbeiten abnehmen. Derzeit werden im Gesundheitswesen solche Roboter erforscht und eingesetzt. Zu den Aufgabenfeldern zählen unter anderem die therapeutische und mentale Unterstützung, sowie die Überwachung von Vitalwerten. Somit können Roboter sowohl physisch wie auch psychisch den Pflegebedürftigen Unterstützung bieten, dabei auch im Haushalt oder im allgemeinen Alltag. Roboter können Pflegebedürftige im Haushalt sowie allgemein im Alltag sowohl physisch als auch psychisch unterstützen. Im Film „Robot & Frank“ aus dem Jahr 2012 wird gezeigt, wie Roboter Pflegebedürftigen und älteren Menschen in der Zukunft den Alltag erleichtern können.

Damit ein Roboter im alltäglichen Leben einer älteren Person helfen kann und von dieser akzeptiert wird, sollte die Nutzerfreundlichkeit durch Personalisierung der Kommunikation angepasst werden. Ziel ist hierbei die Interaktion mit dem Roboter adaptiv zu gestalten, sodass der Roboter selbst die Interaktion variabel an den Benutzer anpassen kann. Laut Müller [Mül16] „[...] ist es [...] für einen sozialen Assistenzroboter essenziell, ein Mindestmaß an Adaptivität in der Gestaltung des Nutzerinterfaces mitzubringen, um die potentiellen Nutzer nicht zu enttäuschen“.

Lösungsansatz Durch die Variabilität der Interaktion kann somit auch eine Barrierefreiheit geschaffen werden, da durch eine ständige Anpassung des Systems die beste Interaktion ausgewählt werden kann. Um solche Anpassungen durchzuführen, kann die sogenannte Softwarevariabilität genutzt werden. Hier werden die Interaktionen als Softwareprodukte mit validen Konfigurationen bezeichnet. Die Variabilitäten, welche unter anderem hardwarespezifisch als auch softwarespezifisch sind, werden durch einen Konfigurationszrozess zu einem Softwareprodukt mit einer validen Konfiguration. In dieser Arbeit vorrangegangenem großem Beleg [Grö21] ist ein solcher

Konfigurationsprozess konzipiert. Hier sind Regeln definiert, sodass ein valides System konfiguriert werden kann. Diese Regeln sind sehr grob gehalten und müssen bei Anpassungen von Interaktionen eventuell neu spezifiziert werden.

Hier besteht die Möglichkeit, dass durch Machine Learning (ML) der Konfigurationsprozess erstellt wird und der Roboter selbst während der Laufzeit erkennt, welche Interaktion er benötigt und wie evtl. diese angepasst werden kann. Ziel hierbei ist es, dass der Roboter selbstständig lernt, wann welche Interaktionsmöglichkeit genutzt werden soll und wie diese Interaktion an den Nutzer am besten angepasst werden kann.

Somit kann folgende wissenschaftlliche Frage gestellt werden:

Wissenschaftliche Frage Welche Schritte sind nötig, damit ein Assistenzroboter selbst lernt eine optimale Interaktion für einen Benutzer zu wählen, welche die Einschränkungen des Benutzers berücksichtigt?

Aufbau der Arbeit Um die wissenschaftliche Frage zu beantworten bilden sich weiter Teilziele:

- **Erfassung von sprachlichen Interaktionsmöglichkeiten bei der älteren Bevölkerung**
  Es gibt eine Vielzahl an Möglichkeiten, welche in der Interaktion zwischen Mensch und Roboter eingesetzt werden können. Dieses Ziel beschränkt sich auf die sprachliche Interaktion zwischen Mensch und Roboter. Es muss dabei erfasst werden, welche Schwierigkeiten die ältere Bevölkerung in der sprachlichen Kommunikation hat und wie damit umgegangen werden kann.

- **Analyse eines Deep-Learning-Modells**
  In diesem Ziel soll analysiert werden, wie die sprachlichen Einschränkungen in einem DL-Modell eingebracht werden können, sodass dieses Modell von alleine lernen kann, welche Interaktion mit dem Nutzer gewählt und angepasst wird.

- **Erstellung eines Konzeptes**
  Sobald ein DL-Modell analysiert wurde, kann ein Konzept erstellt werden, indem der Ablauf des zu erstellenden Systems gezeigt wird. Dabei soll definiert werden, wann die Änderung der Interaktion, sowie eine möglich Anpassung der Interaktion, realisiert wird.

- **Implementierung eines Prototypen**
  Mithilfe eines Computers und einem Intel© Neural Compute Stick 2 soll auf einem Segway Loomo Roboter ein Prototyp erstellt werden. Hierbei sind wichtige Teilabschnitte des Sourcecodes zu dokumentieren und zu erläutern.

- **Test des Prototypen**
  Um herauszufinden, ob dieses Konzept nutzbar ist, können erstellte Anforderungen an dieses Konzept getestet werden. Hier soll beschrieben werden, wann eine Anforderung erfüllt ist, sodass Ziele für jede Anforderung erstellt werden können. Dabei muss beschrieben werden, wie eine Anforderung getestet wurde. Zum Schluss soll das Ergebnis ausgewertet und interpretiert werden.

2 Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten aufgezählt und kurz beschrieben. Dabei wurden diese nach ihrem Inhalt unterteilt, ob diese mit Interaktionen, Machine Learning oder mit SAR in Verbindung stehen.

2.1 Interaktionen

Tanevska et al. [Tan+20] zeigen, wie sich die adaptive Interaktion eines Roboters auf den Benutzer auswirkt. Dabei ist das Ziel zu untersuchen, ob es einen zusätzlichen Nutzen bringen würde oder eher Unsicherheit und Unvorhersehbarkeit. Es wurde ein soziales adaptives Framework für einen humanoiden Roboter entworfen, welcher die affektiven und interaktiven Signale einer Person wahrnimmt und diese als Eingabe verwendet, um seine Interaktion an den Nutzer anzupassen. Hierbei wird die Interaktion zwischen einer Person mit einem adaptiven und einem nicht-adaptiven sozialen Roboter untersucht. Als Roboter wird der iCub benutzt. An den Probanden werden unterschiedliche Aufgaben gestellt, wobei erst das nicht-adaptive soziale System zum Einsatz kommt und danach das adaptive soziale System. Dabei wurde festgestellt, dass subjektiv 27 % der Personen keinen Unterschied zwischen den Systemen festgestellt haben. Weiterhin haben 46 % gesagt, dass die zweite Sitzung interaktiver sei. Diese 46 % teilen sich wiederum in 50 % nicht-adaptiv und 50 % adaptiv auf. Außerdem gaben 23 % an, in der zweiten Sitzung gelernt zu haben, wie man besser mit dem Roboter interagiert. Es wurde festgestellt, dass subjektiv gesehen kein Zusammenhang zwischen der Anpassungsfähigkeit des Roboters und seiner Sympathie oder überhaupt ein Bewusstsein der Teilnehmer für die Existenz eines Unterschieds in den Profilen ergab. Jedoch gibt es implizierte Ergebnisse, welche auf das Gegenteil hindeuten. Dabei zeigt sich, dass sich die Art der Interaktion mit dem Roboter, sowohl in Bezug auf die Häufigkeit als auch auf die Verwendung von Modalitäten, merklich änderte. Die Personen konnten mit dem adaptiven sozialen System interagieren, obwohl sie parallel eine Aufgabe erfüllen sollten.

Die Arbeit von Tanevska et al. zeigt, dass sich die adaptive Interaktion eines Roboters auf den Benutzer auswirkt. Dabei wird festgestellt, wie wichtig adaptive Interaktionen mit einem Roboter sind. Diese adaptive Interaktion kann für Benutzer mit bestimmten Einschränkungen übernommen werden, sodass sich die Interaktion an die Gegebenheiten des Benutzers anpassen kann.

Lee [Lee21] zeigt in dieser Studie, wie wichtig der Sprachrhythmus, sowie die emotionale Sprache des Roboters auf den Benutzer ist. Hier wird eine Studie durchgeführt, wie sich das Verhalten der Interaktion durch den Sprachrhythmus verändert. Dafür wurden zwei Experimente durchgeführt, für die der humanoide Roboter NAO benutzt wurde. Bei dem ersten Experiment mussten die Probanden eine Geschichte anhören, welche vom NAO erzählt wurde. Dabei sind unterschiedliche Sprachrhythmen genutzt worden (neutral, positiv, negativ). Anschließend mussten die Probanden Fragen zu der Geschichte beantworten. Das Ergebnis zeigt, dass die Probanden mehr richtige Antworten geben konnten, wenn der Roboter mit einer positiven oder negativen Stimme

gesprochen hat. Das zweite Experiment konzentriert sich auf die Eingabe des Roboters, unter anderem Selbstvorstellung, Begrüßung und Small-Talk. Dabei wurde die Emotion des Roboters anhand der Mimik des Benutzers angepasst, sowie der Sprachrhythmus an die Pausen zwischen den Sätzen. Die Interaktionszeit betrug ca. 5 Minuten. Das Ergebnis ist, dass die Probanden während der Sitzung hauptsächlich negative Emotionen widerspiegelten. Diese Studie zeigt, dass für eine erfolgreiche Mensch-Roboter-Interaktion z. B. in der Altenpflege, eine personalisierte Sprachprosodie des Roboters benötigt wird. Diese personalisierte Sprachprosodie hilft den Menschen, die vom Roboter bereitgestellten Informationen zu verstehen. Es wurde beobachtet, dass sich die negative Emotion des Roboters auf den Menschen widerspiegelt. Das zeigt, dass die negative Stimmung vom Roboter aus eher vermieden werden sollte, wenn dieser zur Unterstützung älterer Menschen eingesetzt wird.

Die Arbeit von Lee zeigt, dass nicht nur die Interaktion anhand von Einschränkungen gewählt werden muss, sondern auch die Interaktion selbst für den Benutzer personalisiert werden sollte. Besonders bei der älteren Gesellschaft können Einschränkungen in der Interaktion vorhanden sein. Dazu kommt, dass die ältere Gesellschaft vermutlich langsamer redet oder versteht. Dies muss bei der Konfiguration einer Interaktion beachtet werden.

Chivarov et al. [Chi+19] zeigen in ihrer Arbeit unterschiedliche Möglichkeiten der Interaktion mit einem Roboter, um diesen fernzusteuern. Dabei werden Methoden untersucht, welche auf Steuerung über Gesten, Sprachbefehle und webbasierten grafischen Benutzeroberflächen basieren. Es wird eine neue Version des Roboters Robco 19 vorgestellt, welche für die Experimente benutzt wird.

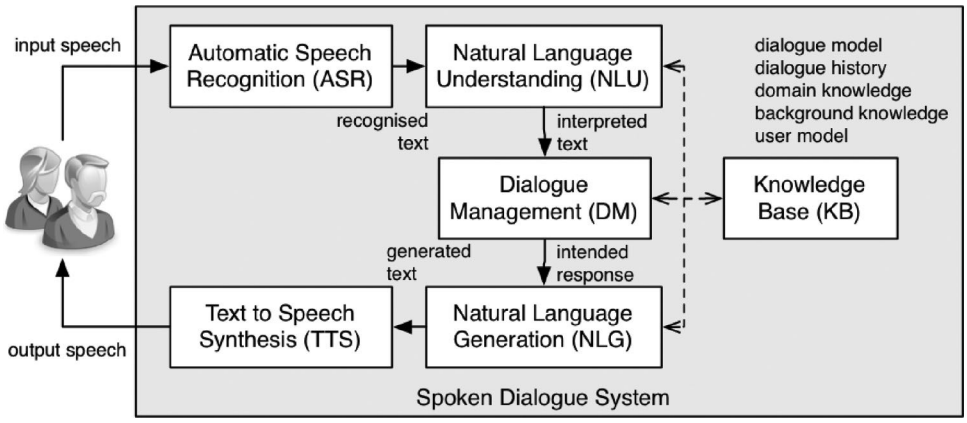


Abbildung 2.1 – Funktionale Architektur eines Dialogsystems mit natürlicher Sprache nach Russo et al. [Rus+19]

Russo et al. [Rus+19] untersuchen die Eigenschaften von Sprachdialogsystemen (siehe Abbildung 2.1) und ihre Rolle bei der Unterstützung der Mensch-Roboter-Interaktion und der Ermöglichung der Kommunikation zwischen SARs und Menschen mit Demenz. Dabei wird ein Überblick über bestimmte Dialogsysteme und deren zugrundeliegenden Technologien gezeigt und erläutert. Es wird eine Literaturrecherche durchgeführt, welche sich auf Sprachverarbeitungswerkzeugen für Menschen mit kognitiven Beeinträchtigungen sowie Menschen über einem Alter von 60 Jahren konzentriert. Hierbei werden Probleme und Herausforderung identifiziert, welche bei

der Verwendung von Konversationsagenten und sprachbasierten Schnittstellen für die Interaktion mit Menschen mit kognitiven Beeinträchtigungen verwendet wurden.

Sayago et al. [SNC19] zeigen in ihrer Arbeit Probleme, welche bei der Verwendung von Sprachassistenten mit älteren Menschen auftreten. Sie skizzieren diese Probleme und erläutern, dass diese Probleme in dieser Bevölkerungsgruppe erforscht und gelöst werden müssen. Dabei werden zu beantwortende Forschungsfragen entworfen.

Pradhan et al. [PLF20] führten eine Studie durch, bei der die Interaktion zwischen älteren Menschen und Sprachassistenten analysiert wurde. Dabei wurde festgestellt, dass die Sprachassistenz überwiegend für den Zugriff auf Online-Information genutzt wurde. Diese Online-Informationen waren überwiegend gesundheitsbezogene Informationen. Obwohl die Teilnehmer anfangs begeistert von den Gedächtnisunterstützungsfunktionen wie Erinnerungen waren, sind diese Funktionen geringer als erwartet zum Einsatz gekommen. Gründe sind unter anderem das Versagen der Technologie sowie das manuelle Einstellen der Erinnrung. Außerdem zeigt diese Arbeit auch vorläufige Einblicke, wie sprachbasierte Technologien einen barrierefreihen Zugang gewähren können.

2.2 Machine Learning

Temple et al. [Tem+17] zeigen in ihrer Arbeit, wie ML genutzt werden kann, um valide Softwaresysteme zu entwickeln. Ziel dieser Arbeit ist, dass passende Softwarekonfigurationen zu einem spezifischen Kontext zugeordnet werden können und umgekehrt. Dabei erstellen Temple et al. ein Beispiel für ein Trackingsystem mittels OpenCV. Dabei werden zwei Variabilitätsmodelle erstellt: ein kontextuelles Modell und ein softwarespezifisches Modell (siehe Abbildung 2.2). Diese beiden Modelle können auch in einem Variabilitätsmodell dargestellt werden. Um nun eine valide Konfiguration aufgrund des Kontext zu erstellen, ist eine sehr komplexe und laborintensive Aktivität notwendig. Die Idee ist hier einige Softwarekonfigurationen mit einigen Kontextkonfigurationen zu beobachten und zu lernen, welche Features des Kontext die Features der Software aktivieren oder deaktivieren. So können mit wenigen Konfigurationen ein DL-System antrainiert werden und kann nun berechnen, welches Feature aus dem Kontext eines Features aus der Software aktiviert oder deaktiviert wird. Zum Evaluieren wurde nun ein Tracking-System entwickelt. Dabei wurden unterschiedliche Kontextkonfigurationen erstellt. Für einen Vergleich wurde ein Lua Video Generator verwendet, welcher die Kontextkonfigurationen mit bestimmten Ergebnissen erstellt. Somit kann die Performance aufgrund von Präzision und Genauigkeit der Verfolgung von Interessensobjekten und Ausführungszeit gemessen und verglichen werden. Es zeigt sich, dass nach der Trainingsphase eine Genauigkeit von über 80 % erzielt werden kann.

Die Arbeit von Temple et al. kann für diese Arbeit ein Grundgerüst darstellen. Sie zeigt, wie mittels DL aus einem Kontext eine Softwarekonfiguration entstehen kann. Da ein Benutzer hier als Kontext gesehen werden kann, kann aufgrund der Person und dessen Einschränkung eine Interkation gewählt und angepasst werden.

Aljunid und Dh [AD20] In dieser Arbeit wird ein DL-Modell für ein Empfehlungssystem vorgestellt. Empfehlungssysteme basieren auf der Modellierung der Benutzerpräferenz für Elemente

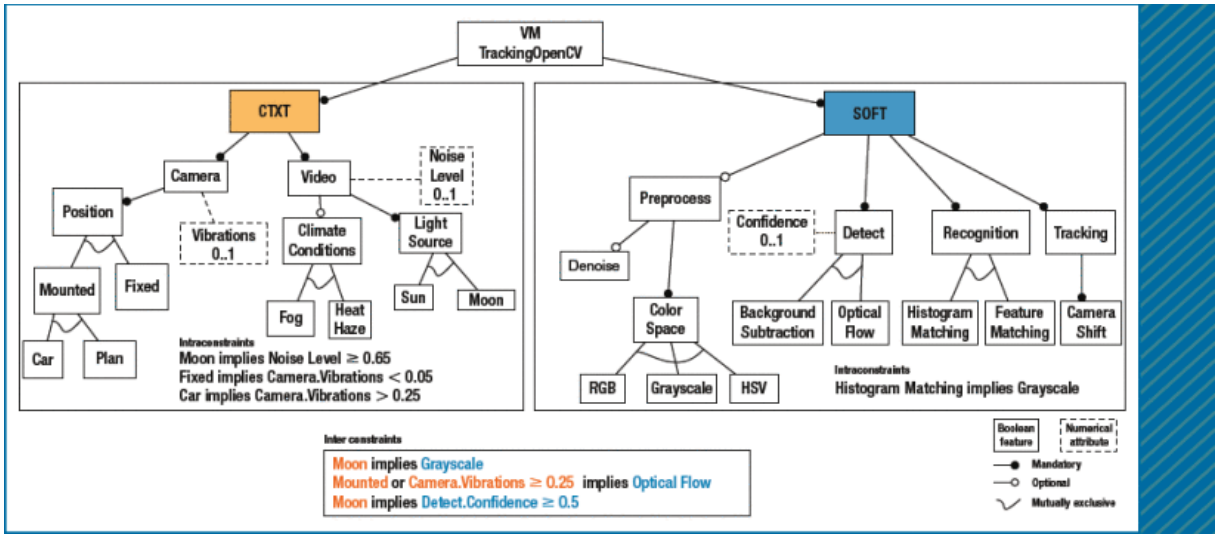


Abbildung 2.2 – Variabilitätsmodell mit Kontext- und Softwarekonfiguration. Mithilfe von Machine Learning sollen Beziehungen zwischen Kontext- und Software entdeckt werden. [Tem+17]

auf der Grundlage ihrer früheren Interaktionen (z. B. Bewertung und Klick), welche als Colaborativ Filtering (CF)-Technik bekannt ist. Diese Filterung hat jedoch Probleme bei der Matrixbewertung und Skalierbarkeit. Um den Fehler zu korrigieren, werden Matrixfaktorisierungstechniken genutzt. Doch bei sparsamen Daten, sowie Annäherung mit niedriger Bewertung, kann es hier immer noch zu Problemen kommen. Demzufolge wird eine DL-Methode für kollaborative Empfehlungssysteme vorgestellt und untersucht. Dabei wurde gezeigt, dass der vorgestellte Ansatz im Vergleich zu bestehenden Methoden verbesserte Ergebnisse liefert.

Hier wird ein anderes DL-Modell vorgeschlagen, welches auf kollaborativer Filterung beruht. Dieses wird in Abbildung 2.3 dargestellt. Dieser Ansatz kann adaptiert und ebenfalls für die Konfiguration einer Interaktion genutzt werden. Zu beachten ist, dass sich die Konfiguration der Interaktion durch Bewertungen anderer Benutzer verändern kann.

Bencomo [Ben20] erläutert wie wichtig es ist, dass sich Softwareproduktlinien dynamisch während der Laufzeit verändern sollten. Er weist auf seine vorigen Arbeiten, wie die nächsten Schritte für dynamische Softwareproduktlinien aussehen könnte, welche unter anderem ML verwenden.

In dieser Arbeit soll mit einem Softwarevariabilitätsmodell gearbeitet werden, welches sich in der Laufzeit ändern kann und somit ein neues Produkt entsteht. Die Arbeiten von Bencomo zeigen, wie sowas evtl. realisiert werden kann.

Bencomo et al. [BBI13] weisen darauf hin, dass die Bayes'sche Entscheidungstheorie zunehmend erfolgreich in der Psychologie und Biomedizin angewandt wird, um Entscheidungsprozesse unter Umgebungsvariabilität und Unsicherheit zu unterstützen. Sie zeigen wie diese Theroie in selbstadaptiven Systemen verwendet werden kann. Sie bauen somit ein Bayes'sches Netzwerk auf, welches sich dynamisch zur Laufzeit an neue Umgebungsvariabeln anpassen kann. Sie zeigen wie dieses Modell angewendet werden kann, um die Entscheidungsfindung in selbstadaptiven Systemen zu unterstützen.

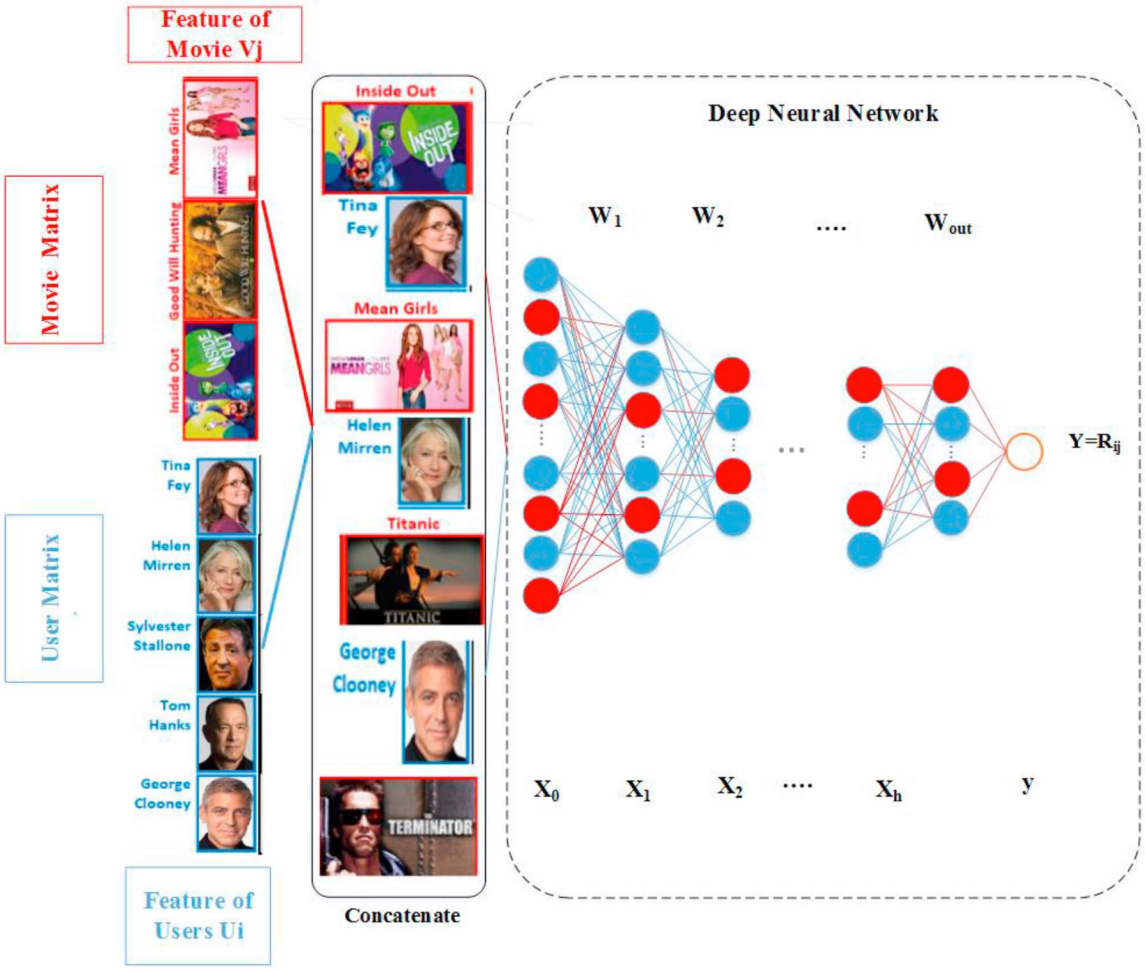


Abbildung 2.3 – Benutzer und Film Features kombiniert als Eingabeschicht für ein DL-Modell nach Aljunid und Dh [AD20]

Da die Interaktion zwischen Mensch und Roboter Adaptivität verlangt, um sich an den Benutzer anzupassen, kann die Arbeit von Bencomo et al. ein Grundgerüst liefern. Bei einer adaptiven Interaktion verändern sich ebenfalls Umgebungvariabeln, wenn sich der Anwender beispielsweise anders verhält. Somit ändern sich die Umgebungsvariabeln während der Laufzeit und das System muss sich demzufolge anpassen.

Garcia und Bencomo [GB19] zeigen wie partiell beobachtbare Markov-Entscheidungsprozesse als Laufzeitmodelle zur Unterstützung der Entscheidungsfindung in selbstadaptiven Systemen angewandt werden können. Hierbei wird das sogenannte Bayes'sche Lernen angewandt. Sie zeigen wie sich das Modell bei Änderungen von Umgebungsvariabeln aktualisiert, um fundierte Entscheidungen zu treffen.

In dieser Arbeit von Garcia und Bencomo wurde die Bayes'sche Theorie erneut aufgegriffen und für selbstadaptive Systeme verwendet. In Abbildung 2.4 wird gezeigt, wie das Modell mit dem Management-System interagiert. Da es sich in dieser Arbeit um ein adaptives System handelt, kann dieses Modell unterstützend sein.

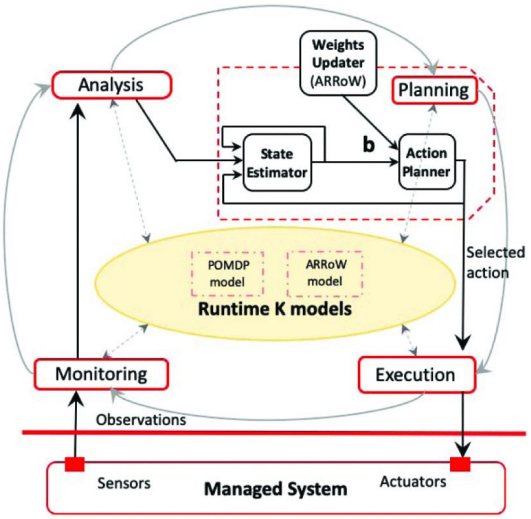


Abbildung 2.4 – Ablauf des K-Modells und dem Management-System [GB19]

Zhang et al. [Zha+20] fassen verschiedene DL-Techniken für sogenannte Empfehlungssysteme zusammen. Sie zeigen dabei, welche DL-Techniken existieren und welche Probleme auftreten. Außerdem erläutern sie, warum DL in Empfehlungssystemen immer mehr Einsatz findet.

In dieser Arbeit soll eine adaptive Interaktion erstellt werden. Da die entsprechende Interaktionsmöglichkeit eine Empfehlung darstellen kann, könnte auf Modelle von Empfehlungssystemen bei adaptiven Interaktionen zurückgegriffen werden. Dabei können DL-Techniken helfen, die bestmögliche Interaktionsmöglichkeit zu finden.

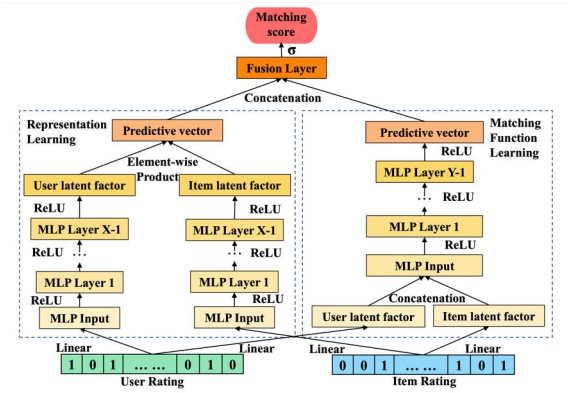


Abbildung 2.5 – Architektur des CF-DL-Modell nach Deng et al. [Den+19]

Dent et al. [Den+19] schlagen ein Framework namens Deep-Collaborative-Filtering für Empfehlungssysteme vor, welches die Vorteile von zwei kollaborativen Filterverfahren kombiniert. In Abbildung 2.5 wird die Architektur dargestellt. Dabei werden Experimente mit öffentlichen Datensätzen durchgeführt, welche die Wirksamkeit des Frameworks zeigt.

Shambour [Sha21] erstellt ein DL basierenden Algorithmus für Empfehlungssysteme, welcher die Empfehlung mithilfe mehrerer Kriterien berechnet. Es wird dabei Deep-Autoencoder eingesetzt, um die nichttrivialen, nichtlinearen Beziehungen zwischen Benutzern in Bezug auf Präferenz mit mehreren Kritieren auszunutzen und bessere Empfehlungen zu berechnen.

2.3 Soziale Assistensroboter

Carros et al. [Car+20] führen eine Studie durch, in der in einer Pflegeeinrichtung ein SAR für physische und psychische Trainingseinheiten genutzt wurde. Es wurden dabei Interviews mit Bewohnern und Pflegekräften geführt, um zu analysieren, wie gut die Interaktionen erfolgten. Das Ergebnis zeigt, dass die Bewohner positiv an den Trainingseinheiten teilnahmen. Somit zeigen sie, dass SARs in Pflegeeinrichtungen schon genutzt werden können, unter der Bedingung, dass eine moderierende Person benötigt wird, welche den Roboter steuert.

Die Studie von Carros et al. zeigt, dass SARs schon eingesetzt werden können, aber für einen Laien noch extrem angepasst werden müssen.

Abbildung 2.6 – MobiKa [Gra+19]

Graf et al. [Gra+19] führen einen neuen mobilen kostengünstigen SAR vor, welcher für die multimodale Mensch-Roboter-Interaktion optimiert ist. Dieser Roboter soll ein Kommunikationsassistent zwischen den vernetzten Geräten und dem Menschen darstellen. Dieser Roboter wurde dabei in realen Szenarien in einem Seniorenwohnheim getestet.

Miseikis et al. [Mis+20] stellen den Roboter Lio vor. Dies ist ein Roboter, welcher speziell für die Mensch-Computer-Interaktion und Aufgaben des persönlichen Pflegeassistenten entwickelt wurde. Der Roboter wird bereits in mehreren Gesundheitseinrichtungen eingesetzt und arbeitete dabei fast vollständig autonom. Lio hat dabei eine gute Akzeptanz beim Pflegepersonal und den Patienten. Während der COVID-19-Pandemie wurde dieser Roboter angepasst, um weitere Aufgaben wie Desinfektion oder Fernerkennung von erhöhter Körpertemperatur auszuführen.

2.4 Zusammenfassung

In diesem Kapitel wurden einige verwandte Arbeiten genannt, kurz beschrieben und inhaltlich auf die Bereiche Interaktion, ML und SAR Im Bereich ML ist zu erkennen, dass viel im Feld der Empfehlungssysteme geforscht wird, was wiederum die Prägnanz von solchen Systemen herausstellt. Mit dieser Erkenntnis wird auch in dieser Arbeit auf den Stellenwert und die Vorteile von Empfehlungssystemen eingegangen und eines eingesetzt.

3 Grundlagen

Zunächst werden in diesem Kapitel fundamentale Grundlagen erklärt. Begriffe wie Softwarevariabilität (SV), Deep Learning (DL) und Sozialer Assistenzroboter (SAR) sollen dabei mehr erläutert werden.

3.1 Soziale Assistenzroboter

In der heutigen Zeit sind Roboter kaum noch wegzudenken. Sie begleiten uns in der Forschung, im Haushalt, in der Industrie und im Gesundheitswesen. Es gibt mehrere Definitionen, welche Roboter beschreiben. Eine stammt aus den VDI-Richtlinien [Vdi90] die wie folgt lautet: „Industrieroboter sind universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolge und Wegen bzw. Winkeln frei (d. h. ohne mechanischen bzw. menschlichen Eingriff) programmierbar und gegebenenfalls sensorgeführt sind. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar und können Handhabungs- und/oder Fertigungsaufgaben ausführen."

Doch diese Definition beschreibt direkt Industrieroboter, wobei diese Arbeit sich auf den Bereich der Roboter des Gesundheitssystems bezieht. Dadurch wird die Definition hinfällig, da Industrieroboter nicht im Gesundheitssystem genutzt werden. Sivaeshu et al. [SRR17] bietet eine allgemeinere Definition für Roboter an:

Definition 1: Roboter

„A robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks." [SRR17]

3.1.1 Klassifizierung von sozialen Assistenzrobotern

Da die Definition 1 sehr allgemein gehalten ist und auf eine Vielzahl von Robotern zutrifft, klassifiziert Heerink et al. [Hee+10] Roboter wie in Abbild 3.1. Es gibt demzufolge die Klassen *Industrieroboter*, *nicht soziale Assistenzroboter*, SAR und *Nicht-Assistenzroboter*, wobei Heerink et al. [Hee+10] SAR noch in *Serviceroboter* und *Companionroboter* unterteilt.

Industrieroboter sind Roboter, die in der Industrie verwendet werden und können nicht sozial interagieren. In Abbildung 3.2a wird solch ein industrieller Roboter von der Firma KUKA Roboter GmbH abgebildet.

Nicht soziale Assistenzroboter können ebenfalls nicht sozial interagieren. Solche Roboter werden beispielsweise in Haushalten verwendet. In Abbildung 3.2b wird ein Staubsaugroboter von Vorwerk dargestellt, welcher assistiv den Boden reinigen kann, jedoch nicht sozial interagieren kann.

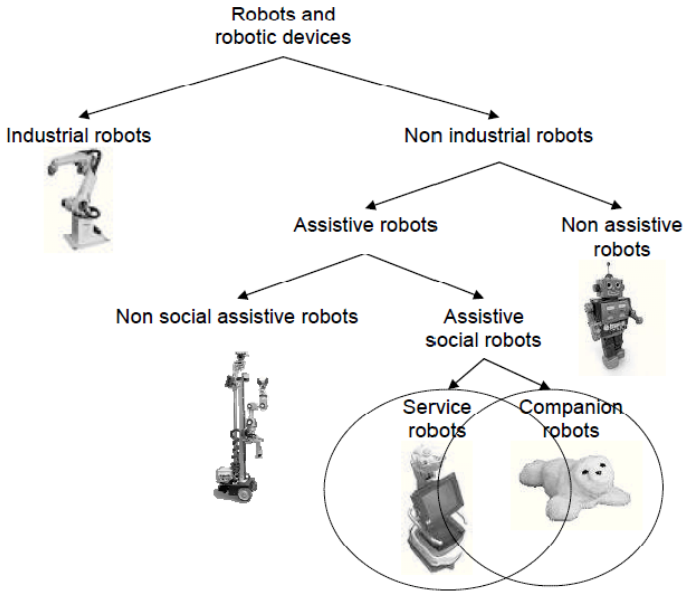


Abbildung 3.1 – Klassifikation nach Heerink et al. [Hee+10]

Ein *Nicht-Assistenzroboter* wird in Abbildung 3.2c gezeigt. Dies sind Roboter, die unter anderem spielerisch genutzt werden können. In dieser Abbildung ist ein LEGO Mindstorm Roboter dargestellt, welcher als Spielzeug definiert ist.

SAR werden in den Abbildungen 3.2d und 3.2e gezeigt. Diese sind assistiv und sozial in der Lage mit Benutzern zu interagieren. Abbildung 3.2d beinhaltet einen *Companionroboter* namens Paro, welcher nicht mobil ist, während Abbildung 3.2e einen *Serviceroboter* namens Pepper darstellt, welcher mobil ist.

3.1.2 Soziale Assistenzroboter im Gesundheitswesen

Roboter sind in der Lage die unterschiedlichsten Aufgaben im Gesundheitswesen zu bewerkstelligen. Dabei werden nach Papadopolous et al. [PKA18] physische Aufgaben durchgeführt, wie zum Beispiel das Heben einer Person oder das Assistieren beim Aufstehen von Stizenden oder Liegenden, beim Anziehen helfen sowie das Abholen bzw. Transportieren von Objekten. Nach Broadbent et al. [Bro+16] gehören zu den physischen Aufgaben auch die Überwachung von Vitalzeichen und Emotionen von Personen. Darragh et al. [Dar+17] bezeichnet dies als die Person „im Auge behalten“. Außerdem können Roboter auf bestimmte Ereignisse reagieren. So können Roboter einen Alarm weiterleiten, wenn die zu betreuende Person hingefallen ist oder um die Person an bestimmte Medikamente zu erinnern.

Laut Darragh et al. [Dar+17] können Roboter auch psychische Aufgaben erledigen, wie das beruhigen von Demenzkranken sowie kognitiv anders Erkrankten, sollten diese in Aufregung geraten. Broadbent et al. [Bro+12; Bir+16; Dar+17] trauen Robotern zu, soziale Interaktionen zwischen Bewohnern einer Pflegeeinrichtung zu fördern, sowie Gruppenaktivitäten oder die Kommunikation mit Demenzkranken zu erleichtern.

(a) Industrieller Roboter von KUKA Roboter GmbH[1]

(b) Staubsaugroboter von Vorwerk[2]

(c) LEGO Mindstormroboter[3]

(d) Companionroboter Paro[4]

(e) Serviceroboter Pepper[5]

Abbildung 3.2 – Arten von Robotern

[1] `http://jorgeatk.blogspot.com/2017/09/robots-kuka-nigel-stanford-automatica-4k.html` (aufgerufen am 06.09.2022)
[2] `http://commons.wikimedia.org/wiki/File:Vorwerk_Kobold_VR100.JPG` (aufgerufen am 06.09.2022)
[3] `http://www.rolandodapiazzola.it/dw/lib/exe/detail.php?id=start&media=4-modles-lego-mindstorms-ev3.png` (aufgerufen am 06.09.2022)
[4] `https://www.flickr.com/photos/7589404@N04/2254307941` (aufgerufen am 06.09.2022)
[5] `https://upload.wikimedia.org/wikipedia/commons/e/ee/Pepper_the_Robot.jpg` (aufgerufen am 06.09.2022)

3.1.3 Beispiele von Assistenzrobotern

Abbildung 3.3 – Pepper[6]

Pepper ist ein humanoider Roboter, welcher laut der Klassifikation nach Heerink et al. [Hee+10] ein SAR ist. Dieser beherrscht die Funktionen als Serviceroboter sowie als Companionroboter ausüben. Nach Pandey und Gelin [PG18] begann die Produktion dieses Roboters 2014. Der Roboter ist in der Lage, Körpersprache zu zeigen, seine Umgebung wahrzunehmen und mit dieser zu interagieren und sich fortzubewegen. Weiterhin kann Pepper Emotionen anhand der Mimik und dem Tonfall von Personen analysieren. Damit ist er in der Lage, Interaktionen anzuregen und multimodal mit einer Person zu kommunizieren. Laut dem Blog von Softbank Robotics [Ald20] soll Pepper helfen, Patienten zu überwachen, Personen zu unterhalten, den Stress des medizinischen Personals zu reduzieren und bei der sozialen Distanzierung zu helfen.

Care-O-Bot lässt sich ebenfalls nach Heerink et al. [Hee+10] als SAR einordnen und ist ein humanoider Roboter. Dabei ist er ebenfalls in der Lage die Funktionen als Serviceroboter sowie als Companionroboter zu übernehmen. Laut IPA [Pro15] wurde mittlerweile die vierte Generation des Care-O-Bot entwickelt, wobei nach Kittmann et al. [Kit+15] die erste Generation 1999 und die vierte 2015 entwickelt wurde. Dieser verfügt über einen Touchscreen an seinem Kopf, sowie mehrere Mikrofone und Lautsprecher zur multimodalen Kommunikation. Der Roboter kann wie Pepper Patienten überwachen und lädt zur Interaktion ein. Laut Kittmann et al. [Kit+15] soll der Roboter in unterschiedlichen Umgebungen seinen Einsatz finden. Dabei kann er in häuslichen, sowie in hospitalen Umgebungen Hilfe leisten.

Abbildung 3.4 – Care-O-Bot [Pro15]

3.2 Deep Learning

In der Welt der Künstlichen Intelligenz (KI) ist ML in den letzten Jahrzehnten ein immer populäreres Teilgebiet geworden, wodurch laut Bishop [Bis06] ein System eigenständig sinnvolle Beziehungen und Muster anhand von Beispielen und Beobachtungen lernen soll. Solche KIs werden derzeit auch schon eingesetzt, um beispielsweise trainierbare Assistenzsysteme zu entwickeln, welche sich an individuelle Benutzerpräferenzen anpassen, wie Fischer et al. [Fis+20] zeigen.

Deutliche Fortschritte im Bereich des ML war die Entwicklung von künstlichen neuronalen Netzwerken (KNNs) hin zu immer tieferen neuronalen Netzarchitekturen, die laut Goodfellow et al. [GBC16; LBH15] als DL zusammengefasst werden.

Janiesch et al. [JZH21] teilen das Lernen von ML-Systemen in drei Bereiche auf: überwachtes Lernen, unüberwachtes Lernen und Verstärkungslernen.

Überwachtes Lernen Für überwachtes Lernen ist ein Trainingsdatensatz notwendig, welcher Eingabedaten und die dazugehörigen Zielwerte der Ausgabe enthält. Mit diesen Paaren von Eingabe-

[6] `https://upload.wikimedia.org/wikipedia/commons/e/ee/Pepper_the_Robot.jpg` (aufgerufen am 06.09.2022)

und Ausgabedaten können offene Parameter des ML-Modells kalibriert werden. Sobald das Modell trainiert wurde, können Vorhersagen zu weiteren Eingabedaten getroffen werden. [JZH21]

Unüberwachtes Lernen Bei unüberwachtes Lernen soll das lernende System Muster erkennen, ohne zuvor bestimmte Spezifikationen zu kennen. Die Trainingsdaten bestehen somit nur aus Eingabedaten, mit dem Ziel, interessante Strukturinformationen zu finden. [JZH21]

Verstärkungslernen Bei Verstärkungslernen wird für das Training anstatt Ein- und Ausgabedaten der aktuelle Zustand des Systems, ein spezifiziertes Ziel, eine Liste zulässiger Aktionen und Umgebungsbedingungen bereitgestellt. Das ML-Modell erreicht sein Ziel durch das sogenannte „trial and error“ Prinzip. [JZH21]

Nach Janiesch et al. [JZH21] ist DL sehr interessant, da es für alle drei Bereiche gut umsetzbar ist.

3.2.1 Funktionsweise von Deep Learning

Laut Janiesch et al. [JZH21] sind neuronale Netze inspiriert vom Prinzip der Informationsverarbeitung in biologischen Systemen, da sie aus mathematischen Darstellungen verbundener Verarbeitungseinheiten, welche als künstliche Neuronen bezeichnet werden, bestehen. Ähnlich wie Synapsen in einem Gehirn übertragen die Verbindungen zwischen den Neuronen Signale, die durch eine Gewichtung verstärkt oder abgeschwächt werden können. Ein Neuron verarbeitet nur dann ein Signal, wenn ein bestimmter Schwellwert überschritten wird, die durch eine Aktivierungsfunktion bestimmt wird. In den meisten Fällen werden KNNs in unterschiedlichen Schichten organisiert, wobei es eine Eingabeschicht geben kann, welche für die Eingabe von Daten zuständig ist, und eine Ausgabeschicht, die das Endergebnis produziert. Zwischen diesen beiden Schichten können auch weitere Schichten enthalten sein, welche Signale weiter verarbeiten.

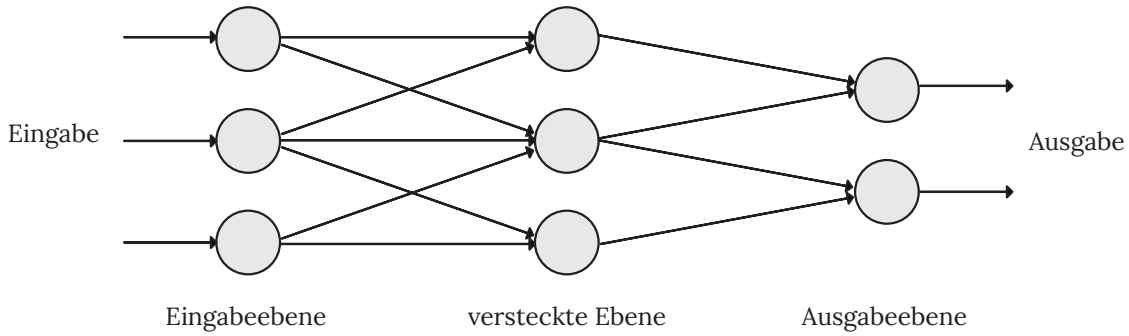


Abbildung 3.5 – Neuronales Netzwerk mit drei Schichten.

Abbildung 3.5 zeigt ein solches neuronales Netzwerk mit drei Schichten, wobei eine Eingabeschicht, eine versteckte Schicht und eine Ausgabeschicht enthalten ist.

Laut Janiesch et al. [JZH21] bezieht sich die Kernfähigkeit von DL darauf, dass sehr tief verschatelte Netzarchitekturen genutzt werden, welche mehr als nur eine verborgene Schicht aufweisen. Außerdem werden fortgeschrittene Neuronen genutzt, die in der Lage sind, fortgeschrittene Operationen oder mehrere Aktivierung verwenden zu können, anstatt einer einfachen Aktivierungsfunktion.

Laut LeCun et al. [LBH15] ist DL besonders nützlich in Bereichen mit großen und hochdimensionalen Daten, wie bei der Verarbeitung von Text-, Bild-, Video-, Sprach- und Audiodaten. In diesen Bereichen übertreffen tiefe neuronale Netze meist sogenannte flache ML-Algorithmen. Nach Zhang und Ling [ZL18] können bei begrenzter Verfügbarkeit von Trainingsdaten ein flacher ML-Algorithmus überlegenere Ergebnisse liefern.

3.2.2 Deep Learning Techniken

Es gibt unterschiedliche Techniken, die auf den drei Lernbereichen aufbauen. In diesem Abschnitt werden einige Techniken genannt und kurz erläutert.

Multi-Layer Perceptron (MLP) Dies ist ein neurales Feed-Forward-Netzwerk, welches neben der Eingabe- und Ausgabeschicht mindestens eine verborgene Schicht enthält. Dabei kann ein Perzeptron eine beliebige Aktivierungfuntkion verwenden, wodurch nicht unbedingt eine binäre Klassifikation dargestellt wird. Solche Systeme sind auch bekannt als universelle Approximatoren. [Zha+20]

Autoencoder Diese Technik gehört zu dem unüberwachten Lernen, wobei das Modell seine Eingabedaten in der Ausgabeschicht rekonstruiert. Hierfür werden drei Schichten im neuronalen Netz verwendet, wobei die mittlere verborgene Schicht als typische Merkmalsdarstellung der Eingabedaten verwendet wird. [JZH21; Che+12; GBC16]

Convolutional Neural Network (CNN) Dies ist eine spezielle Art von Feed-Forward-Netzwerken. Dabei existieren neben der Ein- und Ausgabeschicht die sogenannte Faltungsschicht gefolgt von einer Vereinigungsschicht. Globale und lokale Merkmale können von diesem System erfasst und die Effizienz und Genauigkeit erheblich verbessert werden. [Zha+20]

Recurrent Neural Network (RNN) Im Vergleich zu Feed-Forward-Netzen können bei diesen Netzen die Neuronen auch Verbindungen zu der selben oder der vorigen Schicht haben. Demzufolge können Schleifen entstehen, sodass sich das Netzwerk an frühere Berechnung erinnern kann. [Zha+20; GBC16]

Restricted Boltzmann Machine Dies ist ein zweischichtiges neuronales Netzwerk, welches auf RNN aufbaut. Hierbei gibt es eine sichtbare und eine verborgene Schicht, wobei die Neuronen keine Verbindung zu Neuronen in ihrer eigenen Schicht haben können.
[Zha+20]

3.2.3 Kaltstartproblematik

Empfehlungssysteme (RSs) sind Systeme, in denen dem Nutzer Artefakte, wie z.B. Filme oder Produkte, vorgeschlagen werden können. In diesen Systemen werden oft laut Yuan et al. [Yua+16] flache ML-Algorithmen verwendet wie CF oder kontextbasierte Filterung.

Bei kontextbasierten Filterungstechniken werden laut Aggarwal [Agg16] die Ähnlichkeiten zwischen den Profilen der Artefakte und dem Profil der Benutzer gemessen oder prognostiziert. Für

die Messung werden dabei geeignete Funktionen genommen, während bei den Vorhersagen ML verwendet wird.

Im Gegensatz zu kontextbasierten Methoden, die Artefakte empfehlen, die denen des Zielbenutzers in der Vergangenheit ähneln, nutzen laut Su et al. [SK09; Agg16] CF-Methoden die Präferenzen anderer ähnlicher Benutzer, um dem Zielbenutzer Empfehlungen zu geben.

Laut Yuan et al. [Yua+16] ist eins der größten Herausforderung von CF-basierten RS die sogenannte Kaltstartproblematik. Hierbei kann das System wegen Mangel an Informationen neuen Nutzern keine Artefakte oder Nutzern keine neuen Artefakte vorschlagen. Das RS ist hierbei nicht in der Lage festzustellen, in welcher Beziehung die Benutzer oder Artefakte stehen.

3.3 Softwarevariabilität

Ursprünglich wurde Software sehr statisch entwickelt oder es wurde direkt individuell an den Kunden angepasst. Laut Seidl et al. [SSA14] gab es die sogenannte Standardsoftware, welche eine breite Spanne an Anforderung abdeckt, jedoch keine individuelle Entwicklung benötigt und die individuelle Softwareentwicklung, welche Software individuell nur für den Kunden erstellt. In der heutigen Zeit deckt die Standardsoftware nicht genügend Anforderungen des Kunden ab und die individuelle Softwareentwicklung benötigt einen extrem hohen Entwicklungsaufwand, die zu extrem hohen Kosten führt.

Abhilfe schafft hier die SV, welche laut Pohl et al.[PBL05] den Vorteil bringt durch Ähnlichkeit der jeweiligen Produkte die Kosten und den Entwicklungsaufwand niedrig zu halten.

3.3.1 Variabilität in Softwarevariabilität

SV zeichnet sich durch eine Reihe eng verwandter Softwaresysteme aus, welche auch *Softwarefamilien* genannt werden. Seidl et al. [SSA14] beschreiben Softwarefamilien mit Gemeinsamkeiten und Variabilitäten. Laut Seidl et al. [SSA14; PBL05] sind Gemeinsamkeiten Eigenschaften und Funktionen, die Teil jeder Anwendung aus einer Softwarefamilie sind. Variabilitäten wiederum sind Eigenschaften und Funktionen, die nur von einer Teilmenge der Anwendungen aus einer Softwarefamilie genutzt werden.

Pohl et al. [PBL05] definieren mehrere Begriffe, um die Variabilität in Softwarefamilien besser zu verdeutlichen. Hierfür betrachten Sie folgende Fragen: „Was variiert?", „Warum variiert es?" und „Wie variiert es?". Die erste Frage wird durch das *Variabilitätssubjekt* beantwortet.

Definition 2: Variabilitätssubjekt

„A variability subject is a variable item of the real world or a variable property of such an item." [PBL05]

Die zweite Frage beantworten Pohl et al. [PBL05] so, dass es die unterschiedlichsten Gründe gibt, warum etwas variiert. Dies könnten unterschiedliche Anforderungen der Stakeholder sein, bis hin zu Abhängigkeiten von anderen variablen Eigenschaften und Funktionen.

Die dritte Frage wird mit dem Begriff *Variabilitätsobjekt* beantwortet.

Definition 3: Variabilitätsobjekt

„A variability object is a particular instance of a variability subject.“ [PBL05]

Um den Unterschied zwischen Variabilitätssubjekt und Variabilitätsobjekt zu untermauern, geben Pohl et al. [PBL05] mehrere Beispiele an. Das erste Beispiel ist die Farbe eines Autos. Hierbei ist das Variabilitätssubjekt die Farbe, während die Variabilitätsobjekte explizite Farben wie blau, rot oder grün sind.

Als zweites Beispiel benennen Pohl et al. [PBL05] die Bezahlmethode in Geschäften. Hierbei stellt die Bezahlmethode das Variabilitätssubjekt dar, während die Bezahlung durch Kreditkarte, durch Rechnung oder durch Barzahlung die Variabilitätsobjekte sind.

3.3.2 Realisierung durch Softwarevariabilität

Um die Variabilität in Softwarefamilien verwalten zu können, benötigt man laut Seidl et al. [SSA14] ein *Konfigurationswissen*, das alle Regeln enthält, um Kombinationen von Variablen und gemeinsamen Artefakten zu validieren. Diese Artefakte können laut Seidl et al. [SSA14] in zwei Ebenen aufgeteilt werden: die Konzeptionsebene und die Realisierungsebene.

Die Konzeptionsebene beinhaltet laut Seidl et al. [SSA14] wirtschaftliche sowie logische Bedenken, die die Konfigurationsoptionen einschränken, während auf der Realisierungsebene technische Inkompatibilitäten sowie Einschränkungen in der zugrunde liegenden technologischen Plattform enthalten sind.

Diese Ebenen können nach Czarnecki und Eisenecker [CE00] in zwei Bereiche differenziert werden: der Problembereich und der Lösungsbereich. Bestimmte Anforderungen und konzeptionelle Konfigurationen, welche in einem *Variabilitätsmodell* erfasst werden können, sind im Problembereich enthalten. Somit spiegelt der Problembereich die Konzeptionsebene wider, während der Lösungsbereich analog die Realisierungsebene widerspiegelt. Der Lösungsbereich umfasst dabei die Realisierungsobjekte für alle Softwaresysteme, welche Quellcode, Design oder Dokumentationen sind.

Um ein valides Softwaresystem aus einer Softwarefamilie zu erstellen, muss das Konfigurationswissen sowohl im Lösungsbereich, als auch im Problembereich enthalten sein. Durch ein *Variabilitätsrealisierungsmechanismus* kann nun mit dem Lösungsbereich, Problembereich und dem Konfigurationswissen ein Softwaresystem konfiguriert werden.

3.3.3 Variabilitätsmodell

Um das konzeptionelle Konfigurationswissen darzustellen, werden Variabilitätsmodelle verwendet, welche grafisch oder textuell sein können. Diese Variabilitätsmodelle zeigen laut Pohl et al. [PBL05] an, welche Kombinationen gültig sind, unabhängig von technischen oder wirtschaftlichen Gründen.

Hierbei gibt es etliche unterschiedliche Variabilitätsmodelle, wobei die beliebtesten laut Seidl et al. [SSA14] das Feature-Modell, Entscheidungsmodell und das orthogonale Variabilitätsmodell sind. In dieser Arbeit wird lediglich das Feature-Modell genutzt, weswegen dieses genauer erläutert wird.

Feature-Modell Dieses Modell ist laut Kang et al. [Kan+90; CE00] eine grafisch hierarchische Anordung von Features in einer Baumstruktur, wobei alle sichtbaren Anforderungen vom Benutzer erfasst werden. Features müssen dabei so markiert werden, dass diese anzeigen, ob ein Feature notwendig oder optional ist. Neben der Notwendigkeit von Features können auch Alternativen oder alternative Gruppen angezeigt werden, wobei man zwischen den Features wählen kann. Baumübergreifende Abhängigkeiten können auf unterschiedliche Art und Weise angegeben werden, wobei Seidl et al. [SSA14] solche Abhängigkeiten in Formeln darstellen, während Benavides et al. [Ben+13] solche direkt in der Baumstruktur mit angeben. In der Abbildung 3.6 wird ein Feature-Modell gezeigt. Dabei sind die Features **Engine** oder **Movement** notwendige Feature und müssen immer enthalten sein, während **Webservice** und **Detection** nicht enthalten sein müssen. Im Feature **Movement** gibt es drei weitere Features, wobei nur eins gewählt werden darf. Im Feature **Detection** gibt es ebenfalls drei weitere Features, wobei hier mindestens eins gewählt werden muss, wenn das Feature **Detection** enthalten sein soll.

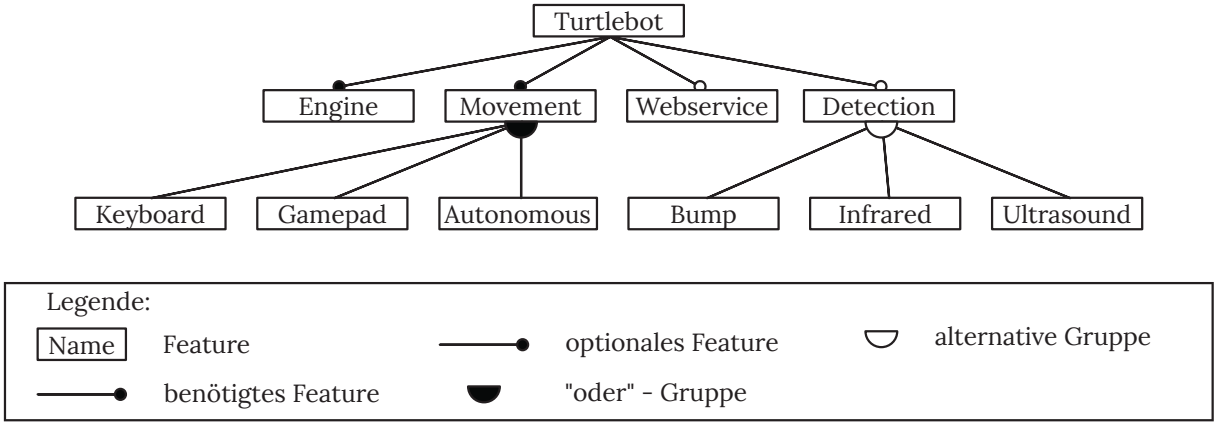


Abbildung 3.6 – Beispiel eines Feature-Modells nach Seidl et al. [SSA14]

3.3.4 Variabilitätsrealisierungsmechanismus

Ein Variabilitätsrealisierungsmechanismus ist notwendig, um aus dem Konfigurationswissen, dem Problembereich und dem Lösungsbereich ein bestimmtes Softwaresystem einer Softwarefamilie zu erzeugen. Laut Seidl et al. [SSA14] existieren drei Arten von Variabilitätsrealisierungsmechanismen: *Annotativer Variabilitätsrealisierungsmechanismus*, *Kompositionaler Variabilitätsrealisierungsmechanismus* und *Transformationaler Variabilitätsrealisierungsmechanismus*.

Annotativer Variabilitätsrealisierungsmechanismus Die Durchführung eines annotativen Variabilitätsrealisierungsmechanismus wird in Abbildung 3.7 verdeutlicht. In diesem Ansatz existiert ein 150 %-Modell, welches mehr Realisierungsobjekte enthält als nötig. Von diesem Modell werden Realisierungsobjekte entfernt, um ein einzelnes Softwaresystem einer Softwarefamilie zu generieren.[SSA14; Sch+12; KAK08]

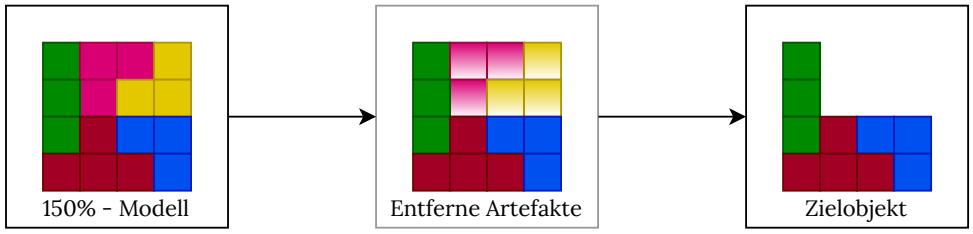


Abbildung 3.7 – Annotativer Variabilitätsrealisierungsmechanismus nach Seidel et al [SSA14]

Kompositionaler Variablitätsrealisierungsmechanismus In Abbildung 3.8 wird gezeigt, wie der kompositionale Variabilitätsrealisierungsmechanismus durchgeführt wird. Es existiert ein *Kernmodell*, welches alle Realisierungsobjekte zusammenfasst, die in allen Mitgliedern einer Softwarefamilie benötigt werden. Weitere Realisierungsobjekte werden dem *Kernmodell* hinzugefügt, um unterschiedliche Varianten von Softwaresystemen einer Softwarefamiele zu erstellen. [SSA14; KAK08; Sch+12]

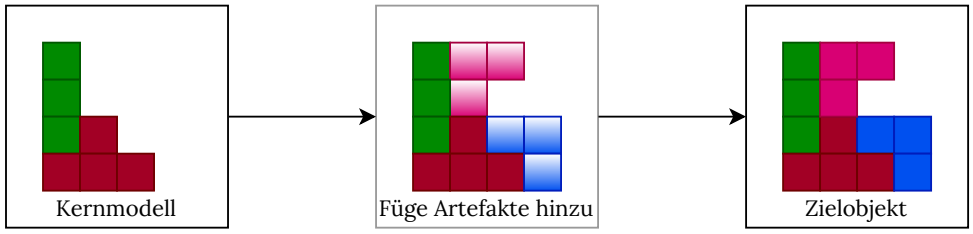


Abbildung 3.8 – Kompositionaler Variablitätsrealisierungsmechanismus nach Seidl et al. [SSA14]

Transformationaler Variabilitätsrealisierungsmechanismus Die Durchführung des transformationalen Variabilitätsrealisierungsmechanismus wird in Abbildung 3.9 gezeigt. Es existiert ein *Basismodell*, welches ein bestehendes Softwaresystem einer Softwarefamilie ist. Dieses wird in ein anderes Softwaresystem transformiert. Hierbei kann das *Basismodell* willkürlich gewählt werden. Es können Realisierungsobjekte hinzugefügt, modifiziert oder entfernt werden.[SSA14; Sch+12]

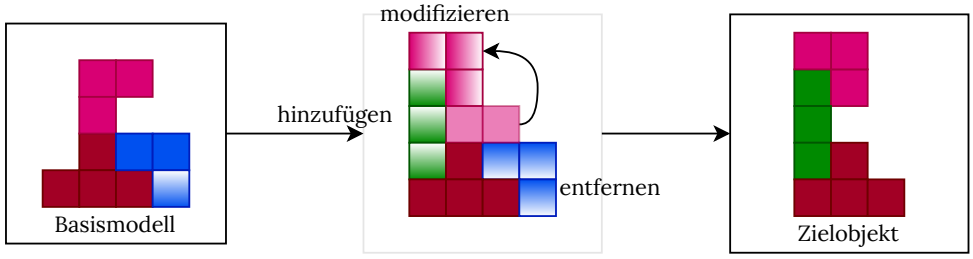


Abbildung 3.9 – Transformationaler Variabilitätsrealisierungsmechanismus nach Seidl et al. [SSA14]

3.3.5 Softwarevariabilität während der Laufzeit

Um eine gute Adaptivität zu gewährleisten, muss sich laut Quinton et al. [Qui+21] ein Softwareprodukt während seiner Laufzeit rekonfigurieren und anpassen. In Abbildung 3.10 wird die Ver-

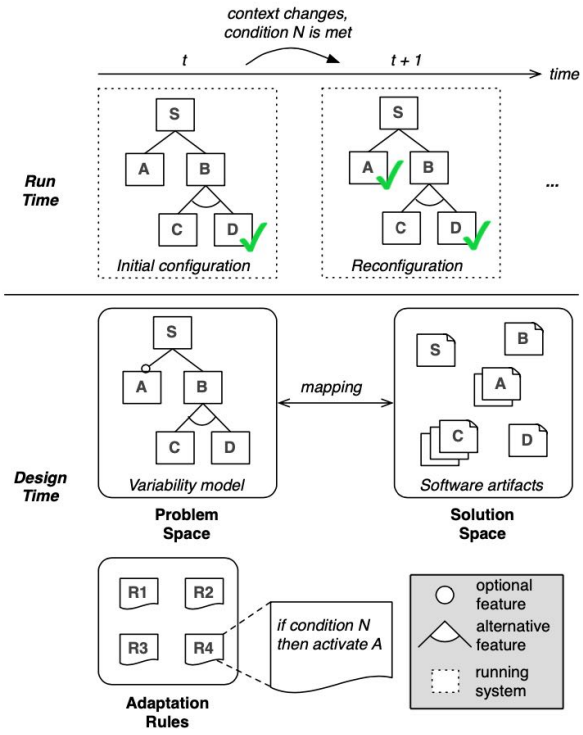


Abbildung 3.10 – Veränderung der Konfiguration während der Laufzeit nach Quinton et al. [Qui+21]

änderung der Konfiguration während der Laufzeit gezeigt. In diesem Beispiel wird ein Feature während der Laufzeit hinzugefügt, wodurch es zu einer neuen validen Konfiguration kommt. Um diese Veränderung zu bewerkstelligen muss der Problembereich dem Lösungsbereich zugeordnet werden, wobei ein Feature aus dem Problembereich durch mehrere Artefakte aus dem Lösungsbereich zugeordnet werden können, sodass es nicht nur eins-zu-eins Zuordnungen gibt. Dadurch existiert neben dem Konfigurationswissen eine Zuordnungsebene. Weiterhin werden laut Quinton et al. [Qui+21] Anpassungsregeln benötigt, welche angeben können, dass Feature **A** aktiviert wird, wenn Bedingung **N** erfüllt wurde.

Während der Laufzeit können somit laut Quinton et al. [Qui+21] Features, Zuordnungen und Artefakte im Problembereich, Zuordnungsbereich und Lösungsbereich hinzugefügt, entfernt oder aktualisiert werden. Dabei können die Veränderungen während der Laufzeit unterschiedliche Auswirkungen haben. Tabelle 3.1 zeigt dabei eine Zusammenfassung, welche Auswirkung eine Veränderung haben kann.

Tabelle 3.1 – Potenzielle Auswirkung von Veränderung während der Laufzeit nach Quinton et al. [Qui+21]

	Veränderung	**potenzielle Auswirkung**
Problembereich	hinzufügen	Featureauswahl hat keinen Effekt
	entfernen	Realisierungsartefakt kann nicht mehr aktiviert werden
	updaten	Featureauswahl hat keinen Effekt
Zuordnungsbereich	hinzufügen	Realisierungsartefakt kann nicht mehr aktiviert werden
	entfernen	Realisierungsartefakt kann nicht mehr aktiviert werden
	updaten	Featureauswahl hat keinen Effekt
Lösungsbereich	hinzufügen	Realisierungsartefakt kann nicht mehr aktiviert werden
	entfernen	Laufzeitadaption kann fehlschlagen
	updaten	unerwartetes Laufzeitverhalten

3.4 Zusammenfassung

In diesem Kapitel wurden Grundlagen genannt, welche in dieser Arbeit verwendet werden. Dabei wurde erläutet, was SAR sind und wo sie ihren Einsatz im Gesundheitswesen finden. Weiterhin wurden zwei SAR kurz vorgestellt.

Damit die Auswahl einer Interaktion anhand der Einschränkung eines Nutzer durchgeführt werden kann, wurden Grundlagen zu DL erklärt. DL ist hierbei unterteilt in überwachtes Lernen, unüberwachtes Lernen und Verstärkungslernen. Weiterhin wurden DL-Techniken aufgezählt und kurz beschrieben.

Anschließend wurde die Softwarevariabilität erläutert. Hier ist kurz erklärt worden, wo Variabilität in Software auftreten kann und wie Softwarevariabilität realisiert wird. Unter anderem wird ein Variabilitätsmodell namens Feature-Modell gezeigt und kurz beschrieben.

4 Analyse

In diesem Kapitel soll analysiert werden, welche Kriterien für die ältere Bevölkerung in der sprachlichen Interaktion wichtig sind und wie diese angepasst werden müssen, da jede Person unterschiedliche Bedürfnisse hat. Hierfür muss zunächst ermittelt werden, welche Einschränkungen die ältere Bevölkerung hat. Neben der Analyse von Kriterien und der Ermittlung der Einschränkungen muss auch ein geeignetes DL-Modell gewählt werden, sodass Anforderungen für ein Konzept extrahiert werden können.

4.1 Interaktionen mit älteren Menschen

In diesem Abschnitt soll gezeigt werden, welche Einschränkungen die ältere Bevölkerung mit der sprachlichen Interaktion hat. Dabei soll ein Feature-Modell erstellt werden, welches zeigt, wie valide Konfigurationen für die sprachliche Kommunikation aussehen können.

4.1.1 Einschränkungen der älteren Bevölkerung

Im höheren Alter kann festgestellt werden, dass es zu immer mehr Einschränkungen in der Kommunikation kommen kann. Dabei treten die Einschränkungen bei jeder Person anders auf. Somit kann nicht allgemein gesagt werden, dass im Alter jeder die selben Einschränkungen mit der gleichen Intensität bekommt. Es kann aber gezeigt werden, welche Einschränkungen am häufigsten vorkommen. Dies wird in diesem Abschnitt in den Bereichen Hören, Kognition und Sprechen verdeutlicht.

Hören Laut Baur et al. [Bau+09] ist die Altersschwerhörigkeit (ASH) eine der häufigsten sensorischen Beeinträchtigung bei älteren Menschen. Dabei weisen 37 % der 61 - 70 Jährigen eine ASH auf; bei 71 - 80 Jährigen sind es 80 %. Die ASH kann durch eine Vielzahl von Faktoren hervorgerufen werden, welche unter anderem genetische Vererbung oder einzelne Umweltfaktoren wie Lärm sein können. Es ist bekannt, dass laut Baur et al. [Bau+09; Wil+08] die Einschränkung in den meisten Fällen auf beiden Ohren etwa gleich stark auftritt. Laut Wisotzki [Wis96] könne neben der ASH auch Geräuschempflindlichkeit sowie Tinnitus auftreten. Geräuschempfindlichkeit besteht darin, dass wenn eine bestimmte Hörschwelle erreicht wird, die Lautstärkeempfindung schneller ansteigt, sodass ein Geräusch unangenehm zu hören ist. Tinnitus hingegen ist die Wahrnehmung von Geräuschen die keine reale Ursache haben. Dies ist laut Lassis et al. [LAG10] bei der älteren Bevölkerung weit verbreitet. Auch diese Einschränkung nimmt im Alter zu. Lassis et al. [LAG10] zeigt, dass 7 % der 65 - 69 Jährigen an Tinnitus leiden und bei den über 80-Jährigen 41 %.

Kognition Fisk et al. [Fis+19] beschreiben die Kognition bei älteren Menschen in unterschiedliche Bereiche. Dazu zählen Kurzzeitgedächtnis, Langzeitgedächtnis sowie bildliches Vorstellungs-

vermögen und Sprachverständnis. Das Langzeitgedächtnis kann weiter in das semantische Gedächtnis, das prospektive Gedächtnis und das prozeduale Gedächtnis aufgeteilt werden.

Kurzzeitgedächtnis Laut Fisk et al. [Fis+19; WFG11] bezieht sich das Kurzzeitgedächtnis auf die Fähigkeit, neue Eindrücke oder Informationen für einen kurzen Zeitraum zu speichern. Dieses Gedächtnis kann das Sprachverständnis, sowie das Entscheidungsvermögen beeinflussen, sodass es evtl. zu häufigen Wiederholungen von beantworteten Fragen kommen kann. Nach Wild-Wall et al. [WFG11] gibt es mehrere Studien, die zeigen, dass das Kurzzeitgedächtnis im Alter nachlässt.

Sprachverständnis Fisk et al. [Fis+19] erklären, dass durch ein nachlassendes Kurzzeitgedächtnis ältere Menschen mehr Zeit benötigen, um komplexe Sätze zu verstehen. Hierbei spielen mehrere Faktoren eine Rolle, wie zum Beispiel die Geschwindigkeit. Unter anderem können leise Hintergrundgeräusche bei schnell gesprochenen Sätzen laut Mundorff [Mun11] das Sprachverständnis beeinträchtigen.

Sprechen Durch physiologische Veränderungen und einem anderen Sprachstil von älteren Menschen, kann die Sprache ebenfalls beeinträchtigt werden. Laut Das et al. [Das+13] sind unter physiologischen Veränderungen die Veränderungen von Kehlkopf und Stimmbändern zu verstehen. Dabei zeigt eine Studie von Tremblay et al. [Tre+18], dass ältere Menschen oft Fehler beid der Aussprache von zufällig generierten Silben machen.

Tabelle 4.1 – Übersicht der Kommunikationsstrategien nach Haustein [Hau20]

Dialogkriterium	**Kommunikationsstrategie**
Angemessenheit für Benutzeraufgaben	Erkennung von veralteten Wörtern und unüblichen Grammatiken
	Umgang mit Dialekten und Sprachstörungen
	Toleranz von Sprachpausen, Wiederholung von Satzteilen und Füllwörtern
	Vereinfachte Verwendung der Aktivierungsphrase
Selbstbeschreibungsfähigkeit	Sorgfältige Portionierung und Strukturierung der dargestellten Informationen
	Eindeutige Kennzeichnung der Eingabephase
	Eindeutige Kennzeichnung der Position im System
Erwartungskonformität	Erfassung von impliziten Informationen aus dem Kontext der Eingabe
	Variabilität der Ausgabeformulierung
	Dynamische Anpassung von Ausgabeparametern
Erlernbarkeit	Erforschung der Funktionen ohne irreversible Aktionen auszulösen
	Präsentation von relevanten oder ähnlichen Funktionen
Steuerbarkeit	Einstellbarkeit von Ausgabeparametern und Persönlichkeit
	Kontrolle über die aktiven und im Hintergrund ablaufenden Aktionen
Fehlerrobustheit	Verifizierung von Eingaben
	Aussagekräftige Fehlermeldungen
Benutzerbindung	Professionelle Atmosphäre
	Positives Feedback

Der Sprachstil wird von Fiehler et al. [Fie97] für ältere Menschen in mehreren Beispielen definiert. Hierbei hat die ältere Bevölkerung durchaus eine höhere Rate an Wortfindungsstörung, sowie eine höhere Rate bei der Verwendung von älteren Begriffen.

Haustein [Hau20] befasst sich mit Erfolgsmethoden für Sprachbenutzungsoberflächen von Assistenzrobotern für Senioren. Dabei stellt Sie Strategien auf, welche für die Sprachinteraktion mit Senioren genutzt werden können. Diese Strategien inklusive der Dialogkriterien werden in Tabelle 4.1 dargestellt.

4.1.2 Erstellung Feature-Modell

Mit den von Haustein [Hau20] erstellten Strategien aus Tabelle 4.1 können Features für das Feature-Modell erstellt werden. Hierbei ist zu beachten, dass nicht jede Strategie in das Feature-Modell eingearbeitet werden kann, da es keine Auswirkung auf die Anpassung der Interaktion hat.

Das Feature-Modell unterteilt sich zunächst in zwei Features: **Eingabe** und **Ausgabe**. Durch die Strategie *Toleranz von Sprachpausen, Wiederholung von Satzteilen und Füllwörtern* lässt sich ein Feature **Wartezeit bis Abschluss** ableiten, welches der **Eingabe** zugeordnet werden kann. Mit diesem Feature soll das System die Wartezeit anpassen, also wie lange das System warten muss, bis mit der Verarbeitung fortgeführt werden kann, sodass der Kommunikationsfluss dynamisch gehalten wird. Somit ist die Wartezeit nicht statisch sehr lange gehalten und verhindert die Verlangsamung des Kommunikationflusses. Hier wird dieses Feature in **kurz**, **mittel**, **lang** unterteilt. Welche Zeit für die Begriffe benötigt wird, ist hierbei nicht definiert.

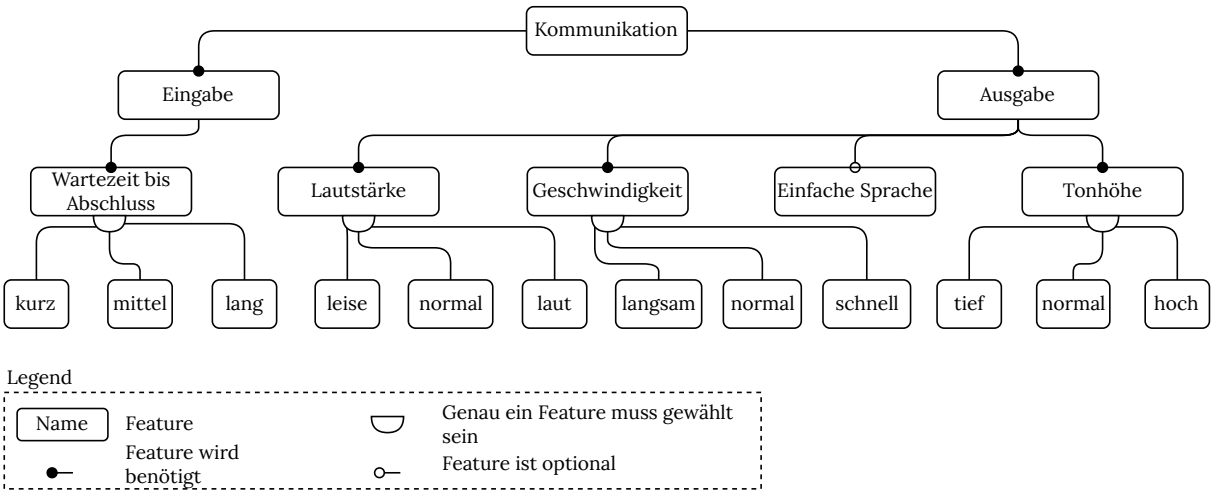


Abbildung 4.1 – Feature Modell

Für das Feature der **Ausgabe** benötigt man die Strategie *Dynamische Anpassung von Ausgabeparametern*. Haustein [Hau20] schreibt in ihrer Arbeit folgendes: „Lautstärke, Sprechgeschwindigkeit und Tonhöhe sollten je nach Kontext angepasst werden.[...] Auch das Verhalten des Benutzers kann Einfluss auf die Parameter nehmen.[...] Die Informationen zur Anpassung der Parameter können auch direkt vom Benutzer erfragt werden.“ Somit lassen sich die Features **Lautstärke**, **Geschwindigkeit** und **Tonhöhe** ableiten. Das Feature **Lautstärke** wird unterteilt in **leise**, **normal** und

laut, wobei der Lautstärkepegel für diese Begriffe nicht definiert ist. Analog wird die **Geschwindigkeit** in **langsam**, **normal** und **schnell** unterteilt, wobei hier ebenfalls die Begriffe nicht an einen festen Wert gebunden sind. Das Feature **Tonhöhe** setzt sich aus **tief**, **normal** und **hoch** zusammen. Auch für diese Begriffe sind die Werte der Tonhöhe nicht definiert. Neben diesen Features kann auch durch die Strategie *Variabilität der Ausgabeformulierung* auf das Feature **einfache Sprache** geschlossen werden. Die *einfache Sprache* sollte dabei Verwendung finden, wenn der Nutzer kognitiv nicht mehr komplexe Satzstrukturen erkennen kann.

4.2 Analyse eines Deep-Learning-Modells

Damit eine valide Konfiguration aus dem Feature-Modell bewertet werden kann, wird zunächst eine Analyse benötigt, welches Modell für DL passend für den Kontext ist. Es werden mehrere DL-Modelle verglichen und durch ein Auschlussverfahren soll ein Modell gewählt werden.

4.2.1 Wahl eines Deep-Learning-Modells

In diesem Abschnitt steht die Wahl eines DL-Modells an, mit dem ein Konzept erstellt werden kann, sodass der Roboter anhand der Einschränkungen des Nutzers eine passende Interaktion wählt.

Um die Auswahl der DL-Modelle einzuschränken wird vorerst beschrieben, welche Daten bekannt sind. Unter anderem sind die Einschränkungen der Nutzer bekannt, welche in Abschnitt 4.1.1 kurz erläutert wurden. In Abschnitt 5.2 werden diese weiter präzisiert, sodass diese für ein Modell zur Nutzung bereitstehen. Weiterhin sind alle möglichen Konfigurationen einer Interaktion bekannt. In Abschnitt 4.1.2 wurde eine Feature-Modell erstellt, welches alle möglichen Konfigurationen einer Interaktion abbildet. Außerdem ist bekannt, dass für einen Nutzer eine Interaktion gewählt werden muss. Daher kann davon ausgegangen werden, dass wir die Ausgabedaten des Modells kennen. Somit sind Daten mit Eingabe- und Ausgabedaten bekannt, sodass das Modell trainiert werden kann.

Dies impliziert, dass es sich nicht um *unüberwachtes Lernen* oder *Verstärkungslernen* handeln kann, sondern um *überwachtes Lernen*. Demzufolge werden in diesem Abschnitt Modelle des *überwachten Lernen* vorgestellt und verglichen.

Vorstellung der Deep-Learning-Modelle

Zum *überwachten Lernen* gehören die Modelle MLP, CNN, RNN und Neural Collaborativ Filtering (NCF). In diesem Abschnitt werden die Modelle kurz erläutert und gezeigt, wo solche Modelle ihren Einsatz finden.

Multi-Layer Perceptron (MLP) Dieses Modell wurde in Abschnitt 3.2.2 beschrieben und in Abbildung 3.5 gezeigt. Dies sind einfache KNNs, welche aus einer Eingabeschicht, Ausgabeschicht und mindestens einer versteckten Schicht bestehen. Dabei sind die Neuronen von Schicht zu Schicht voll verzweigt. Außerdem sind dies Feed-Forward-Netzwerke, welche demzufolge die Daten von einer unteren Schicht immer zu einer höheren Schicht senden. MLPs werden überwiegend für Vorhersagen, Klassifikation oder Regression verwendet. Außerdem macht es sie sich gut, wenn mit tabellarischen Daten gearbeitet wird. [TS15; Ram+16]

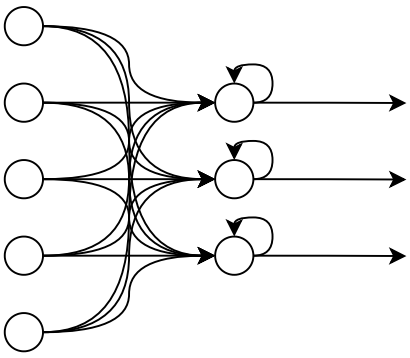


Abbildung 4.2 – Schematische Darstellung eines RNN nach Deru und Ndiaye [DN20]

Recurrent Neural Network (RNN) Auch dieses Modell wurde in Abschnitt 3.2.2 kurz beschrieben. Dieses Modell ist kein Feed-Forward-Netzwerk, da es auf Neuronen zugreift, welche in der selben Schicht oder in der Schicht davor liegen. Dies ist gut in Abbildung 4.2 zu sehen. Laut Deru und Ndiaye [DN20] lösen solche Modelle Aufgaben, welche zeitliche Komponenten, Gedächtnis oder einen Kontext voraussetzen. Solche Modelle werden nach Deru und Ndiaye [DN20] oft für „Spracherkennung, Übersetzungen, Satzvervollständigungen, Wettervorhersagen, Vorhersagen von Aktienkurven [sowie] Zinsprognosen eingesetzt“.

Convolutional Neural Network (CNN) Dieses Modell wurde ebenfalls in Abschnitt 3.2.2 beschrieben. Laut Deru und Ndiaye [DN20] ist die Idee hinter CNN ein zu lösendes globales System in kleineren und leichter zu lösenden Schritten zu abstrahieren. Ein CNN besteht dabei aus einer Eingabeschicht, einer Faltungsschicht, einer Aktivierungsschicht, einem Pooling Layer, aus einem oder mehreren Dense Layern und einer Ausgabeschicht. Die Schichten zwischen Faltungsschicht bis zum Dense Layer können dabei mehrmals hintereinander ausgeführt werden. Die Faltungsschicht wendet einen Filter auf die Eingabedaten an und generiert eine sogenannte Feature Map. Die Aktivierungsschicht wendet eine Aktivierungsfunktion auf die Feature Map an. Der Pooling Layer reduziert danach die Dimension der Feature Map und verstärkt somit markante Stellen in der Eingabe. Diese Modelle sind laut Deru und Ndiaye [DN20] „von der Beobachtung der Informa-

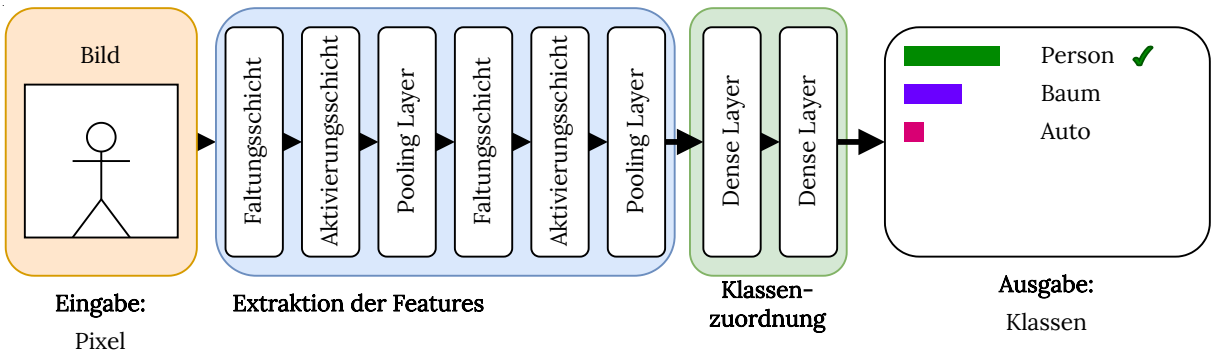


Abbildung 4.3 – Prinzip eines CNN nach Deru und Ndiaye [DN20]

tionsverarbeitung im visuellen Kortext inspiriert“ . Daher eignen sich diese Modelle sehr gut für die Klassifikation von Bild- und Videoerkennung. In Abbildung 4.3 ist das Prinzip sehr gut erklärt. Als Eingabe werden die Pixel eines Bildes in das Modell gegeben. Die Faltungsschicht, Aktivierungsschicht und der Pooling Layer extrahieren die Features. Diese können mehrmals hintereinander ausgeführt werden. Durch Dense Layer werden die Features dann den Klassen zugeordnet. Die Ausgabe enthält die Information, um welche Klasse es sich handelt.

Neural Collaborativ Filtering (NCF) Bei diesem Modell wird nach He et. al [He+17] eine Bewertung von einem Benutzer zu einem bestimmten Objekt vorhergesagt. In Abbildung 4.4 wird solch ein Modell gezeigt. Als Eingabe bekommt hierbei das Modell jeweils einen Benutzer und ein Objekt. Aus dem Benutzer wird ein sogenannter Latenter Vektor erstellt. Das gleiche wird für das Objekt getan. Diese Latenten Vektoren werden an die neuronalen CF Schichten weitergegeben. Als Ausgabe wird eine Bewertung vorhergesagt. Die Idee hinter NCF ist laut He et. al [He+17] die CF mithilfe von DL zu verbessern. Solche Systeme werden für Empfehlungssysteme genutzt, wie zum Beispiel in Videostreamingdiensten oder Shoppingseiten.

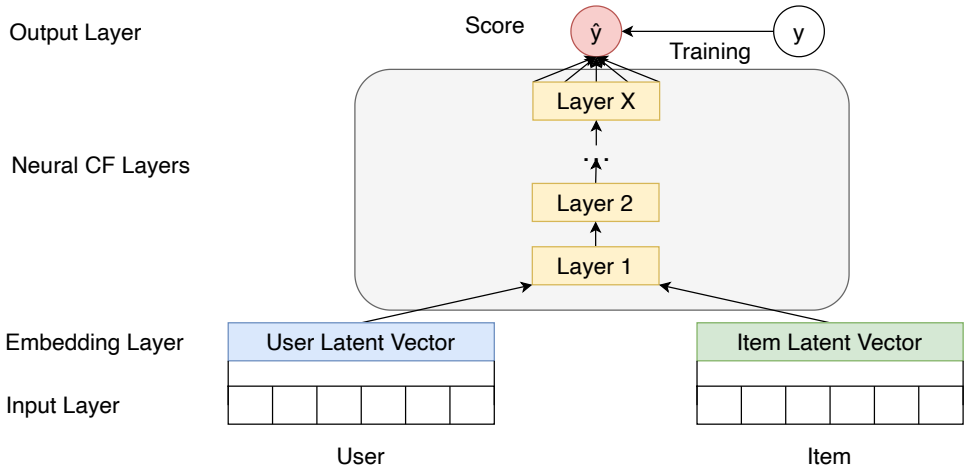


Abbildung 4.4 – Architektur eines NCF

4.2.2 Vergleich der Deep-Learning-Modelle

In Tabelle 4.2 sind nun die vorgestellten Modelle tabellarisch aufgestellt. Nun werden die Modelle, welche nicht den Erwartungen entsprechen, ausgeschlossen. Das Modell, welches übrig bleibt, kann für das Konzept verwendet werden.

Tabelle 4.2 – Vergleich der Deep-Learning-Modelle

	MLP	CNN	RNN	NCF
Lerntyp	überwacht	überwacht	überwacht	überwacht
Netzwerkverkettung	feed forward	feed forward	recurrent	feed forward
Anwendungsfälle	Regression Klassifikation	Klassifikation von visuellen Daten	Spracherkennung Übersetzung Wettervorhersage etc	Empfehlungssysteme

MLP Dieses Modell kann für die Vorhersage von Interaktionen genutzt werden. Hierbei könnten die Eingabedaten für das Modell den Einschränkungen einer Person entsprechen. D. h. die Eingabedaten würden, ähnlich wie in Abschnitt 5.2, den *Eingabedaten des Nutzers* entsprechen. Dies wären die **Stadien der ASH**, der **kognitive Erkrankung**, des **Alter** und **Geschlechts** der Person. Als Ausgabedaten des Modells könnten die einzelnen Features aus dem Feature-Modell, welches in Abschnitt 4.1.2 erstellt wurden, genutzt werden. Demzufolge haben wir als Ausgabedaten die**Wartezeit**, die **Lautstärke**, die **Geschwindigkeit**, die **Tonhöhe** und die **einfache Sprache**.

In Abbildung 4.5 wird das Modell beispielhaft dargestellt. Hierbei gibt es eine Eingabeschicht, in denen die Eingabedaten mitgegeben werden, mehrere versteckte Schichten und eine Ausgabeschicht, welche vorhersagen soll, welche Features wie aktiviert sein müssen. Dabei wird das Modell so trainiert, dass es für eine Kombination aus bestimmten Einschränkungen genau eine bestimmte Kombination der Feature existiert. Demzufogle entspricht die Vorhersage einer Regression.

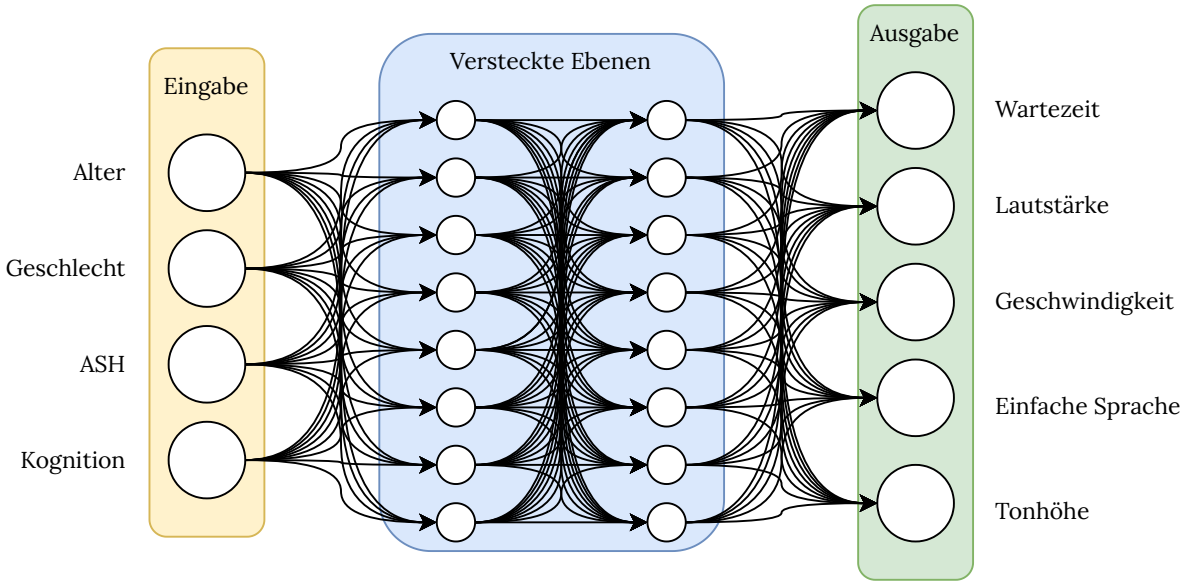


Abbildung 4.5 – MLP mit Ein- und Ausgabedaten

CNN und RNN Wie in Tabelle 4.2 abzulesen, wird CNN überwiegend in den Anwendungsfällen für Klassifikationen von visuellen Daten verwendet. Da hier keine visuellen Daten verarbeitet werden, wird dieses Modell ausgeschlossen.

RNN wird laut Tabelle 4.2 für Spracherkennung genutzt, daher könnte dieses Modell in Betracht gezogen werden. Allerdings beschäftigt sich diese Arbeit allgemein mit der verbalen Interaktion, wobei die Interaktion für eine Person vorhergesagt werden soll. Ein RNN ist demzufolge an einer anderen Stelle sinnvoll, wo die gesprochene Sprache erkannt wird, sodass ein Roboter bzw. ein Computer mit diesen Informationen weiterarbeiten kann. Demzufolge wird auch dieses Modell ausgeschlossen.

NCF Ziel dieser Arbeit ist, dass das System eine passende Interaktion für den Benutzer wählt. Während der Nutzer das System verwendet, soll das System die Interaktion anpassen, wenn die-

se dem Benutzer nicht gut genug ist. Das System soll durch implizites oder explizites Feedback entscheiden, ob die gewählte Interaktion passend ist.

Die Nutzer haben bestimmte Eigenschaften, welche später in 5.2 definiert werden. Die Objekte sind alle möglichen Konfigurationen aus dem erstellten Feature-Modell in Abschnitt 4.1.1. D. h. für dieses Modell müssen alle möglichen Konfigurationen der Interaktionen bekannt sein. Die Benutzer und Objekte sind nun unsere Eingabedaten für dieses Modell. Als Ausgabe entsteht eine Wertung für Nutzer **u** mit Objekt **i**, mit der das beste Objekt ausgewählt werden kann. Vorteil beim NCF ist, dass nicht wie beim MLP jeder Benutzer einzeln betrachtet wird, sondern dass ein Benutzer Ähnlichkeiten mit anderen Benutzern haben kann und somit auch gleiche Objekte verwendet werden können.

Sollte der Nutzer mit der nun gewählten Interaktion nicht zufrieden sein, kann das System mit Hilfe von implizitem und explizitem Feedback die Wahl einer Interaktion verändern. Wenn der Nutzer mit dem Dialog mit dem Roboter einfach fortführt, kann dies als implizites Feedback verstanden werden, dass der Nutzer mit der Interaktion zufrieden ist. Andererseits kann der Nutzer auch direkt sagen, ob er den Roboter verstanden hat, was einem explizitem Feedback entspricht. Der Roboter sollte hierbei auch nach dem Dialog den Nutzer nach einem explizitem Feedback fragen, sodass das DL-Modell weiter lernen kann.

Entscheidung für ein Modell

Durch das Ausschlussverfahren kristallisieren sich zwei mögliche Modelle als geeignet heraus. Zum einen das MLP und das NCF. Das MLP wird für einen Benutzer eine eindeutige valide Konfiguration einer Interaktion vorschlagen. Sollte diese Interaktion dem Nutzer nicht gefallen oder es Schwierigkeiten mit dieser Interaktion geben, ist es hier problematisch, einen weiteren Kandidaten für eine Interaktion für diesen Benutzer zu finden. Beim NCF wird für jede valide Konfiguration einer Interaktion eine Bewertung für einen Benutzer vorhergesagt. Sollte der Benutzer mit der von NCF vorgeschlagenen validen Konfiguration einer Interaktion nicht zufrieden sein, kann diese Konfiguration niedriger bewertet werden und es wird die nächste Konfiguration gewählt.

Aus diesem Grund bietet sich das NCF Modell mehr an und wird im Folgendem weiter ausgebaut.

4.3 Erstellung der Anforderung für ein Konzept

Um im nächsten Kapitel ein Konzept zu entwickeln, ist es notwendig, Anforderungen aufzustellen, welche später an dem Konzept getestet werden können, um zu kontrollieren, dass das erstellte Konzept den Anforderungen entspricht.

In Abschnitt 4.1.2 wurde ein Feature-Modell erstellt, welches valide Konfigurationen der Interaktion abbildet. Daraus folgt eine Anforderung, dass das Konzept valide Konfigurationen von Interaktionen für eine Person voraussagt. Diese Anforderung wird **Gültigkeit** genannt.

In Abschnitt 4.2.2 wird erklärt, dass ein Modell gewählt wird, welches zum Lerntyp des überwachten Lernens gehört. Das heißt, dass mit vorgegebenen Eingabe- und Ausgabewerten das Modell trainiert wird. Demzufolge kann folgende Anforderung erstellt werden: Das Konzept soll für eine manuell bewertete Interaktion eine ähnliche Bewertung für die Interaktion vorhersagen. Diese Anforderung wird **Genauigkeit** genannt.

Eine weitere Anforderung kann ebenfalls aus Abschnitt 4.2.2 entnommen werden. In diesem Abschnitt wurde ein NCF-Modell gewählt. Demzufolge sollen Personen, die Ähnlichkeiten aufweisen, zu den selben Interaktionen ähnliche Bewertungen für die Interaktion aufweisen. Ähnlichkeiten zwischen den Benutzern treten auf, wenn die Eigenschaften der Benutzer gleich sind. Diese Anforderung wird **Ähnlichkeit** genannt.

Abschnitt 4.1.1 zeigt, dass Personen an unterschiedlichen Einschränkungen leiden können. Demzufolge wird eine Anforderung erstellt, dass für eine Person mit bestimmten Einschränkungen Interaktionen mit passenden Features eine höhere Bewertung vorausgesagt wird, als für Interaktionen mit nicht passenden Features. Diese Anforderung wird **Anordnung** gennant.

Es wurden demnach folgende Anforderungen an das Konzept erstellt:

1. **Gültigkeit**
   Es sollen Bewertungen für valide Konfigurationen der Interaktionen für Personen erstellt werden.

2. **Genauigkeit**
   Die vorhergesagte Bewertung von einer Person zu einer Interaktion soll einer manuellen Bewertung der Person für diese Interaktion gleichen.

3. **Ähnlichkeit**
   Personen, die ähnliche Eigenschaften haben, sollen für eine Interaktion ähnliche Bewertungen vorhergesagt bekommen.

4. **Anordnung**
   Für eine Person mit bestimmten Eigenschaften sollen Interaktionen mit passenden Features höhere Bewertungen vorhergesagt werden.
   Welche Eigenschaft zu welchem passenden Feature gehört wird in Abschnitt 5.4 beschrieben.

4.4 Zusammenfassung

Dieses Kapitel zeigte, welche Einschränkungen die ältere Bevölkerung hat. Mithilfe der Strategien von Haustein [Hau20] konnte ein Feature-Modell erstellt werden, wie ein Roboter mit einer älteren Personen zu kommunizieren hat. Außerdem wurden aus den Strategien Attribute extrahiert, welche ältere Menschen in ihrer Kommunikation aufweisen.

Weiterhin wurde ein DL-Modell ausgewählt, welches eine Interaktion anhand von Einschränkungen einer Person vorhersagt. Dabei bezogen sich die DL-Modelle auf Methoden des überwachten Lernens. D. h. um solch ein Modell zu trainieren sind Eingabe- sowie auch Ausgabedaten nötig. Dabei wurden vier Modelle genauer vorgestellt: MLP, RNN, CNN, NCF.

Nach der Vorstellung dieser DL-Modelle wurden nicht passende Modelle ausgeschlossen. Dabei wurde erklärt, dass die ausgeschlossenen Modelle für andere Anwendungsfälle bestimmt sind. Übrig blieben MLP und NCF. Zum Schluss wurde das NCF ausgewählt, da hier mehrere Interaktionmöglichkeiten angeboten werden können und ein Nutzer, wenn ihm die vorgeschlagene Interaktion nicht gefällt, auf eine Zweite ausweichen kann.

Weiterhin wurden Anfordungen an das Konzept erstellt, welche ebenfalls zur Testung des Konzepts genutzt werden sollen.

Das Modell NCF wird nun in Kapitel 5 weiter erläutert und erweitert, sodass gezeigt werden kann, wie dieses Modell für eine Person anhand der Einschränkungen eine Interaktion wählt.

5 Konzeption

In diesem Kapitel wird ein Konzept erstellt, in dem die Eingabe- und Ausgabedaten des Modells aus Abschnitt 4.2 erläutert werden. Hierbei wird erklärt, woher die Eingabedaten kommen und wie die Ausgabedaten ausgewertet werden. Weiterhin wird gezeigt, wie das Ausgewählte Modell aus Kapitel 4 eingesetzt und verwendet wird.

5.1 Einordnung des Prozesses

Damit genau bekannt ist, wo das ausgewählte Modell seinen Einsatz findet, wird in diesem Abschnitt kurz erläutert, wie der Ablauf mit der Interaktion vonstatten geht.

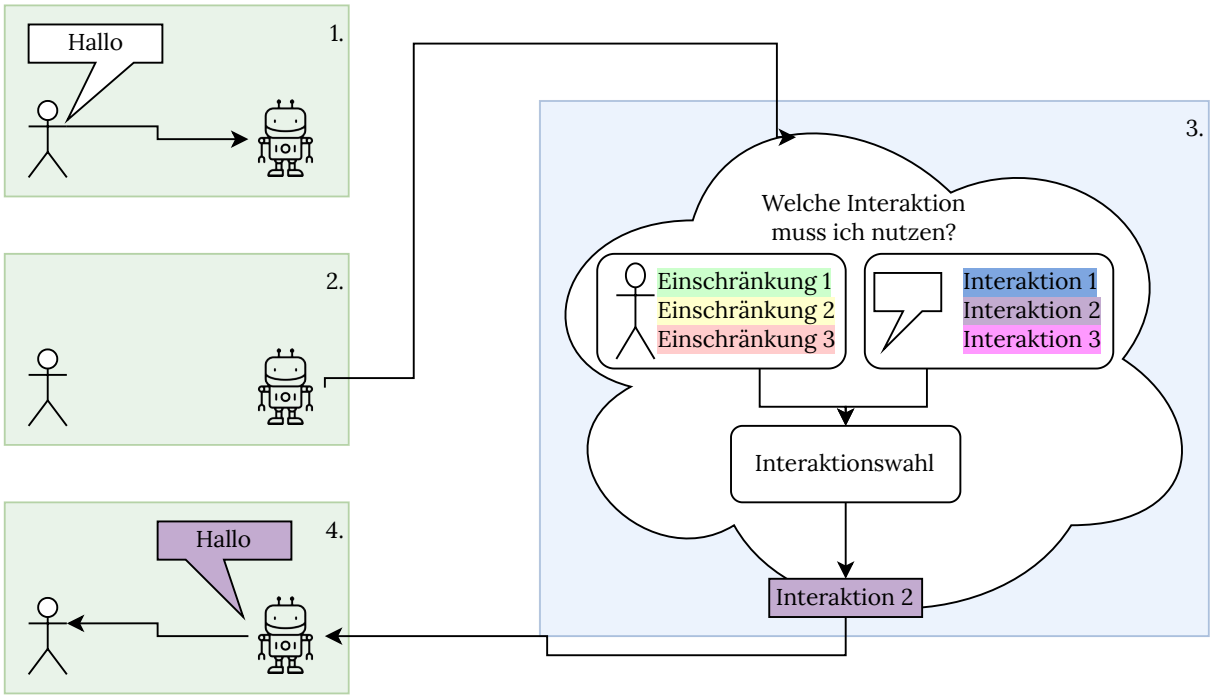


Abbildung 5.1 – Ablauf der Auswahl einer Interaktion

Dieser Ablauf ist sehr gut in Abbildung 5.1 gezeigt. Zunächst ist eine Person mit Einschränkungen und ein SAR vorhanden. Diese Person startet eine Interaktion mit dem SAR. Danach sollte der SAR mit der Person interagieren. Hier muss der SAR zunächst schauen, welche Interaktion mit der Person durchgeführt werden sollte. Dafür kennt der SAR alle Einschränkungen dieser Person und alle möglichen Interaktionen, die der SAR durchführen kann. Anhand dieser Informationen kann der SAR nun die richtige Interaktion wählen. Die gewählte Interaktion wird hierbei dem SAR übermittelt, sodass dieser die gewählte Interaktion zum Schluss ausführen kann.

Das gewählte Modell aus Kapitel 4 wird nun bei der Interaktionswahl eingesetzt. Hierfür muss klar definiert werden, wie solche Einschränkungen und die Interaktionen aussehen können.

Eingrenzung der Arbeit Diese Arbeit befasst sich mit der Wahl einer Interaktion für eine Person anhand ihrer Einschränkungen und Eigenschaften. Dabei wird nicht festgestellt, wie diese Einschränkungen und Eigenschaften erfasst werden und dem SAR mitgeteilt werden. D.h. der SAR kennt die Einschränkungen und kann mit diesen arbeiten.

Bei den Interaktionen verhält es sich hier ähnlich. Die Interaktionen befassen sich in dieser Arbeit nur mit verbaler Kommunikation. D.h. Interaktionen mit Bilderkennung oder Touchscreen sind vorerst ausgenommen.

5.2 Definition der Eingabedaten

Das gewählte Modell aus Kapitel 4 benötigt zwei Eingabedaten: zum einen die Nutzer mit bestimmten Eigenschaften und zum anderen Interaktionen, welche vom Roboter ausgeführt werden sollen.

Eingabedaten des Nutzers Aus Abschnitt 4.1.1 und der Tabelle 4.1 können ähnlich wie in Abschnitt 4.1.2 relevante Eigenschaften von Nutzern abgeleitet werden. Eine Eigenschaft ist die ASH, welche unterschiedliche Stadien haben kann. Hierfür wird definiert, dass es fünf Stadien der ASH geben kann:

- Normalhörigkeit:
  Die Hörschwelle liegt zwischen 1 bis 20 dB.
- Geringgradige Schwerhörigkeit:
  Die Hörschwelle liegt zwischen 25 bis 40 dB
- Mittelgradige Schwerhörigkeit:
  Die Hörschwelle liegt zwischen 40 bis 60 dB
- Hochgradige Schwerhörigkeit:
  Die Hörschwelle liegt zwischen 60 und 80 dB
- Gehörlosigkeit:
  Die Hörschwelle liegt über 80 dB.

Seit 2021 wurden durch die World Health Organisation (WHO) [WHO21] mehr Stadien der Schwerhörigkeit definiert. In dieser Arbeit werden jedoch die Stadien der Schwerhörigkeit der WHO von 2001 genutzt, wie bei Heger und Holube [HH10] beschrieben, da zur Generierung der Daten Statistiken und Studien genutzt werden, welche ebenfalls auf die Definitionen der Stadien der Schwerhörigkeit von der WHO von 2001 aufbauen.

Um die unterschiedlichen Stadien der ASH später nutzen zu können, werden diese mit Zahlen definiert. Dabei werden die Zahlen wie in Tabelle 5.1 zugewiesen.

Eine weitere Eigenschaft ist das Alter der Person. Wie in Abschnitt 4.1.1 beschrieben, leiden 71 - 80 Jährige mehr an ASH als 61 - 70 Jährige. Demzufolge sollte das Alter mit als Eigenschaft genutzt werden.

Tabelle 5.1 – Zuweisung der ASH

Stadium der ASH	Zuweisung der Zahl
Normalhörigkeit	0
Geringgradige Schwerhörigkeit	1
Mittelgradige Schwerhörigkeit	2
Hochgradige Schwerhörigkeit	3
Gehörlosigkeit	4

Eine weitere Eigenschaft wird als kognitive Erkrankung definiert. Hierbei beruht die kognitive Erkrankung auf einer Demenzerkrankung. Obwohl es unterschiedliche Stadien einer Demenzerkrankung gibt, werden diese alle als kognitive Erkrankung zusammengefasst. D. h., wenn eine Person an einer Demenzerkrankung leidet, wird festgelegt, dass diese Person eine kognitive Beeinträchtigung hat. Hier wird wieder eine Zuweisung definiert, sodass das DL-Modell besser damit umgehen kann. Hierbei wird definiert, dass wenn keine Demenzerkrankung vorliegt, der Wert der kognitiven Erkrankung auf 0 gesetzt wird. Wenn eine vorliegt, wird diese auf 1 gesetzt.

Die letzte Eigenschaft soll das Geschlecht des Nutzers sein, da laut Wiley et al. [Wil+08] der Hörverlust zwischen Mann und Frau unterschiedlich sein kann.

Somit haben wir folgende Eingabedaten für den Nutzer: **Alter**, **Geschlecht**, **Stadium der ASH** und **kognitive Erkrankung**.

Eingabedaten der Interaktion Die Eigenschaften der Interaktion lassen sich aus dem Feature-Modell aus Abschnitt 4.1.2 ableiten. Die Eigenschaften sind demzufolge die **Wartezeit**, die **Lautstärke**, die **Geschwindigkeit**, die **Tonhöhe** und ob die **einfache Sprache** verwendet werden soll.

Tabelle 5.2 – Zuweisung der Eingabedaten der Interaktion

(a) Einfachen Sprache

Verwendung der einfachen Sprache	Zuweisung der Zahl
wird nicht verwendet	0
wird verwendet	1

(b) Wartezeit

Länge der Wartezeit	Zuweisung der Zahl
kurz	0
mittel	1
lang	2

(c) Lautstärke

Lautstärke	Zuweisung der Zahl
leise	0
normal	1
laut	2

(d) Geschwindigkeit

Geschwindigkeit	Zuweisung der Zahl
langsam	0
normal	1
schnell	2

(e) Tonhöhe

Tonhöhe	Zuweisung der Zahl
tief	0
normal	1
hoch	2

Die **Wartezeit** kann hierbei *kurz*, *mittel* oder *lang* sein. Die **Lautstärke** kann *leise*, *normal* oder *laut* sein. Die **Geschwindigkeit** kann *langsam*, *normal* oder *schnell* sein und die **Tonhöhe** *tief*, *normal* oder *hoch*.

Diesen Daten werden ebenfalls Zahlen versehen, um später einfacher damit arbeiten zu können. Diese Zuweisungen können der Tabelle 5.2 entnommen werden.

Beispiele der Eingabedaten In Tabelle 5.3a können nun Beispiele für Eingabedaten der Interaktion entnommen werden. Hier gibt es die Interaktion 0 bis 3. Dabei unterstützen diese Interaktionen unterschiedliche Features. So hat zum Beispiel die Interaktion 1 eine lange Wartezeit, eine mittlere Lautstärke, eine langsame Geschwindigkeit und eine normale Tonhöhe.

Tabelle 5.3 – Beispiele der Eingaben

(a) Tabelle der möglichen Interaktionen

Interaktion	Wartezeit	Lautstärke	Geschwindigkeit	Tonhöhe	Einfache Sprache
0	0	2	1	1	0
1	2	1	0	1	0
2	1	1	2	0	1
3	0	0	0	2	1

(b) Benutzer mit ihren Eigenschaften

Benutzer	Alter	Geschlecht	ASH	kognitiven Erkrankung
0	64	f	3	0
1	84	m	2	0
2	70	m	3	0
3	85	f	1	1

In Tabelle 5.3b können Beispiele für die Eingabedaten der Benutzer entnommen werden. Benutzer 3 zum Beispiel ist 85 Jahre alt, gehört dem weiblichen Geschlecht an, leidet an einer geringgradigen Schwerhörigkeit und einer kognitiven Erkrankung.

5.3 Ausgabedaten

Als Ausgabe des Modells aus Kapitel 4 wird eine Vorhersage für die Bewertung einer Interaktion für einen Benutzer getroffen. D. h. jede Interaktion besitzt für jede Person eine bestimmte vorhergesagte Bewertung. Somit kann für die Person, die den Roboter nutzt, die Interaktion mit der höchsten Bewertung ausgewählt werden.

Der Wert der Bewertung kann hierbei einen Wert zwischen 0 und 5 annehmen, wobei 0 die schlechteste und 5 die beste Bewertung ist.

5.4 Erstellung der Trainingsdaten

Die Testdaten werden generisch auf Basis von Studien und Statistiken von Heger und Hubole [HH10], Deutsche Alzheimer Gesellschaft e. V. [Deu22] sowie vom statistischem Bundesamt

[Sta22] hergestellt. Diese Testdaten sind nicht evaluiert und dienen nur dem Zweck, das ausgewählte Modell aus Kapitel 4 zu trainieren, um auf dieser Basis eine Interaktion zu wählen.

Daten für die Interaktionen Um die Daten für Interaktionen zu erstellen, werden alle möglichen Konfigurationen aus dem Feature-Modell gebildet und tabellarisch dargestellt. Dabei werden *Wartezeit bis Abschluss*, *Lautstärke*, *Geschwindigkeit* und *Tonhöhe* mit den Werten 0, 1 und 2 dargestellt. In Tabelle 5.3a wird gezeigt, wie solche Daten aussehen können.

Um diese Interaktionen zu generieren wurde folgendes Skript in Quelltext 5.1 geschrieben. Hier werden alle möglichen Interaktionen tabellarisch erstellt. Dabei werden für die Geschwindigkeit, Lautstärke, Wartezeit, **Tonhöhe** und die Verwendung der einfachen Sprache und zugewiesenen Werte durchiteriert und für eine neue Interaktion gesetzt.

```
import pandas as pd
products = pd.DataFrame(
    columns=["productID", "delay", "volume", "speed", "pitch", "easy_language"]
)
productID = 0
for delay in range(0, 3):
    for volume in range(0, 3):
        for speed in range(0, 3):
            for pitch in range(0, 3):
                for easy in range(0, 2):
                    products = pd.concat(
                        [
                            products,
                            pd.DataFrame(
                                [[productID, delay, volume, speed, pitch, easy]],
                                columns=[
                                    "productID",
                                    "delay",
                                    "volume",
                                    "speed",
                                    "pitch",
                                    "easy_language",
                                ],
                            ),
                        ]
                    )
                    productID += 1
products.reindex()
products.to_csv("products.csv", index=False)
```

Quelltext 5.1 – Erstellung Testdaten für Interaktionen

Daten für die Benutzer Die Benutzer werden ebenfalls durch einen Algorithmus generiert. Hierbei wird zunächst definiert, dass nur Benutzer älter als 55 erstellt werden. Insgesamt werden 200 Nutzer generiert. United Nations [UDP22] sprechen von älteren Personen, wenn diese über 65 Jahre alt sind. In dieser Arbeit wird das Alter ab 55 als Übergang zur älteren Person bezeichnet. Daher werden nur Benutzer älter als 55 erstellt. Welches Alter eine Person hat wird statistisch durch das statistische Bundesamt [Sta22] erhoben.

In Quelltext 5.2 kann dies gut nachvollzogen werden. In Zeile 1 wird eine Variable angelegt, welche eine Zuweisung beherbergt. Hier wird für jedes Alter ab 55 die Anzahl der in Deutschland lebenden Menschen gespeichert. In Zeile 2 werden die Anzahl der in Deutschland lebenden Menschen pro Alter zusammengerechnet, sodass später prozentual entschieden werden kann, welches Alter eine Person hat. In Zeile 3 wird eine Variable angelegt, die beschreibt mit wie viel Prozent eine Person das entsprechende Alter haben kann. In Zeile 4 werden dann 200 Personen erstellt mit unterschiedlichem Alter und der Gewichtung, die in Zeile 3 erstellt wurde.

```
1 ages = {<Alter>: <Anzahl in Deutschland>}
2 all_members = sum(list(ages.values()))
3 ages_percent = {key: value / all_members for key, value in ages.items()}
4 random_ages = random.choices(
5     list(ages.keys()), weights=list(ages_percent.values()), k=200
6 )
```

Quelltext 5.2 – Häufigkeit des Alters

.

Ein Beispiel dafür: Für das Alter 55 bis 60. In der Altersgruppe 55 gibt es 1.381.357 Menschen in Deutschland. In der Altersgruppe 56 gibt es 1.402.572 Menschen in Deutschland. In der Altersgruppe 57 gibt es 1.385.607, 58 gibt es 1.336.643, 59 gibt es 1.311.139 und in der Altersgruppe 60 gibt es 1.261.935 Menschen in Deutschland. Somit gibt es insgesamt 30.907.091 in der Altersgruppe zwischen 55 und 60. Demzufolge liegt die Wahrscheinlichkeit, dass eine Person 55 ist in der Altersgruppe zwischen 55 und 60, bei 17 %; bei 56 17,4 %; bei 57 17,1 %, usw. Somit soll die Verteilung des Alters der 200 Personen der Realität gleichen.

Welches Geschlecht eine Person hat, wird statistisch durch das Statistische Bundesamt [Sta22] ermittelt. Wenn eine Person zwischen 50 und 60 Jahren ist, liegt demzufolge die Wahrscheinlichkeit, dass eine Person dem männlichen Geschlecht angehört, bei 50,2 %; beim weiblichen Geschlecht 49,8 %. So wird das ebenfalls für die Altersgruppen 60 bis 70, 70 bis 80, 80 bis 85 und älter als 85 durchgeführt. Die Wahrscheinlichkeiten können aus Tabelle 5.4 entnommen werden.

Tabelle 5.4 – Gewichtung des Geschlechts nach Altersgruppen [Sta22]

Altersgruppe	männlich	weiblich
50 bis 60	50,2 %	48,9 %
60 bis 70	48,4 %	51,6 %
70 bis 80	45,6 %	54,4 %
80 bis 85	41,4 %	58,6 %
älter als 85	32,4 %	67,6 %

Tabelle 5.5 – Gewichtung der ASH nach Altersgruppen nach Heger und Hubolu [HH10]

Altersgruppe	Normal-hörig-keit	geringe ASH	mittlere ASH	schwere ASH	gehörlos
45 bis 55	93,5 %	6,1 %	0,5 %	0 %	0 %
55 bis 65	84,1 %	14 %	1,7 %	0,2 %	0 %
65 bis 75	62,7 %	31,2 %	5,8 %	0,2 %	0 %
älter als 75	35,5 %	45,1 %	16,7 %	2,7 %	0 %

Durch die Studie von Heber und Hubolu [HH10] kann analysiert werden, welchen Schwerhörigkeitsgrad eine Person hat. In dieser Studie wurden die Grade der ASH in Altersgruppen aufgeteilt. Die Wahrscheinlichkeiten, bei welchem Alter welcher Grad der ASH auftritt, kann Tabelle 5.5 entnommen werden. Hierbei ist zu sehen, dass die Gehörlosigkeit nie auftreten wird.

Durch die Statistik der Deutschen Alzheimer Gesellschaft e.V. [Deu22] kann die Häufigkeit der kognitiven Erkrankung nach Alter bestimmt werden.

In Tabelle 5.3b werden vier Beispiele für die Benutzer gezeigt.

Daten zur Bewertung Da die Interaktion aufgrund einer Bewertung erstellt wird, müssen Daten generiert werden, wie die Benutzer die Interaktionen bewertet haben. Dafür wird zuerst bestimmt, welche Features eine Person benötigt. Zum Beispiel wird durch den Grad der ASH die Lautstärke bestimmt. Ist der Grad der ASH hoch, wird auch die Lautstärke höher. Ist der Grad der kognitiven Erkrankung hoch, wird eine langsamere Geschwindigkeit genutzt und die Verzögerung verlängert.

Die präferierte Lautstärke ist abhängig von dem Schweregrad der ASH. Hat die Person einen Grad von 0, kann die Lautstärke ebenfalls beim Wert 0 liegen. Hat die Person einen Grad von 1 oder 2, soll die Lautstärke bei 1 liegen. Bei allen anderen wird die Lautstärke auf 2 gesetzt.

Die präferierte Tonhöhe wird abhängig vom Alter und Geschlecht gemacht. Wiley et al. [Wil+08] zeigen, dass im steigenden Alter Änderungen im hörbaren Frequenzbereich liegen. Zum Beispiel können Personen im höheren Alter die Frequenzen zwischen 8 kHz bis 4 kHz erst ab eine Lautstärke von ca. 60 dB hören. Sie zeigen auch, dass es dabei leichte Unterschiede zwischen Frauen und Männer gibt: Bei Männern setzt die Änderung zeitiger ein als bei Frauen. Ein normales Gespräch hat in etwa eine Lautstärke von 40 - 50 dB. Demzufolge verstehen Männer von Grund auf tiefere Töne besser. Frauen hören den Frequenzbereich zwischen 6 und 8 kHz ab ca. 70 Jahren nur noch schwer. Somit bevorzugen Frauen eine tiefere Tonlage erst ab einem höheren Alter. Schließlich wird folgendes festgelegt:

- 50 - 60 Jahre: Männer bekommen die Tonhöhe 1 und Frauen die Tonhöhe 2 zugewiesen.
- 60 - 70 Jahre: Männer bekommen die Tonhöhe 1 und Frauen die Tonhöhe 1 zugewiesen.
- ab 70 Jahren: Männer und Frauen bekommen die Tonhöhe 0 zugewiesen.

Um die präferierte Wartezeit und die präferierte Geschwindigkeit zu bestimmen, werden auf die Aussagen von Fisk et al. [Fis+19; Lee15], dass im Alter Störungen im Kurzzeitgedächtnis und Sprachverständnis auftreten, Bezug genommen. Somit werden die präferierte Wartezeit und präferierte Geschwindigkeit anhand des Alters bestimmt. Umso Älter man ist, desto höher muss die Wartezeit und langsamer die Geschwindigkeit sein. Demzufolge werden folgende Werte gesetzt:

- 50 - 60 Jahre: Geschwindigkeit wird auf 2 und Wartezeit auf 0 gesetzt.
- 60 - 70 Jahre: Geschwindigkeit wird auf 1 und Wartezeit auf 1 gesetzt.
- 70 - 80 Jahre: Geschwindigkeit wird auf 0 und Wartezeit auf 1 gesetzt.

- ab 80 Jahre: Geschwindigkeit wird auf 0 und Wartezeit auf 2 gesetzt.

Ob eine einfache Sprache präferiert wird ist abhängig von der Kognition der Person. Ist der Wert der Kognition 1 wird auch die Verwendung der einfachen Sprache auf 1 gesetzt. Sollte diese 0 sein, soll auch die Verwendung der einfachen Sprache auf 0 gesetzt werden.

Nachdem präferierte Werte für die Person erstellt wurden, kann ermittelt werden, welche Bewertung die Person einer Interaktion geben wird. Dafür werden die Feature der zu bewertenden Interaktion genommen und mit den präferierten Werten verglichen. Zum Vergleichen wird die Differenz zwischen dem präferierten Wert eines Features und des eigentlichen Features genommen und daraus der Betrag gebildet. Sollte der präferierte Wert des Features mit dem Feature aus der Interaktion übereinstimmen, ergibt das einen Betrag von 0. Andererseits kann der Wert bei Unterschied einen Wert von 1 oder höher (maximal 2) annehmen.

Mit diesen Beträgen der Differenzen können nun die Wertungen erstellt werden. Die Wertungen können hierbei den Wert zwischen 1 und 5 annehmen. Demzufolge wird für jeden Betrag eine Wertung erstellt und durch die Anzahl aller Beträge geteilt. Es wird zunächst der Wert der Bewertung auf 0 gesetzt. Bei der Bewertung der Werte für die Lautstärke, Geschwindigkeit, Tonhöhe und Wartezeit wird wie folgt bewertet: Ist der Betrag der Differenz 0 so steigt die Bewertung um 5. Ist eine Abweichung von 1 gegeben steigt die Bewertung um 2,5. Dies hat folgenden Grund: Wenn eine Person eine Interaktion nutzt, die nicht vollständig zu den Einschränkungen passt, kann die Person mit der Interaktion dennoch umgehen, würde diese Interaktion jedoch nicht bevorzugen. Somit bekommt diese Interaktion eine schlechte Bewertung. Sollte die Person mit der Interaktion jedoch nicht umgehen können, muss diese Interaktion eine sehr schlechte Bewertung bekommen. Einen Betrag der Differenz des präferierten Wertes und des Features von 1 bedeutet, dass die Person mit der Interaktion umgehen kann, aber nicht die beste Bewertung geben würde.

Bei dem Feature der Verwendung der einfachen Sprache findet folgende Bewertung statt: Liegt der Betrag der Differenz bei 0 wird die Bewertung um 5 erhöht. Sollte der Wert der Bewertung nun immer noch 0 betragen, passt die Interaktion absolut nicht zu den Einschränkungen der Person. Demnach wurde diese Interaktion auch nie benutzt und konnte somit auch nicht bewertet werden.

Andernfalls wird nun der Wert der Bewertung durch 5 geteilt. So erhält man den richtigen Wert der Bewertung für die Interaktion von einer Person.

Beispiel der Bewertung an einer Person Es wird hierfür die Bewertung des Benutzers 2 aus Tabelle 5.3b für die Interaktion 2 aus Tabelle 5.3a durchgeführt. Die präferierten Werte der Features für die Person werden wie folgt ermittelt:

- **Präferierte Lautstärke**: Die Person hat einen Schweregrad der ASH von 3. Demnach erhält die präferierte Lautstärke den Wert 2.
- **Präferierte Tonhöhe**: Die Person hat ein Alter von 70 Jahren und ist männlich. Demnach erhält die präferierte Tonhöhe den Wert 0.
- **Präferierte Geschwindigkeit**: Die Person hat ein Alter von 70. Demnach erhält die präferierte Geschwindigkeit den Wert 0.
- **Präferierte Wartezeit**: Die Person hat ein Alter von 70. Demnach erhält die präferierte Wartezeit den Wert 1.

- **Präferierte Verwendung der einfachen Sprache**: Die Person hat bei der kognitiven Erkrankung den Wert 0. Demnach erhält die präferierte Verwendung der einfachen Sprache den Wert 0.

Nun können die Beträge der Differenzen der Features der Interaktion mit den präferierten Werten der Features berechnet werden. Dabei bekommt der Betrag der Differenz für die Lautstärke den Wert 1, für die Tonhöhe den Wert 0, für die Geschwindigkeit den Wert 2, für die Wartezeit den Wert 0 und für die Verwendung der einfachen Sprache den Wert 1. Nun kann die Bewertung berechnet werden: Die Bewertung besitzt zuerst den Wert 0. Da die Lautstärke einen Betrag der Differenz von 1 aufweist, steigt die Bewertung um 2,5. Der Betrag der Differenz der Tonhöhe hat den Wert 0, wodurch die Bewertung um 5 steigt. Die Geschwindigkeit weicht um 2 ab, weshalb die Bewertung bei 7,5 bleibt. Die Wartezeit hat eine Abweichung von 0, wodurch die Bewertung um 5 steigt. Der Betrag der Differenz für die Verwendung der einfachen Sprache hat den Wert 1, weshalb die Bewertung weiter bei 12,5 bleibt. Hier ist schon zu erkennen, dass einiges Abweicht, weshalb die Interaktion keine gute Bewertung bekommen wird. Der Wert der Bewertung wird nun durch 5 geteilt. Die Person würde demnach die Interaktion mit 2,5 Bewerten, da es einige Abweichungen gab.

5.5 Neural Colaborative Filtering

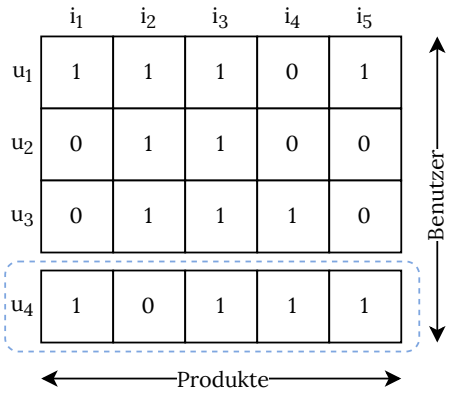
In diesem Abschnitt wird erklärt wie das gewählte Modell aus Abschnitt 4.2.1 genutzt werden kann. Hierfür wird erklärt, wie CF funktioniert und welche Einschränkungen dies haben kann. Weiter wird das Modell aufgeschlüsselt und gezeigt, wo welche Eingabedaten verarbeitet werden.

5.5.1 Collaborative Filtering

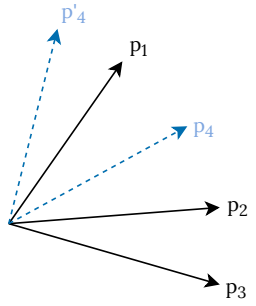
CF basiert auf der Annahme, dass ähnliche Benutzer ähnliche Produkte mögen. Wenn zum Beispiel Benutzer A Produkt A mag und Benutzer B dem Benutzer A ähnlich ist, wird Benutzer B wahrscheinlich auch Produkt A mögen. Die Benutzer sind ähnlich, wenn sie ähnliche Produkte mögen.

Ein Algorithmus um dies zu bewerkstelligen ist die Matrixfaktorisierung (MF). Laut He et al. [He+17] ordnet die MF jedem Benutzer und jedem Produkt ein reelwertigen Vetkor latenter Merkmale zu.

In Abbildung 5.2a zeigt eine solche Benutzer-Produkt-Matrix. Hierbei existieren drei Benutzer: u1, u2 und u3, welche mit unterschiedlichen Produkten interagiert haben. In Abbildung 5.2b wird der dazugehörige latente Raum der Benutzer dargestellt, wobei p1 der latente Vektor für u1, p2 der latente Vektor für u2 und p3 der latente Vektor für u3 darstellt. Es wird nun ein neuer Benutzer u4



(a) Benutzer-Produkt-Matrix



(b) latenter Vektoren der Benutzer

Abbildung 5.2 – Benutzer-Produkt-Matrix und latenter Benutzerraum[He+17]

hinzugefügt. Dieser interagierte mit den meisten Produkten wie u1 (drei von fünf Produkten). Da dieser u1 ähnlich ist, muss der latente Vektor p4 in der nähe von p1 sein.

In diesem Beispiel in Abbildung 5.2 werden auch die Grenzen der MF gezeigt, da dies laut He et al. [He+17] als lineares Modell latenter Faktoren angesehen wird. Es ist in Abbildung 5.2a erkennbar, dass u4 die größte Ähnlichkeit zu u1 hat. Die zweitmeiste Ähnlichkeit hat u4 zu u3, da zwei von fünf Produkten übereinstimmen mit denen die Benutzer interagiert haben. Die geringste Ähnlichkeit hat u4 zu u2, da nur ein Produkt übereinstimmt. Demzufolge sollte der latente Vektor von p4 dem latenten Vektor p1 am nächsten sein. Weiterhin sollte p4 neben p1 am nächsten zu p3 liegen. Hierbei ist erkennbar, dass jedoch p2 am nächsten ist, wenn p4 in der Nähe von p1 gelegt wird, was zu einem großen Rangverlust führt.

He et al. [He+17] stellen fest, dass eine Möglichkeit zur Lösung des Problems darin besteht, eine große Anzahl latenter Faktoren K zu verwenden. Dies kann jedoch nach Rendle [Ren10] die Verallgemeinerung des Modells beeinträchtigen, zum Beispiel durch eine Überanpassung der Daten. Um diesen Einschränkungen entgegen zu wirken, wird nach He et al. [He+17] ein KNN genutzt.

5.5.2 Neuronale Matrixfaktorisierung

Die Idee hinter der Neuronalen Matrixfaktorisierung ist, dass neben der Matrixfaktorisierung ein Neuronales Netz vewendet wird. He et al. [He+17] nutzen neben der Matrixfaktoriesierung ein sogenanntes MLP. Ein MLP ist in Abschnitt 3.2.2 erklärt und in Abbildung 3.5 dargestellt. Somit wird ein lineares Modell (MF) mit einem nicht linearen System (MLP) verbunden, um den Einschränkungen aus Abschnitt 5.5.1 entgegen zu wirken.

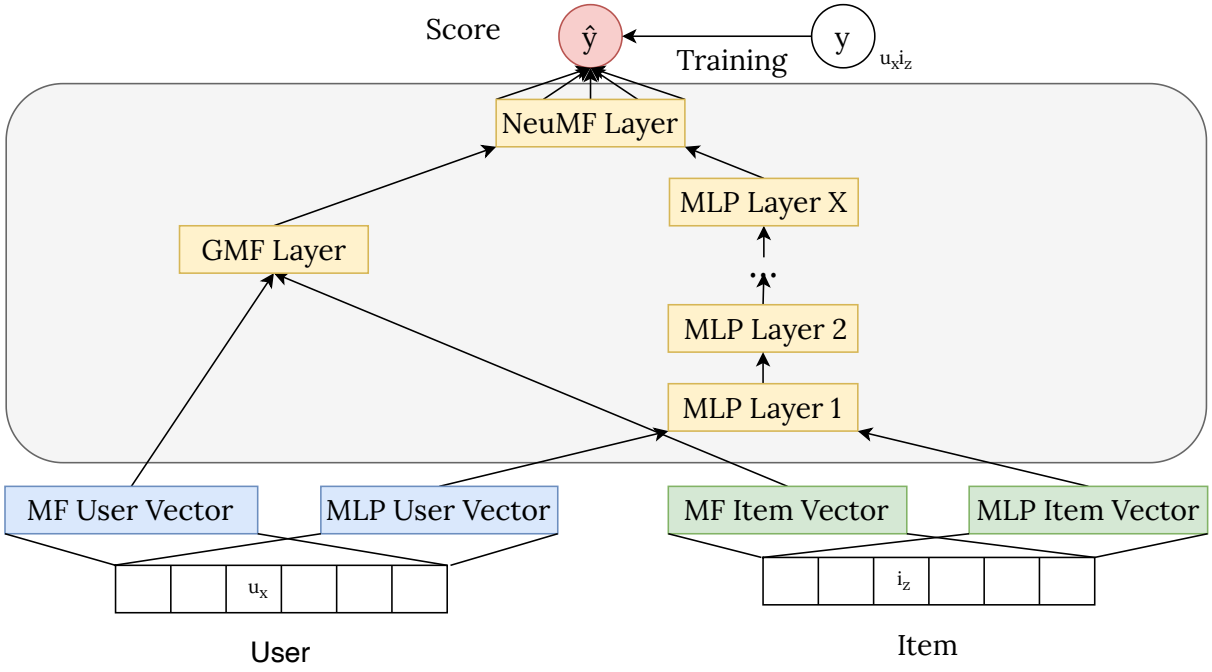


Abbildung 5.3 – Neuronale Matrixfaktorisierungs Modell nach He et al. [He+17]

In Abbildung 5.3 wird das Modell der Neuronalen Matrixfaktoriesierung gezeigt. Dabei werden nach He et al. [He+17] für die MF und MLP zwei unterschiedliche Embeddings-Layer genutzt, da

gemeinsame Embeddings-Layer die Leistung dieses Modells einschränken können.

Bei diesem Modell ist zu beachten, dass die voraussagende Bewertung nur durch existierende Bewertungen von Benutzern und Produkten berechnet wird, d. h. dieses Modell benötigt noch die Eigenschaften der Benutzer und die Features der Produkte.

5.5.3 Hybride neuronale Matrixfaktorisierung

Bei einer hybriden neuronalen Matrixfaktoriesierung werden neben der MF und MLP die Eigenschaften der Benutzer und die Features der Produkte mit einbezogen. Diesbezüglich können die Eigenschaften der Benutzer aus Abschnitt 5.2 entnommen werden. Hierbei handelt es sich um die Eingabedaten der Benutzer. Demzufolge sind die Eigenschaften *Alter*, *Geschlecht*, *Stadium ASH* und *Stadium der Kognition*.

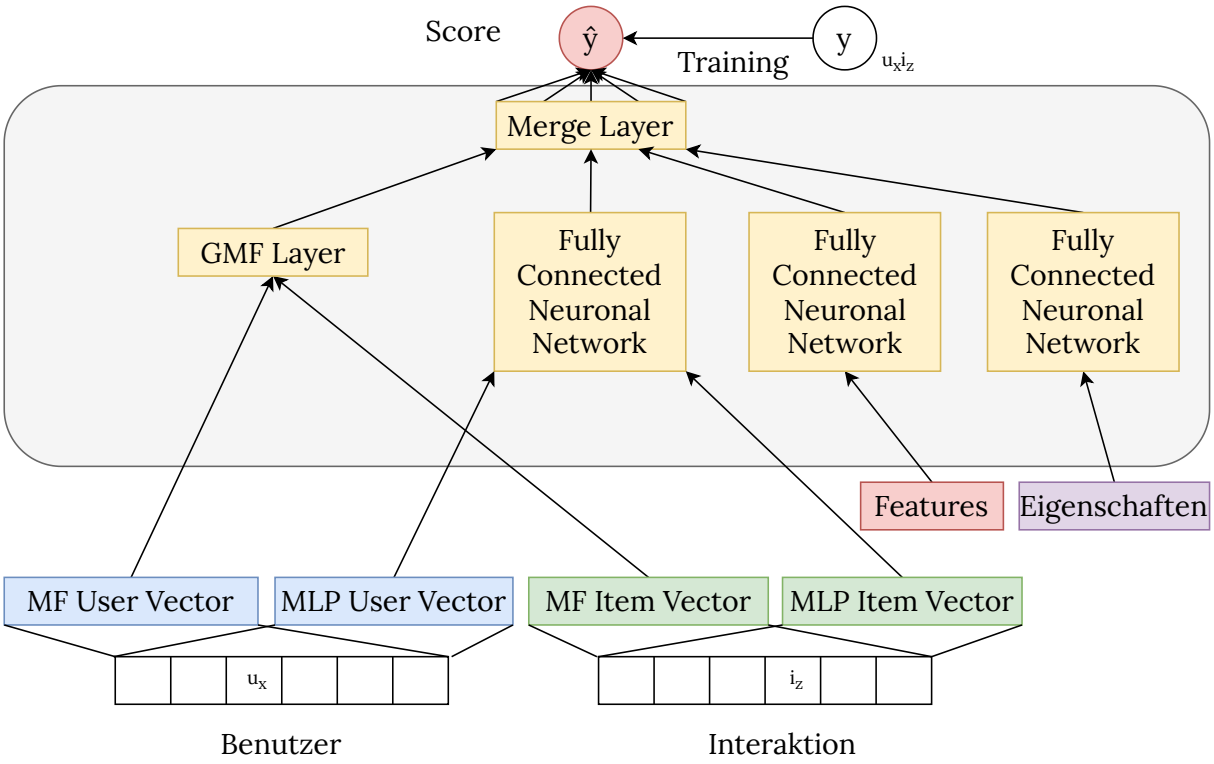


Abbildung 5.4 – Hybride Neuronale Matrixfaktorisierung

Die Features der Interaktion können ebenfalls aus Abschnitt 5.2 entnommen werden. Hierbei handelt es sich um die Eingabedaten der Interaktion, welche aus dem Feature-Modell aus Abschnitt 4.1.2 hergeleitet wurden. Unter den Features zählen die *Geschwindigkeit*, *Tonhöhe*, *Wartezeit*, *Lautstärke* und *einfache Sprache*.

Um die Eigenschaften der Benutzer und die Features der Interaktion mit zu verwenden, werden neben der MF und dem MLP zwei weitere neuronale Netze hinzugefügt, welche die Auswirkung der Eigenschaften und der Features beachten. In Abbildung 5.4 wird gezeigt, wie das hybride Modell aussieht. Bei der letzten Ebene werden die MF und die neuronalen Netze für CF, Eigenschaften und Feature zusammengefasst, sodass für einen Benutzer mit bestimmten Eigenschaften zu einer Interaktion mit bestimmten Features eine Bewertung vorhergesagt werden kann.

5.6 Aufbau des Modells

In diesem Abschnitt wird das Modell mithilfe von TensorFlow und Keras aufgebaut. Quelltext 5.3 zeigt in Zeile 1, welche Imports notwendig sind, um das Modell zu bauen. In Zeile 2 und 3 werden zwei Eingabeschichten definiert. Eine für die Benutzer und eine für die Interaktion.

```
1 from tensorflow.keras import models, layers, utils
2 xusers_in = layers.Input(name="xusers_in", shape=(1,))
3 xinteractions_in = layers.Input(name="xinteractions_in", shape=(1,))
```

Quelltext 5.3 – Eingabeschicht

In Quelltext 5.4 wird die Matrixfaktoriesierung erstellt. Hierbei werden zunächst Embedding-Schichten für die Benutzer und für die Interaktionen für die Matrixfaktoriesierung erstellt. In Zeile 2 zum Beispiel wird die Embedding-Schicht für den Benutzer erstellt. Dieser hat dabei eine Eingangsdimension von der Anzahl aller Benutzer in den Trainingsdaten und eine Ausgabedimension von 50. Die Embedding-Größe wurde hier festgelegt. Diese Embedding-Schicht soll für jeden Benutzer demzufolge einen Vektor von einer Größe von 50 anlegen, sodass dieser Vektor später berechnet werden kann. In Zeile 5 wird eine sogennante Reshape-Schicht aus der Embedding-Schicht angelegt. Diese verändert die Form des Vektors, sodass diese dann in Zeile 16 verknüpft werden kann. Analog wird dies mit den Interaktionen gemacht. In Zeile 16 wird nun eine Dot-Schicht aus den neu geformten Embedding-Schichten der Benutzer und der Interaktionen gebildet. In dieser Dot-Schicht wird ein Skalarprodukt berechnet, wodurch die Matrixfaktoriesierung durchgeführt wird.

```
1  ## embeddings and reshape
2  mf_xusers_emb = layers.Embedding(
3      name="mf_xusers_emb", input_dim=usr, output_dim=embeddings_size
4  )(xusers_in)
5  mf_xusers = layers.Reshape(name="mf_xusers", target_shape=(embeddings_size,))(
6      mf_xusers_emb
7  )
8  ## embeddings and reshape
9  mf_xinteractions_emb = layers.Embedding(
10     name="mf_xinteractions_emb", input_dim=ints, output_dim=embeddings_size
11 )(xinteractions_in)
12 mf_xinteractions = layers.Reshape(name="mf_xinteractions", target_shape=(embeddings_size,))(
13     mf_xinteractions_emb
14 )
15 ## product
16 mf_xx = layers.Dot(name="mf_xx", normalize=True, axes=1)([mf_xusers, mf_xinteractions])
```

Quelltext 5.4 – Matrix Faktorisierung

In Quelltext 5.5 wird ein MLP gebaut, welches neben der Matrixfaktorisierung durchgeführt werden soll. Zeile 1 bis 14 sind analog zum Quelltext 5.4. Hier werden jedoch die Vektoren für ein MLP erstellt. In Zeile 15 werden diese Schichten konkateniert. D.h. der Vektor für die Benutzer und der Vektor für die Interaktionen werden zu einem Vektor vereint. In Zeile 16 wird eine Dense-Schicht aus der konkatenierten Schicht erstellt. Eine Dense-Schicht ist eine voll verzweigte Schicht, in denen DL durchgeführt wird. `units` gibt dabei an, wie viele Neuronen verwendet werden. Als Aktivierungsfunktion wird hier *ReLU* verwendet.

```
## embeddings and reshape
nn_xusers_emb = layers.Embedding(
    name="nn_xusers_emb", input_dim=usr, output_dim=embeddings_size
)(xusers_in)
nn_xusers = layers.Reshape(name="nn_xusers", target_shape=(embeddings_size,))(
    nn_xusers_emb
)
## embeddings and reshape
nn_xinteractions_emb = layers.Embedding(
    name="nn_xinteractions_emb", input_dim=ints, output_dim=embeddings_size
)(xinteractions_in)
nn_xinteractions = layers.Reshape(name="nn_xinteractions", target_shape=(embeddings_size,))(
    nn_xinteractions_emb
)
## concat and dense
nn_xx = layers.Concatenate()([nn_xusers, nn_xinteractions])
nn_xx = layers.Dense(name="nn_xx", units=int(embeddings_size / 2), activation="relu")(
    nn_xx
)
```

Quelltext 5.5 – MLP

In Quelltext 5.6 werden MLPs für die Feature und Eigenschaften der Benutzer hinzugefügt. Demzufolge wird erst eine Eingabeschicht und danach eine Dense-Schicht erstellt. Als Aktivierungsfunktion wird hier wieder *ReLU* verwendet.

```
# Features of interactions
features_in = layers.Input(name="features_in", shape=(feat,))
features_x = layers.Dense(name="features_x", units=feat, activation="relu")(features_in)
# Properties
props_in = layers.Input(name="props_in", shape=(props,))
prop_x = layers.Dense(name="prop_x", units=props, activation="relu")(props_in)
```

Quelltext 5.6 – Hinzufügen von Feature und Eigenschaften der Nutzer

In Quelltext 5.7 werden nun alle Komponenten zusammengefügt. In Zeile 2 wird wieder eine Konkatenationsschicht aus der Matrixfaktorisierung aus Quelltext 5.4, des MLP aus Quelltext 5.5 und den MLPs aus Quelltext 5.6 erstellt. Danach wird wieder eine Dense-Schicht gebaut, wobei diesmal nur noch ein einziger Wert entstehen soll. Dieser ist die vorhergesagte Bewertung des Benutzers für eine Interaktion. Diesmal wird die Aktivierungsfunktion *linear* gewählt.

In Zeile 5 wird nun das Model gebildet, wobei hier die Eingabeschichten und Ausgabeschicht definiert werden. In Zeile 10 wird das Model zum Schluss kompiliert. Es wird hierfür der *Adams-optimierer* und *mittlerer absoluter Fehler* als Ausgleichungsfunktion verwendet. Diese wurden gewählt, da diese mit am häufigsten in DL benutzt werden.

```
# Merge all
y_out = layers.Concatenate()([mf_xx, nn_xx, features_x, prop_x])
y_out = layers.Dense(name="predict_rating", units=1, activation="linear")(y_out)
# Compile
model = models.Model(
    inputs=[xusers_in, xinteractions_in, features_in, props_in],
    outputs=y_out,
    name="Hybrid_Model",
)
```

```
10 model.compile(
11     optimizer="adam",
12     loss="mean_absolute_error",
13     metrics=["mean_absolute_percentage_error"],
14 )
```

Quelltext 5.7 – Zusammenfügen der MF MLP Features und Eigenschaften

5.7 Nutzung des Modells

In diesem Abschnitt wird erklärt, wie dieses Modell nun angewendet werden kann. Hierfür muss zunächst das Modell trainiert werden. Dafür werden die erstellten Eingabedaten aus Abschnitt 5.4 verwendet. Dabei wurden drei Tabellen erstellt: die Tabelle der Benutzer mit ihren Eigenschaften, die Tabelle mit den Interaktionen und deren Features sowie die Tabelle welche Bewertung ein Benutzer einer Interaktion gegeben hat. Diese sollten am besten in einer Tabelle vereint werden, mit der das Modell arbeiten kann.

```
1  import numpy as np
2  import pandas as pd
3  from sklearn import preprocessing
4  # read users
5  dtf_users = pd.read_csv("users.csv")
6  # read interactions
7  dtf_interactions = pd.read_csv("interactions.csv")
8  dtf_interactions = dtf_interactions.set_index("interactionID")
9  # read ratings
10 dtf_ratings = pd.read_csv("ratings.csv")
```

Quelltext 5.8 – Einlesen der Tabellen

In Quelltext 5.8 werden zunächst die Tabellen eingelesen. In Zeile 5 wird die Tabelle der Nutzer, in Zeile 7 die Tabelle der Interaktionen und in Zeile 10 die Tabelle der Bewertungen erfasst. In Zeile 8 werden die Indizes der Tabelle Interaktionen jeweils den Identifikationsnummern der Interaktionen zugeordnet. In den Tabellen 5.6a, 5.6b und 5.6c wird ein Auszug aus den Tabellen gezeigt.

```
1 dtf_users = dtf_users.merge(dtf_ratings, how="left")
2 _tmp = pd.DataFrame(
3     preprocessing.MinMaxScaler((0, 1)).fit_transform(dtf_users.values),
4     columns=dtf_users.columns,
5     index=dtf_users.index,
6 )
7 _test = dtf_users.copy()
8 dtf_users["rating"] = _tmp["rating"]
9 dtf_props = dtf_users[["userID", "age", "sex", "hoh", "kognition", "interactionID"]]
```

Quelltext 5.9 – Zusammenfügen der Tabellen Benutzer und Bewertung

In Quelltext 5.9 werden die Tabellen der Benutzer und der manuellen Bewertungen zusammen gefügt. Dabei werden die Bewertungen in Zeile 2 bis 6 auf einen Wert zwischen 0 und 1 runter gerechnet und in Zeile 8 werden die alten Bewertungen mit den neuen Bewertungen ersetzt. In

Tabelle 5.6 – Auszüge der Trainingsdaten

(a) Auszug der Benutzertabelle.

Benutzer ID	Alter	Geschlecht	Schweregrad der ASH	kognitive Erkrankung
9	85	f	1	0
50	55	m	0	0
83	74	m	0	0
29	64	m	0	0
41	77	m	0	0

(b) Auszug der Bewertungstabelle.

Interktions ID	Benutzer ID	manuelle Bewertung
102	52	3.0
148	91	2.5
38	90	3.5
92	43	3.5
111	17	3.0

(c) Auszug der Interaktionstabelle

Inderaktions ID	einfache Sprache	Lautstärke	Wartezeit	Tonhöhe	Geschwindigkeit
71	1	0	1	2	2
28	0	1	0	2	1
6	0	0	0	0	1
92	0	2	1	1	0
103	1	2	1	0	2

Zeile 9 wird eine Tabelle ohne die Bewertung erstellt. Diese wird später verwendet. In Tabelle 5.7 wird ein Auszug der zusammengeführten Tabelle dargestellt.

```
1 tmp = dtf_users.copy()
2 dtf_users = tmp.pivot_table(index="userID", columns="interactionID", values="rating")
3 missing_cols = list(set(dtf_interactions.index) - set(dtf_users.columns))
4 for col in missing_cols:
5     dtf_users[col] = np.nan
6 dtf_users = dtf_users[sorted(dtf_users.columns)]
```

Quelltext 5.10 – Erstellen einer Pivot-Tabelle

In Quelltext 5.10 wird eine Pivot-Tabelle erstellt. Hier werden fehlende Informationen ebenfalls hinzugefügt: Sollten alle Benutzer eine Interaktion nicht bewertet haben, kommt diese auch nicht in der Tabelle vor. Daher wird in Zeile 3 identifiziert, um welche Interaktion es sich handelt. In Zeile 4 und 5 werden die fehlenden Interaktionen hinzugefügt und mit `NaN` gefüllt. Ein Auszug dieser Pivot-Tabelle wird in Tabelle 5.8 gezeigt.

```
1 split = int(0.8 * dtf_users.shape[0])
2 dtf_train = dtf_users.loc[: split - 1, :]
3 dtf_test = dtf_users.loc[split:, :]
```

Quelltext 5.11 – Aufteilung in Test- und Trainingsdaten

Tabelle 5.7 – Auszug der Zusammengeführten Benutzer- und Bewertungstabelle

Benutzer ID	Interaktions ID	manuelle Bewertung	Grad der ASH	kognitive Erkrankung	Geschlecht	Alter
44	72	0.60	2	4	1	79
17	152	0.40	1	0	1	82
57	10	0.60	0	0	0	64
52	45	0.00	0	0	0	57
42	147	0.20	1	4	1	81

Interaktions ID / Benutzer ID	117	73	14	80	71
33	0.40	0.0	0.60	0.8	0.20
21	0.0	0.40	0.20	0.40	NaN
99	NaN	NaN	0.60	0.40	0.20
56	0.0	0.40	0.20	0.40	NaN
47	0.0	0.40	NaN	0.40	NaN

Tabelle 5.8 – Auszug der Pivot-Tabelle

In Quelltext 5.11 werden nun die Daten in Trainingsdaten und Testdaten aufgespalten. Hierbei werden ca. 80 % als Trainingsdaten und 20 % als Testdaten definiert. Diese Anzahl wird in Zeile 1 berechnet. In Zeile 2 werden infolgedessen die Trainingsdaten und in Zeile 3 die Testdaten festgelegt.

```
1 features = dtf_interactions.columns
2 props = dtf_props.drop(["userID", "interactionID"], axis=1).columns
3
4 train = dtf_train.stack(dropna=True).reset_index().rename(columns={0: "rating"})
5 train = train.merge(
6     dtf_interactions[features], how="left", left_on="interactionID", right_index=True
7 )
8 train = train.merge(dtf_props, how="left")
9
10 test = dtf_test.stack(dropna=True).reset_index().rename(columns={0: "rating"})
11 test = test.merge(
12     dtf_interactions[features], how="left", left_on="interactionID", right_index=True
13 )
14 test = test.merge(dtf_props, how="left")
15 usr_size, ita_size = dtf_users.shape[0], dtf_users.shape[1]
```

Quelltext 5.12 – Zusammenführen aller Daten

In Quelltext 5.12 werden nun die Trainings- und Testdaten vollendet, indem die Eigenschaften des Benutzers und die Feature der Interaktion mit hinzugefügt werden. Zunächst werden die Spalten für die Feature und der Eigenschaften in Zeile 1 und 2 bestimmt, wobei Zeile 2 später erst Verwendung findet. In Zeile 4 wird die Pivot-Tabelle rückgängig gemacht. In Zeile 5 bis 8 werden die Feature der Interaktionen und die Eigenschaften der Benutzer hinzugefügt. In Zeile 10 bis 14 wird dies Analog mit den Testdaten durchgeführt. In Zeile 15 werden die Anzahl der Benutzer und

die Anzahl der Interaktionen bestimmt, welche in Abschnitt 5.6 gebraucht werden. Diese ist die Größe der Eingangsdimensionen für die Benutzer und der Interaktionen.

```
1 model = model.fit(
2     x=[train["userID"], train["interactionID"], train[features], train[props]],
3     y=train["rating"],
4     epochs=300,
5     batch_size=128,
6     shuffle=True,
7     verbose=2,
8     validation_split=0.2,
9 )
10 values = model.predict(
11     [test["userID"], test["interactionID"], test[features], test[props]]
12 )
13 test["predict_rating"] = values
```

Quelltext 5.13 – Training und Anwendung des Modells

In Quelltext 5.13 wird nun das Modell trainiert. Dies passiert in Zeile 1 bis 9. Dabei werden die Trainingsdaten genutzt, die erstellt wurden. In der Variable `x` werden die Eingabedaten eingegeben. Dies sind in der Trainingstabelle jeweils die Benutzer, die Interaktionen, die Feature der Interaktionen und die Eigenschaften der Benutzer. `y` ist der Ausgabewert, welche in der Trainingstabelle die manuelle Bewertung ist. Weiterhin wird festgelegt, dass in 300 Epochen trainiert werden soll.

In Zeile 10 wird gezeigt, wie nun für Personen die Interaktion vorhergesagt werden kann. Sobald das Modell trainiert ist, werden ähnlich wie beim trainieren des Modells die Eingabedaten angegeben. Zurück kommt eine Liste mit allen Vorhersagen in der Reihenfolge, wie diese in den Eingangsdaten hineingegeben wurden. Demzufolge kann den Testdaten eine neue Spalte hinzugefügt werden mit den vorhergesagten Bewertungen des Modells. In Tabelle 5.9 ist ein Auszug der Testdaten zu sehen. Hier wurden auch die vorhergesagten Bewertungen mit eingefügt. Ähnlich, nur ohne die vorhergesagten Bewertungen, sehen auch die Trainingsdaten aus.

Tabelle 5.9 – Auszug aus der Testdatentabelle

Benutzer ID	**Interaktions ID**	**manuelle Bewertung**	**vorhergesagte Bewertung**	**Geschwindigkeit**	**Tonhöhe**	**Alter**	**Grad der ASH**	**...**
87	142	0.5	0.5563363	2	2	77	0	...
87	86	0.7	0.77737653	2	1	77	0	...
92	48	0.5	0.5533368	2	0	61	0	...
91	53	0.1	-0.031441346	2	2	78	1	...
99	95	0.3	0.108465314	0	2	85	1	...

In Abschnitt 5.1 wurde gezeigt, wo das Modell eingesetzt wird. In diesem Abschnitt wurde gezeigt wie es eingesetzt wird. Die vorhergesagten Bewertungen können nun nach Benutzer sortiert werden. Die Interaktionen, welche die höchste Bewertung hat, kann demzufolge eingesetzt werden, vorausgesetzt, dass keine manuelle Bewertung für die Interaktion existiert.

5.8 Zusammenfassung

In diesem Kapitel wurden die Eingabedaten und Ausgabedaten des DL-Modells beschrieben und definiert.

Weiterhin wurden Testdaten synthetisch erstellt, mit dem das DL-Modell trainiert werden soll. Dabei musste erklärt werden, woher die Daten stammen und wie die Daten genutzt werden konnten, um solche synthetischen Testdaten zu erstellen. Die Daten für die Interaktion wurden dabei aus dem Feature-Modell aus dem Kapitel 4 entnommen; Informationen zu den Personen wurden aus unterschiedlichen Studien zusammen getragen. Es wurde erklärt wie synthetisch eine Bewertung von den Personen zu den jeweiligen Interaktionen durchgeführt werden konnte.

Anschließend wurde schrittweise das Modell aus Abschnitt 4.2 in kleinen Teilen beschrieben und ein hybrides neuronales Matrixfaktorisierungsmodell erstellt, welches eine Bewertung vorhersagen soll, die aufgrund der Eigenschaften der Benutzer, Feature der Interaktionen und der Ähnlichkeit von Benutzern berechnet wird.

6 Implementierung des Prototypen

Mit dem Konzept aus Kapitel 5 kann nun ein Prototyp entwickelt werden. In diesem Kapitel werden Testhardware sowie Software kurz vorgestellt. Anschließend gibt es eine Dokumentation, wie das Konzept umgesetzt wurde.

6.1 Vorstellung des Roboters Loomo

Abbildung 6.1 – Segway Loomo[1]

Die Firma Segway ist bekannt geworden durch ihre Fortbewegungsmittel Segway Personal Transporter, welche sich anhand der Verlagerung des Körperschwerpunktes des Fahrers fortbewegen. Lehnt der Fahrer sich nach vorne, so fährt auch das Segway nach vorn. Analog bewegt sich der Segway nach hinten, wenn sich der Fahrer nach hinten lehnt. Dabei ist die Verlagerung des Körpergewichts der Sollwert für die Geschwindigkeit. Der Roboter Loomo basiert auf dem gleichen Prinzip wie die Segway Personal Transporter. In den Roboter wurde ein Intel Atom Z8750 als Prozessor, eine Intel RealSense ZR300-Kamera als Tiefenkamera, ein 4,3 Zoll großer Touchdisplay und andere diverse Sensoren eingebaut. Die Tiefenkamera ermöglicht, dass Personen und Gegenstände erkannt und wichtige Tiefenkoordinaten ermittelt werden, um z. B. Hindernissen auszuweichen. Als Betriebssystem nutzt der Roboter Android 5.1 Lollipop. Neben der Android SDK können Entwickler mithilfe des zusätzlichen Loomo SDK Apps entwickeln, welche auf bestimmte Sensoren und Funktionen des Loomos zugreifen können. [Sch18]

6.2 Vorstellung von Android

Android wurde ursprünglich von Android Inc. entwickelt, welches 2007 von Google aufgekauft und als Android Open Source Projekt veröffentlicht wurde. Darauf hin gründete sich das Konsortium Open Handset Alliance, welches die Entwicklung von Android fortführen und vertreiben soll. Darunter befinden sich mehrere Hardware-, Software- und Telekommunikationsunternehmen wie z. B. Google, Intel, NVIDIA und T-Mobile. Android ist ein Betriebssystem, welches auf unterschiedlichen Mobilen Geräten seinen Einsatz findet wie Smartphones, SmartTVs, Smartwatches und vielen mehr. [GS15].

[1] `https://de-de.segway.com/products/segway-loomo-robot` (aufgerufen am 30.07.2022)

6.3 Vorstellung MyCroft.ai

MyCroft.ai ist ein Open-Source Sprachassistent wie Alexa, Siri oder Google, wobei eine Firma mit gleichem Namen diesen entwickelt. MyCroft.ai unterscheidet sich von anderen Sprachassistenten in folgenden Punkten: Datenschutz, Anpassung, Benutzerverwaltung und Open Data. MyCroft.ai löscht Anfragen von Nutzern in Echtzeit, sodass diese nicht verarbeitet oder verkauft werden können. Außerdem bietet MyCroft.ai eine Anpassung an, sodass neben der Aktivierungsphrase auch die Stimme verändert werden kann. Außerdem repräsentiert MyCroft.ai den Endnutzer. Mit Open Data ist gemeint, dass nur veröffentlichte Daten zur Verbesserung des Sprachverständnis und der Sprachsynthese genutzt werden, für die Nutzer eingewilligt haben. [Rei22]

6.4 Vorstellung RASA

RASA ist ein Open-Source-Framework, welches ML nutzt, um automatisierte text- und sprachbasierte Konversationen zu führen. Demzufolge können mit RASA sogenannte Chatbots entwickelt werden. RASA selbst unterscheidet sich zwischen RASA PRO und RASA Open Source. Neben der text- und sprachbasierten Konversation bietet RASA Open Source die Möglichkeit, über verschiedene APIs eine Verbindung zu Messenger-Diensten von Drittanbietern herzustellen. RASA Pro hingegen baut auf RASA Open Source auf, bietet jedoch mehr Features, APIs und weitere Dienste, insbesondere für Unternehmen für Sicherheit, Beobachtbarkeit und Skalierung benötigen. [Ras22]

6.5 Ablauf des Prototypen

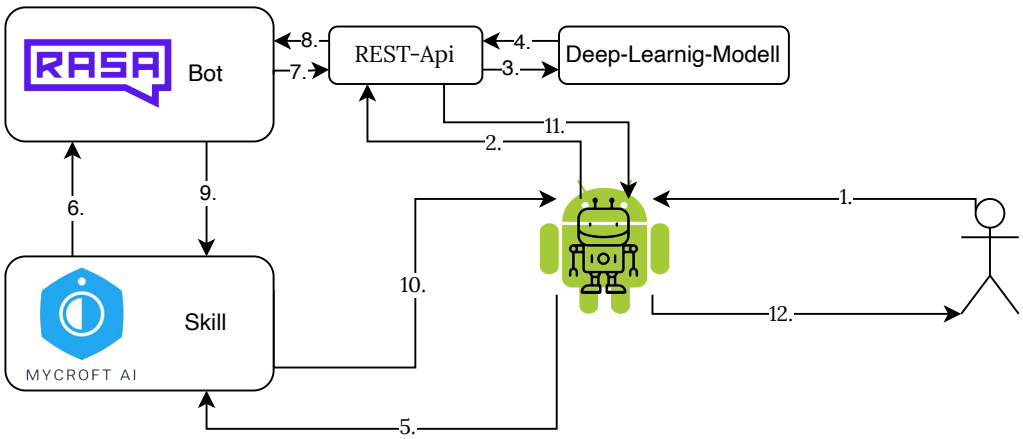


Abbildung 6.2 – Ablauf des Prototypen

In Abbildung 6.2 wird der Prozessablauf des Prototypen dargestellt. Als erstes (**1.**) interagiert ein Benutzer mit der Android-App auf dem Roboter. Hier wird gewählt, welche Person mit dem Roboter interagiert. Die Einschränkung dieser Person sind der Android-App bekannt. Es spielt dabei keine Rolle, woher die Android-App die Einschränkungen erhält, da diese Arbeit sich darauf bezieht, wie mit Hilfe der Einschränkung eine passende Interaktion gewählt werden kann.

Die Android-App gibt die Einschränkungen an eine *REST-API* weiter (**2.**). Die *REST-API* gibt die Informationen dem DL-Modell weiter (**3.**) und erhält eine Liste von Interaktionen, die für den Benutzer geeignet sind(**4.**). Die *REST-API* wird die erstbeste Interaktion wählen, falls keine andere Bewertung des Benutzers vorliegt und speichert diese.

Der Mycroft.ai-Skill ist eine Brücke für den RASA-Bot. Demnach kommuniziert der Mycroft.ai-Skill mit dem RASA-Bot (**6.**). Der RASA-Bot verarbeitet die gesagten Informationen. Außerdem holt sich der Bot die Informationen der Person von der *REST-API* (**7.** & **8.**). Nun übersendet der RASA-Bot die Antwort dem Mycroft.ai-Skill (**9.**), welche die Informationen der Android-App überreicht (**10.**).

Die Android-App fragt die *REST-API* ab, welche Interaktion verwendet werden soll (**11.**) und wird mit der ausgewählten Interaktion die Antwort an den Benutzer ausrichten (**12.**).

6.6 Erklärung der Bestandteile

In diesem Abschnitt werden die einzelnen Bestandteile näher erläutert und gezeigt, welche Funktionen diese bieten.

6.6.1 REST-API

Die *REST-API* ist zum Austausch der Information über den Benutzer und deren Interaktion zuständig. Sie startet außerdem das DL-Modell und speichert die Daten über einen Benutzer. Das Abrufen bzw. Speichern von Informationen wird über eine REST-Schnittstelle bewerkstelligt. Hier wird kurz dokumentiert, wie diese Schnittstelle Daten speichert und abruft.

`/user/<id:int>`

- `GET`:
  Mit dieser Funktion kann der Benutzer mit seinen Interaktionenen abgerufen werden. Als Antwort wird der Inhalt wie in Quelltext 6.1 gesendet.

```
{
  "userID": <int>,
  "sex": <string>,
  "hoh": <int>,
  "kognition": <int>,
  "age": <int>,
  "ratings": [
    {
      "interactionID": <int>,
      "interaction": {
        "interactionID": <int>,
        "delay": <int>,
        "pitch": <int>,
        "volume": <int>,
        "esay_language": <bool>,
      }
      "rating": <float|null>,
      "predict_rating": <float>
    }
  ]
```

```
21 }
```

Quelltext 6.1 – Dateninhalt des Benutzers bei GET

Die ratings sind hierbei nach der Bewertung sortiert. Die Interaktionen mit einer höheren Bewertung sind am Anfang der Liste platziert, die mit einer niedrigen Bewertung am Ende der Liste. Die Sortierung prüft hierbei die Werte rating und predict_rating. Hierbei wird geprüft, ob rating existiert. Wenn diese Bewertung existiert, wird diese für die Sortierung verwendet, sonst wird predict_rating für die Sortierung verwendet.

/user/<userID:int>/interaction/<interactionID:int>

- GET:
  Hiermit kann die Bewertung für eine Interaktion abgerufen werden. Sollte diese Bewertung nicht existieren, wird das DL-Modell eine Bewertung vorhersagen. Sollte der Benutzer oder die Interaktion nicht existieren, erhält man den Fehlercode 400. In Quelltext 6.2 wird gezeigt, wie ein erfolgreicher Abruf aussieht.

```
{
  "interaction": {
    "delay": <int>,
    "easy_language": <bool>,
    "interactionID": <int>,
    "pitch": <int>,
    "speed": <int>,
    "volume": <int>
  },
  "interactionID": <int>,
  "predict_rating": <float>,
  "rating": <float|null>,
  "userID": <int>
}
```

Quelltext 6.2 – Dateninhalt der Bewertung bei GET

- POST:
  Mit dieser Funktion kann eine manuelle Bewertung gespeichert werden. Dabei muss die gesetzte Bewertung mithilfe einer JSON mitgesendet werden. Die zu sendende JSON wird in Quelltext 6.3 dargestellt. Bei einem erfolgreichen Abruf erhält man die gleiche JSON wie bei einem GET-Aufruf. Dieser wird in Quelltext 6.2 dargestellt.

```
{
  "rating": <float>
}
```

Quelltext 6.3 – Dateninhalt zur Erstellung einer manuellen Bewertung bei POST

- DELETE:
  Mit der DELETE-Methode kann eine manuelle Bewertung entfernt werden. Hier muss nichts weiter mitgesendet werden. Man erhält dieselbe JSON wie bei einem GET-Aufruf.

/user/current

- GET:
  Mit dieser Funktion kann der aktuelle Benutzer abgerufen werden. Hierbei wird das gleiche Ergebnis geliefert wie in Quelltext 6.1 beschrieben.

- POST:
  Mit der POST-Methode kann der aktuelle Benutzer festgelegt werden. Hierfür muss eine JSON mitgesendet werden, welche die userID des Benutzers beinhaltet. In Quelltext 6.4 wird dieser Dateninhalt dargestellt. Als Ergebnis wird das gleiche Ergebnis wie in der GET-Methode geliefert.

```
1 {
2   "userID": <int>
3 }
```

Quelltext 6.4 – Dateninhalt zur Erstellung des aktuellen Benutzers bei POST

6.6.2 Mycroft.ai-Skill & Android-App

Der Mycroft.ai-Skill dient wie in Abschnitt 6.5 beschrieben als Brücke zwischen der Android-App und dem RASA-Bot. Dieser leitet alle Informationen von der Android-App zum RASA-Bot und vice versa. Hierbei wird zwischen der Android-App und dem Mycroft.ai-Skill ein Websocket verwendet. Zwischen dem Mycroft.ai-Skill und dem RASA-Bot findet die Kommunikation mittels REST-Schnittstelle statt. Mycroft.ai bietet hierbei eine App an, welche für die Implementierung genutzt und angepasst wurde.

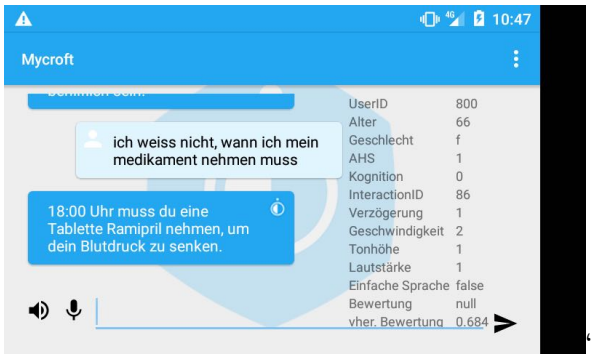


Abbildung 6.3 – Main-Activty der App

In der Abbildung 6.3 kann die Main-Activity der App betrachtet werden. Auf der linken Seite wird der Nachrichtenverlauf angezeigt. Jede Nachricht, welche von dem Mycorft.ai-Skill kommt, wird sprachlich mit der ausgewählten Interaktion ausgegeben. Auf der rechten Seite werden Informationen über den Benutzer, sowie über die ausgewählte Interaktion angegeben. Wird der Benutzer gewechselt, aktualisiert sich die Ausgabe. Im unteren Bereich können Eingaben durch Text oder Sprache erfolgen.

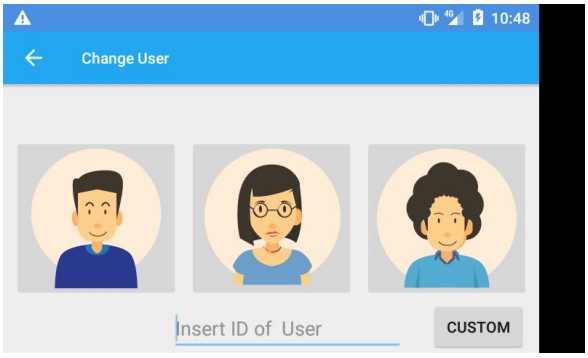


Abbildung 6.4 – Activity um den Benutzer zu wählen

Die Android-App hat noch mehr Funktionen. Mit dieser App wird der Benutzer ausgewählt. Die App übermittelt der REST-API die Daten des Benutzers und erhält die Information, welche Interaktion ausgeführt werden soll. Hierbei werden dann Tonhöhe, Lautstärke und Geschwindigkeit angepasst. Die einfache Sprache wird bei einer anderen Komponente benutzt. Bei der Eingabe muss auch die Wartezeit angepasst werden. Da hier die Google-Tastatur verwendet wird, kann dies nicht angepasst werden, weil die Google-Tastatur keine Schnittstelle dafür bietet. In Abbildung 6.4 wird gezeigt, wie die Activity aussieht, um einen Benutzer festzulegen. Hierbei wurden drei Voreinstellungen implementiert. Es ist möglich durch das Texteingabefeld am unteren Rand einen spezifischen Benutzer auszuwählen. Hierfür muss die `userID` des Benutzers angegeben werden.

In Abbildung 6.5 wird die Activity für die Einstellungen gezeigt. Hier müssen die IP-Adressen für Mycroft.ai und für die REST-API angegeben werden. Wenn diese nicht vorhanden sind, kann die Android-App keinen Dialog starten.

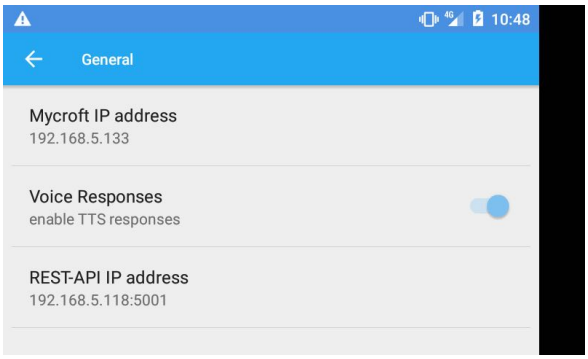


Abbildung 6.5 – Einstellungen der Android-App

Mycroft.ai-Skill Ein Mycroft.ai-Skill wird in der Regel mit einem Aktivierungswort gestartet. Demzufolge müsste dieses Wort jedes mal aufs Neue gesagt werden, um den Mycroft.ai-Skill zu starten, welcher die Brücke zu RASA aufbaut. Um dies zu umgehen und direkt mit RASA den Dialog führen zu können, wurde entschieden, einen sogenannten Mycorft.ai-Fallback-Skill zu implementieren. Sobald Mycroft.ai mitbekommt, dass keine Mycroft.ai-Skill aktiv ist und der eingegangene Text nicht zu einem Mycroft.ai-Skill passt, wird dieser Text direkt an den Mycroft.ai-Fallback-Skill gesendet, welcher den Text direkt RASA mitteilt.

In Quelltext 6.5 wird der Quellcode des gesamten Mycroft.ai-Fallback-Skill gezeigt. In Zeile 9 und 10 werden die IP-Adressen für die RASA-API und für die REST-API gesetzt. Ab Zeile 13 wird der Quellcode ausgeführt, wenn Mycroft.ai den Text keinem anderen Mycroft.ai-Skill zuordnen kann. Dafür wird zunächst der Text in der Variable `utterance` gespeichert. Danach wird die REST-API nach dem aktuellen Benutzer gefragt. Somit kann, wenn die RASA-API benachrichtigt wird, der sogenannte Sender eindeutig der `userID` des Benutzers zugeordnet werden. Ab Zeile 22 werden die Nachrichten von RASA gespeichert und ab Zeile 25 werden diese Nachrichten weiter an die Android-App weitergegeben.

```
1 from mycroft import FallbackSkill
2 import requests
```

```
3
4  class RasaFallback(FallbackSkill):
5      def __init__(self):
6          super(RasaFallback, self).__init__("Rasa FallbackSkill")
7
8      def initialize(self):
9          self.rasa_api = "http://172.17.0.1:8000/webhooks/rest/webhook"
10         self.rest_api = "http://192.168.5.118:5001/user/current"
11         self.register_fallback(self.handle_fallback, 10)
12
13     def handle_fallback(self, message):
14         utterance = message.data.get("utterance")
15         self.messages = []
16         response = requests.get(self.rest_api)
17         userdata = response.json()
18         userid = userdata["userID"]
19         data = requests.post(
20             self.rasa_api, json={"message": utterance, "sender": f"user{userid}"}
21         )
22         for next_response in data.json():
23             if "text" in next_response:
24                 self.messages.append(next_response["text"])
25         if len(self.messages) > 0:
26             for rasa_message in self.messages:
27                 self.speak(rasa_message)
28         return True
29
30     def shutdown(self):
31         self.remove_fallback(self.handle_fallback)
32         super(FallbackSkill, self).shutdown()
33
34 def create_skill():
35     return RasaFallback()
```

Quelltext 6.5 – Mycroft.ai-Fallback-Skill

6.6.3 DL-Modell

Das DL-Modell ist für die Vorhersage der Bewertung für die Interaktionen verantwortlich. Hierbei wurde das Modell aus Kapitel 5 leicht angepasst. Es wurden mehrere Dense Layer hinzugefügt, um eventuell bessere Ergebnisse zu erhalten.

Für das Modell wurde eine Speicher- und Ladefunktion implementiert, sodass ein gelerntes Modell wiederverwendet werden kann. Außerdem ist es möglich, vortrainierte Modelle zu laden, sodass beim Start des Programms keine lange Wartezeit gebraucht wird. In Quelltext 6.6 wird gezeigt, wie die Initialisierung des Modells aussieht. Wenn ein Modell verfügbar ist, wird es geladen. Wenn kein Modell verfügbar ist, muss es erst gebaut werden. Weiterhin wird hier die Möglichkeit gegeben, eine Intel VPU als Backend zu nutzen, um auf leistungsschwachen Geräten das Traineren des Modells zu beschleunigen.

```
1 def __init__(self, openvino=False, path: str = "./ncf_model", load: bool = False):
2     if importlib.util.find_spec("openvino_tensorflow") and openvino:
3         import openvino_tensorflow
4
```

```
5         if "MYRIAD" in openvino_tensorflow.list_backends():
6             openvino_tensorflow.set_backend("MYRIAD")
7             print("set backend to MYRIAD")
8     self.model: models.Model
9     self.isLoaded: bool = False
10    if os.path.exists(path) and load:
11        self.loadModel(path)
12        self.isLoaded = True
13        print("loaded model")
```

Quelltext 6.6 – Initialisierung des DL-Modells

Das Modell kommt jedes mal erneut zum Einsatz, wenn die REST-API einen Benutzer abfragt. In Abbildung 6.6 wird der Ablauf zur Abfrage eines Benutzers dargestellt. Zunächst wird der Benutzer aus der Datenbank gelesen. Im Anschluss werden alle Interaktionen aus der Datenbank gelesen. Damit das DL-Modell später Vorhersagen zu den Bewertungen treffen kann, müssen diese geladenen Objekte in ein geeignetes Format überführt werden. Daher werden die Interaktionen in einen Dataframe der Python-Library Pandas konvertiert. Analog wird dies mit dem Benutzer getan, sodass zu jeder Interaktion in dem Dataframe der Interaktionen der Benutzer mit seinen Eigenschaften hinzugefügt werden kann. Mit dem erstellten Dataframe ist nun das DL-Modell in der Lage, Vorhersagen für die Bewertungen zu treffen. Anschließend wird von dem Benutzer zu jeder Interaktion ein Bewertungsmodell in der Datenbank gesucht. Das Bewertungsmodell beinhaltet die `userID` des Benutzers, `interactionID` der Interaktion, die vorhergesagte Bewertung, die manuelle Bewertung sowie die Features der Interaktion. In Quelltext 6.2 wurde so ein Bewertungsmodell dargestellt. Sollte so ein Bewertungsmodell nicht existieren, wird es angelegt. Dabei wird die manuelle Bewertung auf `null` gesetzt. Die vorhergesagte Bewertung wird angepasst und das Bewertungsmodell wird gespeichert. In dem Fall, dass das Bewertungsmodell existiert, bleibt die manuelle Bewertung unberührt und es wird die vorhergesagte Bewertung angepasst. Somit ist gegeben, wenn das DL-Modell weiter trainiert wird, dass die vorhergesagten Bewertungen für die Interaktionen erneuert werden.

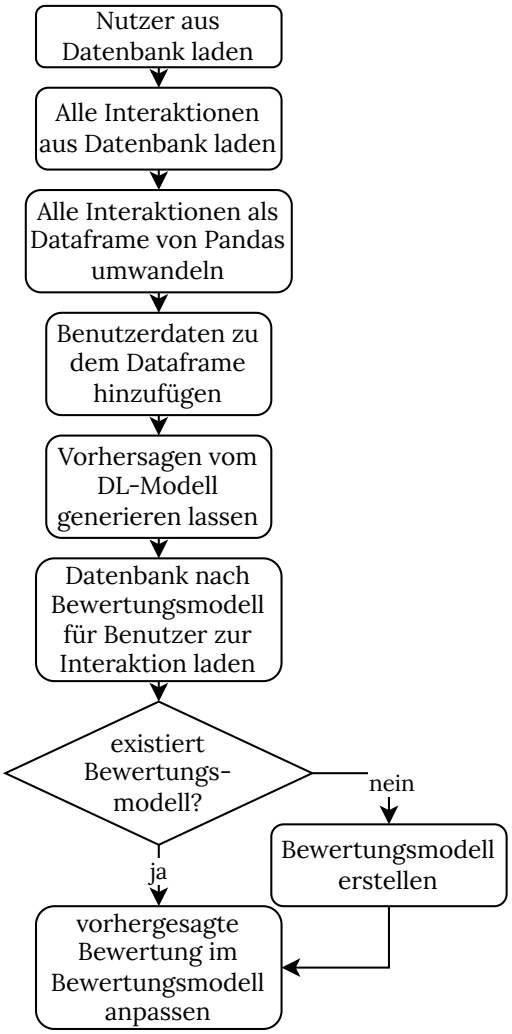


Abbildung 6.6 – Ablauf beim Abfragen eines Benutzers

6.6.4 RASA-Bot

Der RASA-Bot ist das Kernstück für einen Dialog. Dieser wertet die Texte aus und baut somit eine Antwort zusammen. Es wurden folgende zwei Dialoge für den Prototypen ausgewählt:

- Erfragen, wann ein Medikament genutzt werden soll.
- Einen Anruf mit einer Person starten.

Diese Dialoge sind im Prototypen statische Dialoge. Es wird nicht ausgerechnet, wann ein Medikament und welches genutzt werden soll. Weiterhin wird gesagt, dass der Anruf mit einer Person startet, doch es wird kein echter Anruf durchgeführt.

Es sind mehrere sogenannte Intents implementiert. In diesen Intents werden Textbeispiele angegeben, damit RASA später trainieren kann, wann welches Intent ausgelöst wird. Unter anderem existieren die Intents `greet`, `medicine call` und `end_greet`. In Quelltext 6.7 wird das `medicine`-Intent beispielhaft gezeigt.

```
1 - intent: medicine
2   examples: |
3     - wann ist das nächste medikament dran
4     - wann muss ich das nächste medikament einnehemen
5     - medikament
6     - wann medikament
7     - medizin
8     - wann muss ich medizin einnehmen
```

Quelltext 6.7 – Beispiel des `medicine`-Intents

Es werden Stories erstellt, welche einen Ablauf des Dialogs geben. In Quelltext 6.8 wird so eine Story gezeigt. Wenn das `medicine`-Intent ausgelöst wird soll die Aktion `action_medicine` starten. In dieser Aktion wird geprüft, ob eine einfache Sprache genutzt werden soll.

```
1 - story: medicine
2   steps:
3     - intent: medicine
4     - action: action_medicine
```

Quelltext 6.8 – Beispiel der `medicine`-Story

In Quelltext 6.9 wird gezeigt, wie so eine Aktion aussieht. Die Aktion überprüft, ob eine einfache Sprache genutzt werden soll. Hierfür wird eine Anfrage an die *REST-API* gesendet. Dabei wird die Interaktion mit der höchsten Bewertung genutzt. In Zeile 12 wird ausgelesen, ob die Interaktion die einfache Sprache nutzt. Wird diese genutzt, kann eine Antwort aus der Vorlage `utter_medicine_easy` verwendet werden. Wenn nicht, wird eine Antwort aus der Vorlage `utter_medicine_difficult` genutzt.

```
1 url = "http://192.168.5.118:5001/user/current"
2
3 def get_easy_language() -> bool:
4     try:
5         response = requests.get(url)
6         if response.status_code != 200:
7             print(f"wrong status code: {response.status_code}")
8             return True
```

```
 9         data = response.json()
10         user = data["userID"]
11         interaction = data["ratings"][0]["interactionID"]
12         easy_language = data["ratings"][0]["interaction"]["easy_language"]
13         print(f"user: {user}, interaction: {interaction}, easy: {easy_language}")
14         return easy_language
15     except:
16         traceback.print_exc()
17         return True
18
19 class ActionMedicine(Action):
20     def name(self) -> Text:
21         return "action_medicine"
22
23     def run(
24         self,
25         dispatcher: CollectingDispatcher,
26         tracker: Tracker,
27         domain: Dict[Text, Any],
28     ) -> List[Dict[Text, Any]]:
29         if get_easy_language():
30             dispatcher.utter_message(response="utter_medicine_easy")
31         else:
32             dispatcher.utter_message(response="utter_medicine_difficult")
33         return []
```

Quelltext 6.9 – Beispiel einer Aktion

Quelltext 6.10 zeigt einen Auszug aus den Vorlagen für die einfache Sprache und der Alltagssprache. Hier ist zu erkennen, dass für die einfache Sprache einfache Sätze definiert sind, während unter Alltagssprache komplexere Sätze genutzt werden.

```
1  utter_medicine_easy:
2    - text: "Du musst eine halbe Tablette Ramipril nehmen. Um 9:00 Uhr."
3    - text: "Du musst eine Tablette Ramipril nehmen. Um 18:00 Uhr."
4  utter_medicine_difficult:
5    - text: "Als nächstes ist Ramipril dran. Du solltest eine halbe Tablette gegen 9:00 Uhr
6    nehmen."
7    - text: "18:00 Uhr muss du eine Tablette Ramipril nehmen, um dein Blutdruck zu senken."
```

Quelltext 6.10 – Auszug der Beispiele der Antworten

6.7 Zusammenfassung

In diesem Kapitel wurde gezeigt, wie der Prototyp implementiert ist. Zunächst wurden der Roboter Loomo, Android, MyCroft.ai und RASA vorgestellt und kurz erläutert. Weiterhin ist der Ablauf des Prototypen erklärt worden. Hierbei wird festgestellt, dass mehrere Komponenten zuständig sind, damit ein Dialog mit dem Roboter durchgeführt werden kann. Diese Komponenten sind die Android-App, ein MyCroft-Skill, ein RASA-Bot sowie die REST-API, welche das DL-Modell enthält.

Jede Komponente wurde erklärt und erläutert, welche Aufgaben diese zu erfüllen hat. Es wurden wichtige Abläufe erklärt, sowie die Anwendung dokumentiert. Mit diesem Prototypen können nun Tests durchgeführt werden.

7 Test des Konzepts

Das Ziel der Tests ist zu zeigen, dass das entwickelte Konzept seine Aufgaben richtig erfüllt. Hierfür soll der entwickelte Prototyp aus dem vorigen Kapitel verwendet werden. Da hier ein neues technisches Konzept vorgestellt wird, welches auf generierten Daten trainiert ist, soll keine Nutzerstudie durchgeführt werden. Es muss geprüft werden, ob die Anforderungen aus Abschnitt 4.3 erfüllt werden. Zunächst soll das Ziel der Tests beschrieben werden. Danach kann eine Planung der Tests stattfinden. Nach der Durchführung der Tests erfolgt eine Auswertung der Daten.

7.1 Ziel der Tests

Bevor Tests durchgeführt werden können, muss definiert sein, welche Ziele damit erreicht werden sollen. Hierfür sollen die erstellten Anforderungen aus Abschnitt 4.3 geprüft werden, um zu sehen, ob diese erfüllt wurden. Jede Anforderung benötigt hierbei ein Ziel, welches beschreibt, wann eine Anforderung erfüllt ist. Hierbei werden mehrere Toleranzwerte definiert. Nach der statistischen Signifikanz wird üblicherweise eine Toleranz von 5 % definiert. Diese Toleranz ist jedoch frei wählbar. Da nicht genau bekannt ist, welche Toleranz aktzeptabel ist, werden dazu die Toleranzen 10 % und 20 % ebenfalls definiert. Die Ziele werden in nachfolgender Liste beschrieben.

- Anforderung **Gültigkeit** ist erfüllt, wenn
  - für jede Person eine Liste von Interaktionen mit vorhergesagten Bewertungen existiert.
  - jede Interaktion in der Liste einer validen Konfiguration entspricht.
    Eine valide Konfiguration wird in Abschnitt 4.1.2 und 5.2 erklärt. In Tabelle A.2 werden alle validen Konfigurationen gezeigt. Eine Interaktion besitzt eine valide Konfiguration, wenn diese in dieser Tabelle enthalten ist.
- Anforderung **Genauigkeit** ist erfüllt, wenn
  - die manuelle Bewertung von einer Person für eine Interaktion der vorhergesagten Bewertung gleicht.
    Hierbei werden die Toleranzwerte bei 0,05, 0,1 und 0,2 definiert. Dies entspricht der Toleranzen von 5 %, 10 % und 20 %.
- Anforderung **Ähnlichkeit** ist erfüllt, wenn
  - Personen, welche die gleichen Eigenschaften besitzen, für eine neue Interaktion ähnliche Bewertungen vorhergesagt bekommen.
    Hier liegen erneut die Toleranzwerte für die vorhergesagten Bewertungen bei 0,05, 0,1, 0,2. Dies entspricht der Toleranz von 5 %, 10 % und 20 %.
    Die Eigenschaften von Personen sind gleich, wenn alle folgende Punkte zutreffen:
    * *Alter*, *Geschlecht*, *Stadium der* ASH und *kognitive Erkrankung* stimmen überein.

- Anforderung **Anordnung** ist erfüllt, wenn
  - die Liste der manuell bewerteten Interaktionen mit der Liste der vorhergesagten bewerteten Interaktionen ähnlich ist.
    Ähnlich bedeutet hierbei, dass diese Listen nicht vollständig identisch sind. Es ist möglich, dass das Modell bessere oder schlechtere Vorhersagen bringt, da die Vorlieben des Nutzers nicht beachtet werden.

7.2 Durchführung der Tests

Für die Tests wird der Testdatensatz verwendet, welcher in Abschnitt 5.7 erzeugt wurde. Mit diesem Testdatensatz können nun die Tests für jede Anforderung geplant und durchgeführt werden.

7.2.1 Test der Gültigkeit

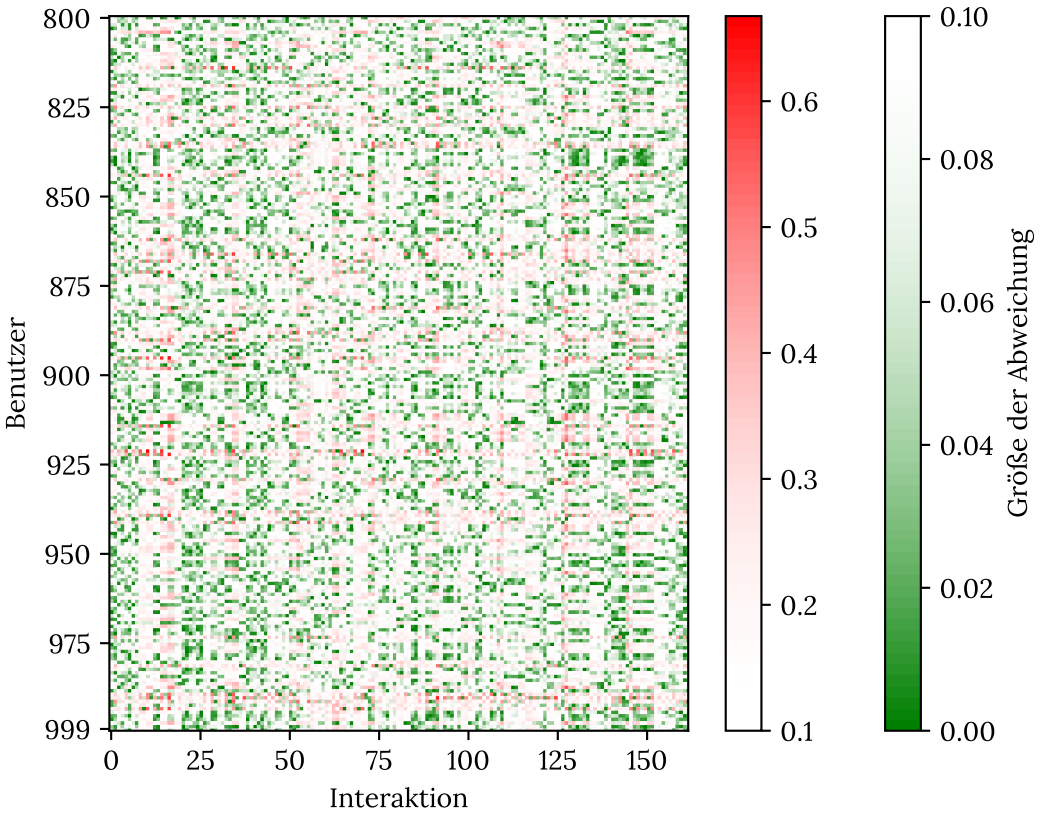
Um die Anforderung **Gültigkeit** zu testen, wird für jeden Benutzer aus dem Testdatensatz die Liste der bewerteten Interaktion von der *REST-Api* angefordert. Die Interaktionen in der Liste können mit der Tabelle A.2 abgeglichen werden. In dieser Tabelle werden alle Interaktionen mit ihren jeweiligen Features gelistet. Sollte eine Interaktion nicht in der Tabelle vorkommen, ist diese Interaktion keine valide Konfiguration und das Ziel wurde nicht erfüllt.

Ergebnis Es wurde festgestellt, das jede Interaktion in den Listen der Benutzer vorhanden ist. In der Tabelle A.1 werden alle Benutzer mit ihren Eigenschaften gelistet. Weiterhin beinhaltet die Tabelle die Spalte **Liste ist valide**, welche aussagt, ob die gegebene Liste valide ist. Da die Listen mit den Interaktionen sehr lang sind, werden diese nicht in der Tabelle mit aufgeführt. Da jede Interaktion in jeder Liste eine valide Konfiguration ist, wird diese Anforderung als bestanden betrachtet.

7.2.2 Test der Genauigkeit

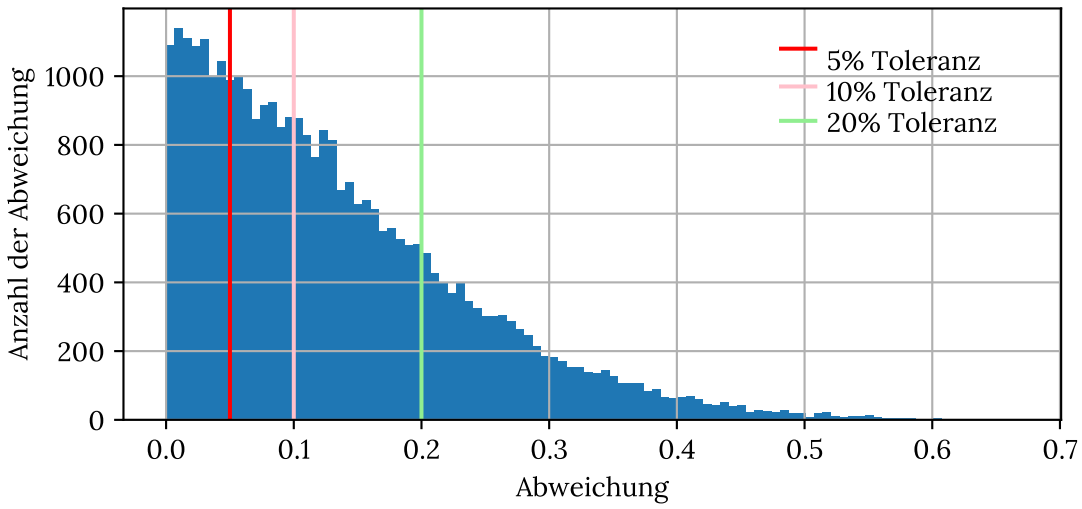
Um die Anforderung **Genauigkeit** zu testen, soll für jeden Benutzer des Testdatensatzes die manuelle Bewertung mit der vorhergesagten Bewertung verglichen werden. Hierfür wird für jeden Benutzer zu jeder Interaktion eine Distanz berechnet, welche zwischen der manuellen Bewertung und der vorhergesagten Bewertung der Interaktion existiert. Da Toleranzwerte definiert sind, kann gesagt werden, ob die vorhergesagte Bewertung korrekt bewertet wurde. Für jeden Toleranzwert wird für jeden Benutzer die Anzahl der korrekten vorhergesagten Bewertungen und der nicht korrekten vorhergesagten Bewertungen ermittelt. Außerdem wurde die durchschnittliche Abweichung zwischen manueller Bewertung und vorhergesagter Bewertung berechnet.

Ergebnis und Interpretation In Abbildung 7.1 wird eine Heatmap dargestellt. Diese zeigt an, wie niedrig die Abweichung der Bewertung von einem Benutzer zu einer Interaktion ist. Dunkelgrün bedeutet, dass die Abweichung sehr niedrig ist. Weiß bedeutet, dass die Abweichung den Wert von 0,1 erreicht hat. Dunkelrot bedeutet, dass eine Abweichung von ca. 0,7 erreicht wurde. Der Wert 0,1 wurde in dieser Grafik als Wendepunkt genommen, da dies ungefähr die durchschnittliche Abweichung darstellt. Aus der Abbildung kann entnommen werden, dass die durchschnittliche Abweichung bei einem Wert von 0,133 liegt.

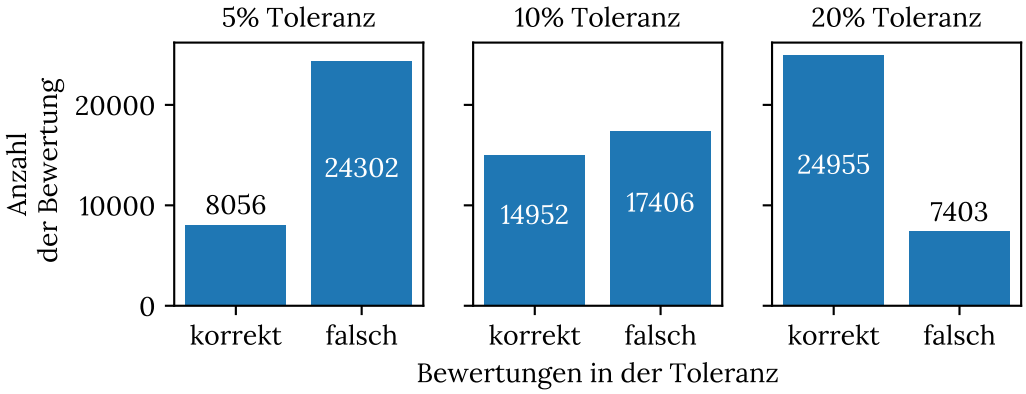


Durchschnittliche Abweichung: 0.133
Maximale Abweichung: 0.668
Minimale Abweichung: 0.0

Abbildung 7.1 – Abweichungen der Bewertungen



(a) Histogramm der Abweichungen zwischen manueller und vorhergesagter Bewertung



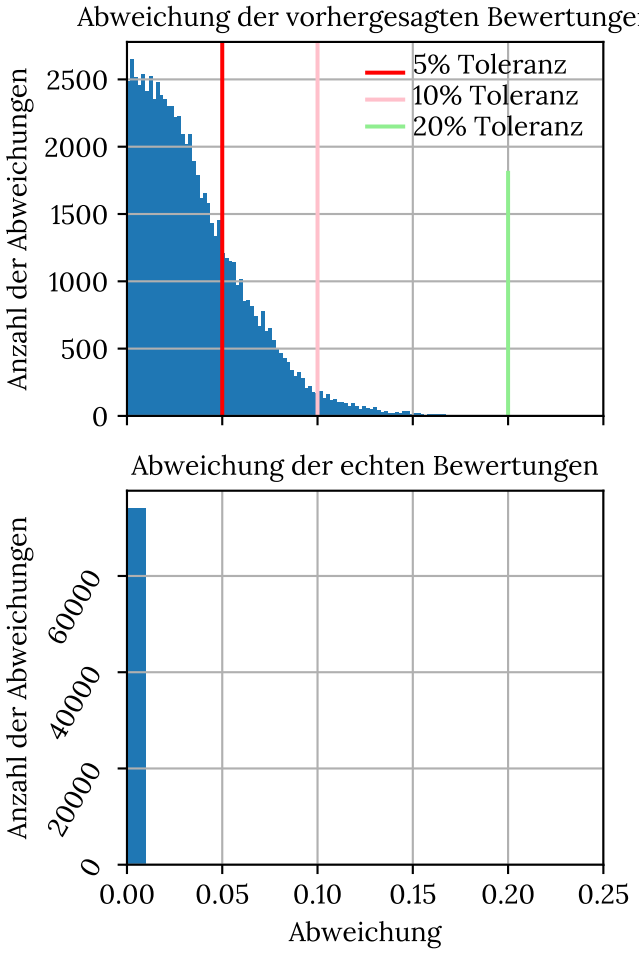
(b) Anzahl der korrekten und falschen Bewertungen

In Abbildung 7.2a wird ein Histogramm gezeigt, welches die Anzahl aller Abweichungen aufweist. Hierbei ist zu sehen, dass deutlich mehr Abweichungen existieren, welche den Toleranzwert von 0,1 übersteigen. Bei dem Toleranzwert von 0,2 erkennt man, dass mehr Abweichungen existieren, welche den Toleranzwert von 0,2 nicht übersteigen. Allgemein zeigt diese Grafik, dass umso geringer die Abweichung wird, desto größer wird die Anzahl Abweichungen. Mit anderen Worten: Umso höher die Abweichung wird, desto geringer wird die Anzahl der Abweichungen.

In Abbildung 7.2b wird gezeigt, wie viele Bewertungen innerhalb der Toleranz bewertet wurden und wie viele außerhalb der Toleranz liegen. Hierbei wurde für jeden Toleranzwert eine Grafik erstellt. Diese Abbildung soll verdeutlichen, dass sich je nach Toleranzwert die Anzahl der korrekten Bewertungen verändert.Man kann erkennen, dass der Wendepunkt bei ca. 10 % liegt.

Aus den gewonnen Erkenntnissen ist zu schließen, dass eine Unbekannte existiert. Dies ist die Toleranz. Es muss abgewogen werden, welche Toleranz akzeptabel ist.

7.2.3 Test der Ähnlichkeit



Durchschnittliche Abweichung: 0.035
Maximale Abweichung: 0.19
Minimale Abweichung: 0.0

Abbildung 7.3 – Histogramm der Abweichungen zwischen vorhergesagten Bewertungen und zwischen manuellen Bewertungen

Um die Anforderung der **Ähnlichkeit** zu testen, sollen für jeden Benutzer ähnliche Benutzer gefunden werden. Hierfür ist definiert, dass ein Benutzer ähnlich ist, wenn die Eigenschaften des Benutzers *Geschlecht*, *Stadium der ASH* und *kognitive Erkankung*, sowie das *Alter* identisch sind. Für jeden ähnlichen Benutzer zu einem Benutzer können nun für jede Interaktion die vorhergesagten Bewertungen des NCF-Models verglichen werden. Dabei werden die Distanzen zwischen den Bewertungen berechnet.

Ergebnis und Interpretation In Abbildung 7.3 ist ein Histogramm dargestellt, welches veranschaulicht, wie oft welche Bewertung vorkam. Hier sind zwei Histogramme zu sehen, wobei das obere das Histogramm für die Abweichungen der vorhergesagten Bewertungen und das untere die Abweichungen der manuellen Bewertungen darstellt. Somit wird gezeigt, dass die Benutzer, welche ähnlich sind, manuell exakt die selbe Bewertung abgegeben haben. Aus dem oberen Histogramm kann entnommen werden, dass deutlich mehr Bewertungen innerhalb der Toleranzen von 0,05 und 0,1 liegen. Die Maximale Abweichung liegt bei 0,19, weshalb bei einem Toleranzwert von 0,2 alle Bewertungen korrekt bewertet wären. Die durchschnittliche Abweichung beträgt hierbei 0,035.

In Abbildung 7.4 wird dargestellt, wie viele vorhergesagte Bewertungen zwischen ähnlichen Benutzern innerhalb der Toleranzwerte liegen. Hierfür wurde für jeden Toleranzwert eine Grafik erstellt. Dabei ist zu erkennen, dass ca. 97 % aller vorhergesagten Bewertungen zwischen ähnlichen Benutzern innerhalb der Toleranz von 10 % liegen. Bei 5 % Toleranz sind ca. 75 % aller vorhergesagten Bewertungen korrekt. Bei 20 % Toleranz sind alle vorgesagten Bewertungen korrekt.

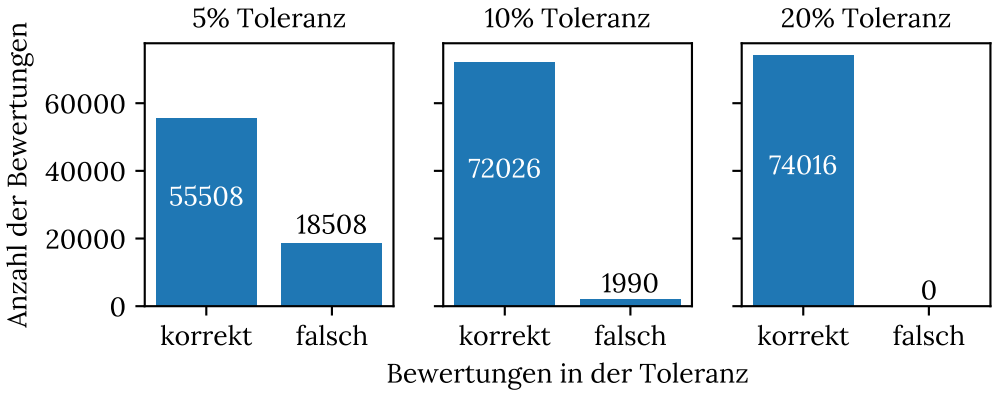


Abbildung 7.4 – Anzahl der korrekten und falschen Bewertungen

Aus diesen Erkenntnissen, kann geschlossen werden, dass für ähnliche Benutzer auch ähnliche Bewertungen zu einer Interaktion vorhergesagt werden. Demzufolge kann das Ziel dieser Anforderung als bestanden betrachtet werden.

7.2.4 Test der Anordnung

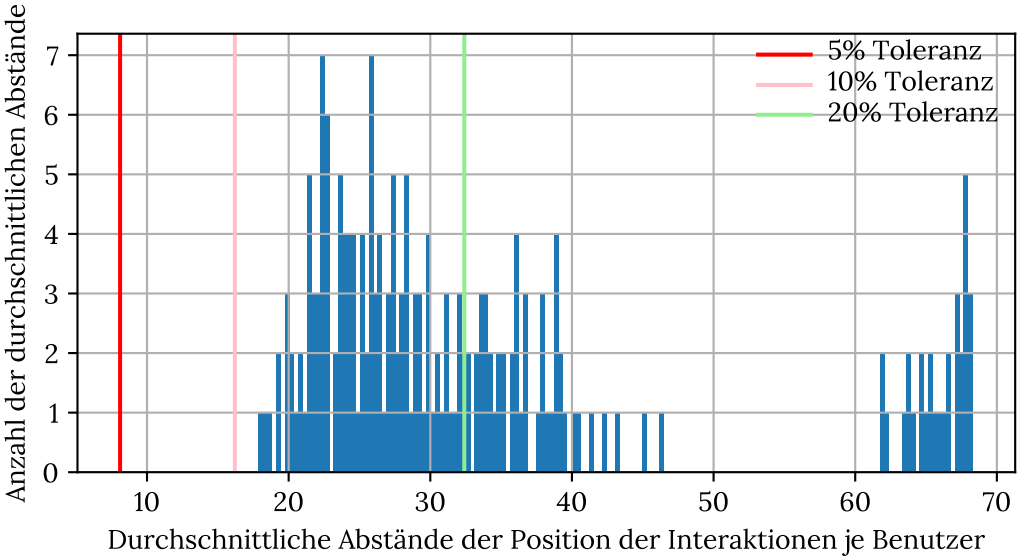


Abbildung 7.5 – Histogramm der durchschnittlichen Abstände der Positionen der Interaktion je Benutzer

Um die Anforderung **Anordnung** zu testen, sollen zwei sortierte Listen miteinander verglichen

werden. Die erste Liste enthält alle Interaktionen, sortiert nach den manuellen Bewertungen. Diese Liste wird zum Überprüfen genutzt, da angenommen wird, dass der Algorithmus zum Erstellen der Testdaten aus Abschnitt 5.2 optimale Bewertungen zu einer Interaktion abgibt.

Die zweite Liste, enthält alle Interaktionen sortiert nach den vorhergesagten Bewertungen. Diese Liste soll der Prüfliste ähneln.

Um eine Ähnlichkeit zwischen diesen Listen zu bestimmen, wird ein durchschnittlicher Abstand berechnet. Hierfür wird für jede Interaktion die Position in den beiden Listen bestimmt. Mit den Positionen kann der Abstand berechnet werden, indem der absolute Wert der Differenz beider Position bestimmt wird.

Ergebnis und Interpretation Abbildung 7.5 zeigt ein Histogramm, welches beschreibt, wie oft ein durchschnittlicher Abstand vorkommt. Hierbei kann man entnehmen, dass der durchschnittliche Abstand ca. 34,2 beträgt. Der minimale durchschnittliche Abstand liegt bei 17,89, während die maximale durchschnittliche Abstand bei 68,28 liegt. Es ist zu erkennen, dass die meisten durchschnittlichen Abstände zwischen 20 und 40 liegen.

Es ist zu beachten, dass der maximale Abstand 162 erreichen kann. Dies kann vorkommen, wenn eine Interaktion beispielsweise als die beste Interaktion manuell bewertet wurde, allerdings vom Modell als die schlechteste Interaktion bewertet wurde oder vice versa. Demzufolge ist die Interaktion jeweils auf dem ersten Platz und auf dem letzten Platz. Dieser Abstand beträgt dann 162, da es 162 Interaktionen gibt, die bewertet werden können.

In der Abbildung 7.5 wurden die Toleranzwerte gekennzeichnet. Bei einer Toleranz von 5 % liegt der Toleranzwert des Abstands, bei einem maximalen Abstand von 162 Interaktionen, bei 8,1. Für eine Toleranz von 10 % liegt der Toleranzwert bei 16,2 und bei einer Toleranz von 20 % liegt der Toleranzwert bei 32,4.

In Abbildung 7.6 werden die korrekten und falschen Anordnungen je nach Toleranz gezeigt. Demzufolge ist erneut abzuschätzen, welche Toleranz akzeptabel ist.

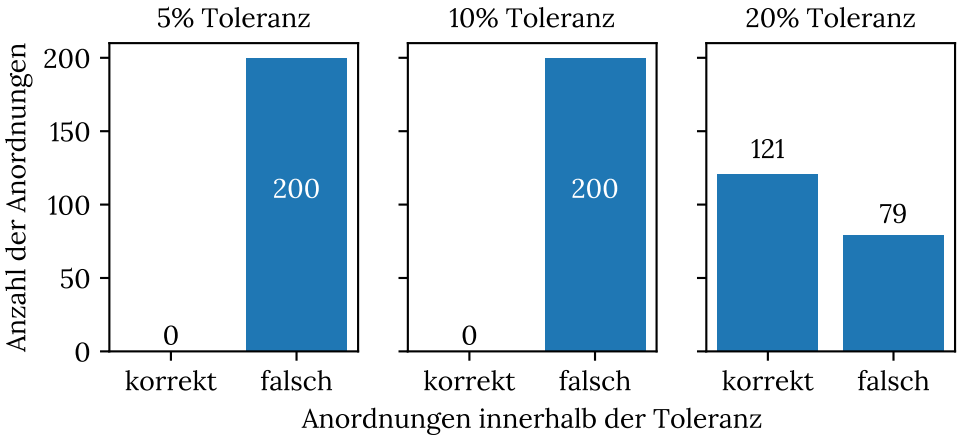


Abbildung 7.6 – Anzahl der korrekten und falschen Anordnungen

7.3 Zusammenfassung

In diesem Kapitel wurden Tests geplant, durchgeführt und ausgewertet, welche die erstellten Anforderungen aus Abschnitt 4.3 überprüfen. Hierfür wurden zunächst für jede Anforderungen Ziele erstellt, welche erreicht werden müssen, damit eine Anforderung erfüllt ist. Es wurden mehrere Toleranzen definiert, welche bei den Zielen der Anforderungen **Genauigkeit**, **Ähnlichkeit** sowie **Anordnung** verwendet werden. Die Toleranzen sind 5 %, 10 % sowie 20 %

Weiterhin ist beschrieben, wie diese Tests durchgeführt worden sind. Die Ergebnisse wurden visuell dargestellt und interpretiert. Dabei konnten die Anforderungen **Gültigkeit** und **Ähnlichkeit** die Ziele erreichen. Die Ziele der Anforderungen **Genauigkeit** und **Anordnung** können je nach Toleranz erreicht werden. Es ist zu erkennen, dass abgewogen werden muss, welche Toleranz akzeptabel ist und welche nicht.

8 Schlussfolgerung

In diesem Kapitel wird die gesamte Arbeit kurz zusammengefasst. Weiterhin soll die Arbeit nochmals kritisch betrachtet werden. Dabei zeigt der Disskussionsteil eventuelle Einschränkungen auf, wie etwas bearbeitet ist. Anschließend wird eine Auswertung stattfinden, welche das Ergebnis dieser Diplomarbeit zeigt. Zum Schluss findet ein Ausblick statt, welcher Ideen zusammenfasst, wie diese Arbeit fortgesetzt werden kann.

8.1 Zusammenfassung

Da sich die Weltbevölkerung im ständigem Wachstum befindet und demzufolge auch das Alter der Weltbevölkerung immer steigt, wird die Belastung der Pflegekräfte immer höher. Allerdings wird es in Deutschland nicht genügend Ausbildungsplätze für Pflegefachkräfte geben, um später pflegebedürftige Menschen zu pflegen. Eine Abhilfe sollen hier Roboter schaffen. Damit solche Roboter eingesetzt werden können, muss die Interaktion adaptiv gestaltet werden um eine Barrierefreiheit zu schaffen.

Da eine Vielzahl an Robotern existiert, klassifiziert Heerink et al. [Hee+10] diese. Unter anderem ist eine Klassifikation die sogenannten SAR, welche soziale Interaktionen beherrschen und assistiv den Benutzern zur Seite stehen sollen.

SAR werden im Gesundheitswesen schon eingesetzt. Hierbei können diese Roboter physische oder psychische Aufgaben erfüllen. Unter den physischen Aufgaben finden sich Tätigkeiten, wie das Aufhelfen einer Person aus dem Sitz oder dem Liegen, das Helfen beim Anziehen oder das Abholen bzw. Transportieren von Objekten. Außerdem zählen darunter auch Aufgaben wie das Überwachen von Vitalzeichen und Emotionen von Personen. Unter den psychischen Aufgaben werden Aufgaben verstanden, wie das Beruhigen von kognitiven Erkrankungen.

Damit SAR Interaktionen an den Benutzer adaptiv anpassen können, soll DL genutzt werden. DL ist ein Begriff aus der Welt der KI. Hierbei ist das Ziel, dass ein System selbst sinnvolle Beziehungen und Muster anhand von Beispielen und Beobachtungen lernen und erkennen soll. Es gibt drei Arten, wie KI-Systeme lernen: überwachtes Lernen, unüberwachtes Lernen und Verstärkungslernen. Beim überwachten Lernen sind Ein- und Ausgabedaten vorhanden. Mit diesen sollen offene Parameter kalibriert werden. Beim unüberwachten Lernen gibt es nur Eingabedaten, woraus das System interessante Strukturinformationen finden soll. Beim Verstärkungslernen werden neben den Ein- und Ausgabedaten auch der aktuelle Zustand des Systems, ein zu erreichendes Ziel, eine Liste zulässiger Aktionen sowie Umgebungsbedingungen bereitgestellt, sodass das System durch das sogenannte ”trial and error”Prinzip das Ziel erreicht.

DL sind neuronale Netze inspiriert vom Prinzip der Informationsverarbeitung in biologischen Systemen. Solche Netze bestehen aus Neuronen, welche Signale verarbeiten und das Signal verstärken oder abschwächen. Hierbei sind die Neuronen in Ebenen aufgeteilt. Dabei gibt es eine Eingabeebene, ein oder mehrerer versteckte Ebenen und eine Ausgabeebene.

Neben der KI wird Softwarevariabilität verwendet, um variable Interaktionen erstellen zu können. Softwarevariabilität zeichnet eng verwandte Softwaresysteme aus, welche auch Softwarefamilien genannt werden. Dabei bestehen Softwarefamilien aus Gemeinsamkeiten und Variabilitäten. Gemeinsamkeiten sind Objekte, die in jeder Software in einer Softwarefamilie enthalten sind, während Variabilitäten nur in einer Teilmenge vorkommen können. Durch ein Konfigurationswissen, welche alle Regeln enthält, um Kombinationen von variablen und gemeinsamen Artefakten zu validieren, und der Konzeptionsebene, welche wirtschaftliche und logische Bedenken, sowie Einschränkungen der zugrunde liegenden Technologie enthalten, kann ein valides Softwaresystem erstellt werden.

Da die unterschiedlichsten Arten der Interatkion existieren, muss zunächst gezeigt werden, welche Einschränkungen die ältere Bevölkerung hat und wie eine Interaktion mit dieser aussehen kann. Die Einschränkungen umfassen hierbei das Hören und die Kognition, welche Kurzzeitgedächtnis und Sprachverständnis mit einbezieht. Hausstein [Hau20] hat zu diesen Einschränkungen Kommunikationsstrategien für die Dialogkriterien vorgeschlagen. Mithilfe dieser Strategien wird ein Variabilitätsmodell für die Interaktion erstellt. Hierbei handelt es sich um ein sogenanntes Feature-Modell. Dieses beinhaltet die Feature *Wartezeit*, *Lautstärke*, *Geschwindigkeit*, *Einfache Sprache* sowie *Tonhöhe*. Mit diesem Feature-Modell könnnen nun Interaktionen erstellt werden.

Um eine der validen Konfiguration einer Interaktion zu wählen, muss ein DL-Modell ausgesucht werden. Dafür werden zunächst DL-Modelle vorgestellt und kurz erläutert. Darunter befindet sich ein MLP, CNN, RNN und ein NCF. Beim Vergleich der Modelle ist zu erkennen, dass alle Modelle zum Lerntyp des überwachten Lernens gehören. Bis auf RNN sind alles *feed forward* Netzwerkverkettungen. Nach Betrachtung der Anwendungsfälle konnte durch Ausschlussverfahren CNN und RNN entfernt werden. MLP wird für Regressionen und Klassifikationen genutzt, während NCF für Empfehlungssysteme genutzt wird. Hier entsteht die Idee, dass Benutzer, welche Ähnlichkeiten aufweisen, auch die gleichen Interaktionen nutzen würden. Demzufolge wird ein NCF gewählt, da dies für Empfehlungssysteme genutzt wird, und andere Benutzer mit in Betracht gezogen werden können.

Um ein Konzept zu erarbeiten, welches getestet werden kann, müssen Anforderungen erstellt werden. Dabei wurden die Anforderungen **Gültigkeit**, **Genauigkeit**, **Ähnlichkeit** und **Anordnung** definiert. Die Anforderung **Gültigkeit** beschreibt, dass nur für valide Konfigurationen einer Interaktion Bewertungen vorhergesagt werden sollen. Die Anforderung **Genauigkeit** beschreibt, dass das DL-Modell die Vorhersagen der Bewertungen so treffen soll, wie es eine Person auch selbst bewerten würde. Die Anforderung **Ähnlichkeit** beschreibt, dass Personen die ähnlich sind, für die selbe Interaktion auch eine ähnliche Bewertung treffen und das DL-System ebenfalls ähnliche Bewertungen vorhersagen soll. Die letzte Anforderung **Anordnung** beschreibt, dass für Personen mit bestimmten Eigenschaften Interaktionen mit passenden Features eine höhere Bewertung bekommen, als Interaktionen mit weniger passenden Features.

Mit diesen Anforderungen kann nun ein Konzept erstellt werden. Hierbei wird gezeigt, wo dieses Konzept seinen Einsatz findet. Damit das Konzept als ein DL-Modell implementiert werden kann, müssen zunächst die Eingabedaten definiert werden. Dabei gibt es Eingabedaten der Nutzer, welche die Eigenschaften der Nutzer darstellt. Darunter zählen die Eigenschaften: *Alter*, *Geschlecht*, *Stadium der* ASH und *kognitive Einschränkung*. Zu den Eingabedaten der Interaktionen werden die Feature aus dem Feature-Modell entnommen. Demzufolge gibt es die folgenden Feature: *einfache Sprache*, *Länge der Wartezeit*, *Lautstärke*, *Geschwindigkeit* sowie die *Tonhöhe*. Um ein Modell trainieren zu können, müssen auch Ausgabedaten definiert werden. Ziel ist es für ei-

nen Benutzer zu einer Interaktion eine Bewertung vorhersagen zu lassen. Demzufolge sind die Ausgabedaten nur auf die Bewertung beschränkt.

Um ein DL-Modell zu trainieren, werden Traingsdaten benötigt. Diese Trainingsdaten können durch Umfragen, Interviews mit Experten oder aus Studien generiert werden. Mit den Studien und Statistiken von Heger und Hubole [HH10], Deutsche Alzheimer Gesellschaft e. V.[Deu22] sowie vom Statistischen Bundesamt [Sta22] konnte ein Algorithmus entwickelt werden, welcher Testpersonen statistisch erstellt und diese zu Interaktionen bewerten lassen kann.

Da das NCF-Modell auf CF basiert, wurde dies kurz erläutert und die Grenzen gezeigt. He et al.[He+17] zeigt, wie diese Grenzen mit Hilfe von DL überwunden werden können. Dieses NCF-Modell wurde genommen und weiter ausgebaut, sodass die Features der Interaktionen und die Eigenschaften der Personen mit einbezogen werden. Zum Schluss wurde dieses Modell mithilfe von TensorFlow und Keras implementiert.

Dieses Modell konnte nun in einem einfachen Prototypen implementiert werden. Der Prototyp sind einige vereinzelte Systeme, die zusammenarbeiten. Der Roboter Loomo soll mit einer Android-App die Interaktion durchführen. Dieser sendet dabei die Eigenschaften der Person zu einer REST-Schnittstelle, welche das NCF-Modell implementiert hat. Weiterhin sendet die App die gesprochenen Informationen zu dem Mycroft.ai-Skill weiter. Der Mycroft.ai-Skill dient hierbei nur als Brücke zwischen der Android-App und RASA. RASA verarbeitet die Informationen und ist für den Dialog zuständig. Dieser holt sich auch die Information von der REST-Schnittstelle, welche Interaktion durchgeführt wird, damit der Skill eine einfache Sprache oder die normale Sprache verwenden kann. Die Antwort wird von RASA zurück zur Android-App über den Mycroft.ai-Skill gesendet. Die Android-App holt sich ebenfalls die Informationen von der REST-Schnittstelle, um die Interaktion weiter anzupassen.

Nach der Implementierung des Prototypen können nun die Anforderungen getestet werden. Definierte Ziele für die genannten Anforderungen beschreiben, wann eine Anforderung erfüllt ist. Für jede Anforderung wurden Tests geplant und durchgeführt. Die Ergebnisse sind in Grafiken visualisiert und interpretiert. Die Tests zeigen, dass zwei von vier Anforderungen bestanden sind: **Gültigkeit**, **Ähnlichkeit**. Ob die Anforderungen **Genauigkeit** und **Anordnung** bestanden sind, hängt von der Definition der Toleranz ab.

8.2 Diskussion

Das erstellte Modell ist fähig, für einen Benutzer anhand seiner Eigenschaften Interaktionen mit unterschiedlichen Features zu bewerten. Hierbei wird CF benutzt, wodurch Benutzer die ähnlich sind auch die selben Interaktionen ähnlich bewerten würden. Hierbei gibt es jedoch Grenzen.

In den Tests aus Kapitel 7 ist zu erkennen, dass bei zwei Anforderungen noch die Toleranz definiert werden muss, ab wann welche Bewertung akzeptabel ist. Wenn eine Toleranz von 5 % oder 10 % genutzt wird, sind die Anforderungen **Genauigkeit** und **Anordnung** nicht bestanden. Wenn jedoch eine Toleranz von 20 % genutzt wird, sind diese Anforderungen bestanden. Hier ist zu entscheiden, welche Toleranz im akzeptablen Bereich liegt.

Weiterhin wurden für das Trainieren des Modells und der Tests des Modells generierte Daten verwendet. Hierbei wurden Studien und Statistiken genutzt, um eine Verteilung des Alters sowie des Geschlechts natürlich erscheinen zu lassen. Welche möglichen Einschränkung eine Person hat, wurden hier ebenfalls aus Studien entnommen und an Alter und Geschlecht einer Person natürlich verteilt. Um Bewertungen generieren zu können, wurden ebenfalls Studien benutzt. Da-

bei wurden jedoch auch Vereinfachungen und Annahmen getroffen. Da sich diese Arbeit nur auf sprachliche Interaktion konzentriert und ein hoher Schweregrad der ASH in der Realität durch schriftlichen Text kompensiert wird, wurde hier die Annahme getroffen, dass Lautstärke ebenfalls höher sein muss. Demzufolge gibt es hier die Einschränkung der generierten Daten. Es sollten hier die Daten auf Korrektheit überprüft werden oder Studien durchgeführt werden, um echte Daten zum Trainieren und Testen zu erhalten.

Das Modell wurde außerdem immer mit der Aktivierungsfunktion Rectified Linear Unit (ReLU) erstellt. Diese Aktivierungsfunktion wurde verwendet, da diese mit am häufigsten in DL-Modellen verwendet wird. Hier wäre zu betrachten, ob das Modell bessere Ergebnisse liefert, wenn eine andere Aktivierungsfunktion wie *Sigmoid* oder *tanh* verwendet wird.

In den Tests aus Kapitel 7 ist ebenfalls zu erkennen, dass die Anforderung **Genauigkeit** im Vergleich zum Algorithmus, welcher die Bewertungen generiert hat, einige Abweichungen hat. Hier wäre es interessant zu betrachten, welche Unterschiede ein anderes Modell machen würde. Bei der Wahl eines DL-Modells standen zwei Modelle nach dem Ausschlussverfahren noch zur Auswahl. Das NCF wurde hierbei mit der Idee gewählt, dass ähnliche Nutzer die selben Interaktionen auch ähnlich bewerten würden. Bei der Betrachtung des erstellten Modells aus Abschnitt 4.2, lässt sich erkennen, dass die Matrixfaktorisierung, sowie mehrere MLPs zum Schluss zu einem Wert zusammengefasst werden. Hier könnte eventuell eine Gewichtung eine Rolle spielen. Es muss geprüft werden, welche Gewichtung zu welchen MLP sowie zur CF genutzt werden kann, um bessere Ergebnisse zu erhalten.

Ein weiterer Faktor, welcher das Modell eingrenzt, ist die Nutzung von vollständig konfigurierten Interaktionen. Es gibt eine feste Anzahl von Interaktionen und es ist schwer, diese zu erweitern, da definiert ist, welche Features eine Interaktion haben kann. Um dies zu verbessern, könnten die Features der Interaktionen bewertet werden um dadurch eine valide Konfiguration einer Interaktion erstellen zu lassen. Somit gibt es die Möglichkeit neue Features hinzuzufügen.

Bei der Verwendung eines NCF-Modells gehören Feedbacks dazu, welche das Modell weiter trainieren. So kann das System weiter lernen, welche Interaktion eine Person bevorzugt. In dieser Arbeit wurden Feedbacks nicht weiter betrachtet. Außerdem beschäftigt sich diese Arbeit nicht damit, woher das System die Informationen von dem Benutzer oder von den Interaktionen bekommt bzw. beschafft.

8.3 Ergebnis

In diesem Abschnitt werden zunächst die Teilziele aufgegriffen und deren Ergebnis beschrieben. Im Abschluss wird die wissenschaftliche Frage aus Kapitel 1 beantwortet.

Erfassung von sprachlichen Interaktionsmöglichkeiten bei der älteren Bevölkerung Um dieses Teilziel zu erfüllen, wurden Einschränkungen der älteren Bevölkerung kurz erläutert. Mithilfe der Kommunikationsstrategien von Haustein [Hau20] konnte ein Feature-Modell erstellt werden. Durch dieses Feature-Modell ist es möglich, valide Konfigurationen einer Interaktion zu erstellen, welche bei der sprachlichen Interaktion bei der älteren Bevölkerung genutzt werden können.

Analyse eines DL-Modells Hierfür wurden mehrere DL-Modelle aufgezählt und kurz erläutert. Diese Modelle konnten miteinander Vergleichen werden. Mit einem Ausschlussverfahren konnten

nicht passende Modelle ausgeschlossen werden. Zum Schluss standen die Modell MLP und NCF zur Auswahl. Es wurde sich für das NCF entschieden, da hier CF verwendet wird, wodurch die Ähnlichkeit von Personen ebenfalls betrachtet werden kann.

Erstellung des Konzepts Um ein Konzept zu entwickeln, wurden zunächst die Eingabe- und Ausgabedaten definiert und beschrieben. Das DL-Modell NCF wurde erweitert, um die Features der Interaktionen und die Eigenschaften der Personen mit zu betrachten. Um das Modell trainieren zu können, wurden Trainings- und Testdaten mit Hilfe von Studien und Statistiken generiert. Zum Schluss wurde das Modell mit TensorFlow und Keras in der Programmiersprache Python umgesetzt.

Implementierung des Prototypen Das Modell wurde für den Prototypen nochmals erweitert. Dabei wurden mehrere Dense-Layer in den MLPs verwendet, statt nur einem Dense-Layer. Dieses Modell ist in eine REST-Schnittstelle implementiert worden, sodass andere Services wie RASA oder die Android-App die Bewertungen von Interaktionen bei der REST-Schnittstelle abrufen können, um die passende Interaktion mit einem Benutzer durchzuführen.

Tests des Prototypen Um den Prototypen zu testen, wurden Anforderungen erstellt, welche der Prototyp erfüllen sollte. Mithilfe der generierten Testdaten konnten diese Tests durchgeführt werden. Hierbei sind Ziele für die Anforderung definiert worden. Für jede Anforderung führte man Tests durch, dessen Ergebnisse visuell festgehalten wurden.

Wissenschaftlich Frage Die wissenschaftliche Frage **"Welche Schritte sind nötig, damit ein Assistenzroboter selbst lernt eine optimale Interaktion für einen Benutzer zu wählen, welche die Einschränkungen des Benutzers berücksichtigt?"** wird wie folgt beantwortet: Es ist nötig zu wissen, welche Einschränkungen ein Benutzer haben kann. Mit dem Wissen über die Einschränkungen kann analysiert werden, welche Interaktion mit dem Nutzer durchgeführt werden kann. Hierbei ist ein Feature-Modell entstanden, welches alle validen Konfigurationen einer sprachlichen Interaktion mit einer älteren Person darstellt. Dieses Feature-Modell unterteilt sich dabei zwischen einer Eingabe und einer Ausgabe. Für die Ausgabe gibt es das Feature **Wartezeit**, welche kurz, mittel oder lang sein kann. Es ist nicht definiert, was kurz, mittel oder lang hierbei bedeutet. Analog dazu, können die Features in der Ausgabe **Lautstärke** den Wert leise, normal oder laut, **Geschwindigkeit** die Werte langsam, normal, schnell und die **Tonhöhe** die Werte tief, normal oder hoch annehmen. Das Feature **einfache Sprache** kann aktiviert oder nicht aktiviert sein. Mit diesem Feature-Modell konnten 162 Interaktionen erstellt werden, welche später vom DL-Modell bewertet werden sollen. Weiterhin muss eine KI erstellt werden, welche durch ein DL-Modell lernen kann, welche Interaktionen zu welchen Einschränkungen passen. Dafür wurde ein NCF-Modell genutzt und erweitert, damit dieses nicht nur kollaboratives Filtern durchführt, sondern auch die Features der Interaktionen und die Eigenschaften der Personen zur Bewertung mit einbezieht. Um dieses Modell trainieren zu können, werden Trainingsdaten benötigt. Diese sind aus Studien und Statistiken zusammengetragen. Mit einem erstellten Algorithmus konnten somit Benutzer erstellt werden, welche Interaktionen bewertet haben. Im Anschluss kann dieses Modell in einem Assistenzroboter als Service implementiert werden.

8.4 Ausblick

In Abschnitt 8.2 wurden die Grenzen des erarbeiteten Konzepts genannt und mögliche Lösungsansätze aufgezählt.

Diese Arbeit bietet eine Grundlage für ein Konzept mittels DL für eine Person mit bestimmten Einschränkungen und Eigenschaften eine Interaktion zu wählen. Hierbei könnte diese Arbeit vervollständigt werden, wenn dieses Modell mit echten erhobenen Daten trainiert wird.

Weiterhin wurde in Abschnitt 8.2 genannt, dass fest konfigurierte Interaktionen zur Bewertung genutzt werden. Dieses Modell könnte erweitert werden, indem statt fest konfigurierte Interaktionen nur die Features der Interaktion bewertet werden. Dies sollte mehr Variabilität und eine einfache Erweiterbarkeit bieten. Mit den bewerteten Features kann durch einen Konfigurationsrealisierungsmechanismus eine Interaktion erstellt werden.

Außerdem ist in Abschnitt 8.2 beschrieben, dass nicht definiert ist, welche Toleranz akzeptabel ist. Hier kann überprüft werden, welche Toleranzen für ein NCF-Modell akzeptabel sind.

Doch um dieses Modell in der Realität benutzen zu können, ist es wichtig zu wissen, woher das System die Informationen über den Benutzer oder den Interaktionen beschaffen kann. Demzufolge kann weiter geforscht werden, wie das System automatisiert die Daten eines Benutzers erfasst, um somit die Einschränkungen und Eigenschaften sowie Feedback an das DL-Modell weiter zu geben.

Es ist zu erkennen, dass noch viele Probleme gelöst werden müssen, damit ein SAR einem Benutzer im Haushalt und bei Aktivitäten unterstützen kann. Die Forschung ist hier auf einem guten Weg und es werden immer mehr Probleme gelöst oder bereits existente Lösungen verbessert. Diese Arbeit soll ebenfalls als eine Grundlage für einen Lösungsansatz dienen.

A Anhang

In diesem Kapitel werden Tabellen oder Textpassagen, welche nicht in den Fließtext der Diplomarbeit passen oder zu groß sind, präsentiert.

A.1 Verwendung des Prototypen

In diesem Kapitel wird erklärt, wie der Prototyp anzuwenden ist, welche Abhängigkeiten gebraucht werden und wie diese installiert werden können. Außerdem wird gezeigt, welche Konfigurationen angepasst werden müssen.

Der Prototyp besteht aus mehreren Programmen: die REST-API, die Android-App, der Mycroft.ai-Skill und dem RASA-Bot.

A.1.1 Android-App

Die Android-App liegt fertiggebaut als `apk`-Datei bereit und kann auf sämtlichen Android-Betriebssystemen ab der Version 5.1 (Lollipop) installiert werden. Die minimal unterstützte API ist Version 22.

Wenn nicht die fertiggebaute App genutzt werden soll, muss diese mithilfe von Android Studio gebaut werden. Hierfür öffnet man das Projekt **Mycroft.ai-Android** in Android Studio. Sollte Android Studio verlangen, dass Gradle geupdatet werden soll, ist dies zu unterbinden. Wenn eine neue Version von Gradle eingespielt wird, kann es möglich sein, dass Anpassungen am Projekt notwendig sind.

Sobald das Projekt geladen ist, kann unter **Build** -> **Build Bundle(s) / APK(s)** -> **Build APK(s)** die App gebaut werden. Je nach dem, ob als Bauvariante **debug** oder **release** ist die App wie folgt zu finden:

- **debug**: `mobile/build/apk/debug/mobile-debug.apk`
- **release**: `mobile/build/apk/release/mobile-release-unsigned.apk`

Diese kann auf einem Android-Gerät installiert und ausgeführt werden. Wichtig ist dabei zu beachten, dass in den Einstellungen von Android, das Installieren von Drittanbieter-Software aktiviert ist.

Sobald diese App gestartet ist, müssen die IP-Adressen der REST-API, sowie der Mycroft.ai-Skill angegeben werden. Bevor die App ausgeführt werden kann, sollten die REST-API sowie die Mycorft.ai-Skill bereit stehen.

A.1.2 REST-API

Für die REST-API wird Python in der Version 3.9 benötigt. Hierfür sollte zunächst eine virtuelle Umgebung für Python eingerichtet werden. In Quelltext A.1 werden alle benötigten Befehle an-

gezeigt. In Zeile 2 wird eine virtuelle Umgebung für Python angelegt. Zeile 4 zeigt, wie so eine virtuelle Umgebung für Python auf einem Linux Betriebssystem gestartet werden kann.

Als nächstes müssen alle benötigten Bibliotheken installiert werden. Dies kann mit dem Befehl auf Zeile 6 erledigt werden. Sollten noch keine Testdaten vorhanden sein, müssen mit dem Befehl auf Zeile 8 diese erstellt werden. In `create_data.py` befindet sich der erstellte Algorithmus aus dem Abschnitt 5.4.

Wenn noch keine Datenbank vorhanden ist, muss mit dem Befehl auf Zeile 10 die Datenbank erstellt werden. Diese befindet sich unter `instances/test.db`. Dabei wird das DL-Modell aktiviert. Sollte kein vortrainiertes Modell existieren, trainiert dieses Modell erst. Das kann einige Zeit in Anspruch nehmen.

Um die *REST-API* zu starten, muss der Befehl auf Zeile 12 ausgeführt werden. Die *REST-API* sollte dann auf dem Port 5000 verfügbar sein.

```
1 # create virtual environtment
2 python3.9 -m venv venv
3 # activate virtual environment (only linux)
4 source venv/bin/activate
5 # install requirements
6 pip install -r requirements.txt
7 # create test-data
8 python3 create_data.py
9 # create database
10 python3 -c "from main import create_db; create_db()"
11 # run REST-API
12 python3 main.py
```

Quelltext A.1 – Befehle um die *REST-API* zu starten

A.1.3 Mycroft.ai-Skill

Um den Mycroft.ai-Skill zu verwenden, muss Mycroft.ai wie in deren Anleitung[1] zunächst installiert werden.

Der Mycroft.ai-Skill muss dann in folgenden Ordner verschoben werden:

`<Pfad zu MYCROFT_HOME>/skills`

Dann sollte Mycroft.ai den Skill automatisch laden.

Doch bevor der Skill richtig verwendet werden kann, müssen die IP-Adressen der *REST-API* und dem RASA-Bot in der Datei `__init__.py` angepasst werden.

A.1.4 RASA-Bot

Für den RASA-Bot ist Python in der Version 3.9 notwendig. Es sollte wieder eine virtuelle Umgebung für Python erstellt werden. Die Befehle, um die virtuelle Umgebung zu erstellen und zu aktivieren, werden in Zeile 2 und 4 im Quelltext A.2 gezeigt.

In Zeile 6 wird RASA mit allen Abhängigkeiten in der virtuellen Umgebung installiert.

Sollte für den RASA-Bot noch kein trainiertes Modell vorhanden sein, kann mit dem Befehl auf Zeile 8 ein Modell erstellt werden. Sonst kann der RASA-Action-Server mit dem Befehl auf Zeile

[1] `https://mycroft-ai.gitbook.io/docs/using-mycroft-ai/get-mycroft`

10 gestartet werden. Bevor der RASA-Bot gestartet wird, muss unter `actions/actions.py` die IP-Adresse der *REST-API* konfiguriert werden. Danach kann der RASA-Bot selber mit dem Befehl auf Zeile 12 gestartet werden. Die Restschnittstelle von RASA hört hier auf den Port 8000.

```
1 # create virtual environtment
2 python3.9 -m venv venv
3 # activate virtual environment (only linux)
4 source venv/bin/activate
5 # install rasa
6 pip install rasa
7 # train rasa
8 rasa train
9 # run action server
10 rasa run actions
11 # run RASA-Bot
12 rasa run -p 8000
```

Quelltext A.2 – Befehle um die *REST-API* zu starten

A.2 Test des Konzepts

In diesem Abschnitt werden Tabellen für die Tests des Konzepts gelistet, welche für den Fließtext der Diplomarbeit deutlich zu groß sind .

Tabelle A.1 – Tabelle aller Benutzer mit Status der Sortierten Liste der Interaktionen

Benutzer ID	Geschlecht	Alter	Grad der ASH	kognitive Erkrankung	Liste ist valide
800	f	66	1	0	ok
801	f	71	2	0	ok
802	f	64	0	0	ok
803	m	58	0	0	ok
804	f	82	3	0	ok
805	m	70	0	0	ok
806	f	64	0	0	ok
807	m	82	0	0	ok
808	f	58	1	0	ok
809	f	60	0	0	ok
810	m	58	0	0	ok
811	m	79	1	0	ok
812	m	57	0	0	ok
813	f	85	1	0	ok
814	f	85	0	1	ok
815	f	60	0	0	ok
816	m	67	1	0	ok
817	m	60	0	0	ok
818	m	78	1	0	ok
819	m	77	0	1	ok

Tabelle A.1 – Tabelle aller Benutzer mit Status der Sortierten Liste der Interaktionen

Benutzer ID	Geschlecht	Alter	Grad der ASH	kognitive Erkrankung	Liste ist valide
820	f	72	0	0	ok
821	m	68	0	0	ok
822	f	57	0	0	ok
823	m	58	0	0	ok
824	f	71	0	0	ok
825	f	73	0	1	ok
826	m	61	0	0	ok
827	f	59	0	0	ok
828	f	75	2	1	ok
829	f	85	1	0	ok
830	f	73	0	1	ok
831	m	72	2	0	ok
832	m	78	1	0	ok
833	f	63	0	0	ok
834	m	56	1	0	ok
835	f	71	1	1	ok
836	f	73	1	1	ok
837	m	68	0	0	ok
838	m	71	0	0	ok
839	f	61	0	0	ok
840	f	63	0	0	ok
841	m	60	0	0	ok
842	f	55	0	0	ok
843	m	85	0	0	ok
844	f	82	1	1	ok
845	f	58	0	0	ok
846	m	62	0	0	ok
847	f	79	2	0	ok
848	f	64	1	0	ok
849	f	56	1	0	ok
850	f	61	0	0	ok
851	m	69	0	0	ok
852	f	57	0	0	ok
853	f	85	2	0	ok
854	m	78	1	0	ok
855	f	55	0	0	ok
856	m	69	0	0	ok
857	f	75	0	0	ok
858	m	56	0	0	ok
859	f	61	0	0	ok
860	m	69	0	0	ok

Tabelle A.1 – Tabelle aller Benutzer mit Status der Sortierten Liste der Interaktionen

Benutzer ID	Geschlecht	Alter	Grad der ASH	kognitive Erkrankung	Liste ist valide
861	f	55	0	0	ok
862	m	76	1	1	ok
863	m	55	0	0	ok
864	f	58	0	0	ok
865	f	57	0	0	ok
866	m	84	0	1	ok
867	f	55	0	0	ok
868	f	85	1	1	ok
869	m	67	0	0	ok
870	m	57	1	0	ok
871	m	81	1	1	ok
872	f	67	0	0	ok
873	f	79	0	1	ok
874	f	85	1	0	ok
875	m	70	0	0	ok
876	f	66	0	0	ok
877	f	60	0	0	ok
878	m	85	1	0	ok
879	f	55	0	0	ok
880	m	72	0	0	ok
881	f	77	1	1	ok
882	f	56	0	0	ok
883	f	67	1	0	ok
884	m	76	0	0	ok
885	m	58	0	0	ok
886	m	64	0	0	ok
887	m	85	0	0	ok
888	f	72	0	1	ok
889	f	57	0	0	ok
890	f	83	1	1	ok
891	m	81	1	0	ok
892	f	67	0	0	ok
893	m	65	1	0	ok
894	f	63	0	0	ok
895	f	81	1	1	ok
896	f	58	0	0	ok
897	f	66	2	0	ok
898	m	85	3	0	ok
899	m	76	1	0	ok
900	f	68	0	0	ok
901	m	59	2	0	ok

Tabelle A.1 – Tabelle aller Benutzer mit Status der Sortierten Liste der Interaktionen

Benutzer ID	Geschlecht	Alter	Grad der ASH	kognitive Erkrankung	Liste ist valide
902	m	61	0	0	ok
903	m	72	0	0	ok
904	f	66	0	0	ok
905	f	66	0	0	ok
906	f	63	0	0	ok
907	f	68	1	0	ok
908	m	71	0	0	ok
909	f	66	0	0	ok
910	f	70	0	0	ok
911	f	59	0	0	ok
912	f	59	0	0	ok
913	f	61	1	0	ok
914	f	85	2	1	ok
915	f	55	0	0	ok
916	f	55	1	0	ok
917	f	57	0	0	ok
918	f	58	0	0	ok
919	f	63	0	0	ok
920	f	66	0	0	ok
921	f	80	3	1	ok
922	f	85	1	1	ok
923	m	55	1	0	ok
924	f	64	0	0	ok
925	f	67	0	0	ok
926	f	59	0	0	ok
927	f	78	0	0	ok
928	m	65	0	0	ok
929	f	79	1	1	ok
930	f	58	0	0	ok
931	m	70	1	0	ok
932	f	78	0	0	ok
933	f	56	1	0	ok
934	m	55	0	0	ok
935	f	55	0	0	ok
936	f	65	0	0	ok
937	m	58	0	0	ok
938	m	73	0	1	ok
939	f	83	1	1	ok
940	f	68	1	0	ok
941	m	71	0	1	ok
942	f	57	0	0	ok

Tabelle A.1 – Tabelle aller Benutzer mit Status der Sortierten Liste der Interaktionen

Benutzer ID	Geschlecht	Alter	Grad der ASH	kognitive Erkrankung	Liste ist valide
943	f	59	0	0	ok
944	f	65	0	0	ok
945	m	58	0	0	ok
946	f	85	0	0	ok
947	f	56	0	0	ok
948	m	83	1	0	ok
949	f	81	1	0	ok
950	f	63	0	0	ok
951	f	85	0	0	ok
952	f	62	0	0	ok
953	f	61	0	0	ok
954	m	85	0	0	ok
955	f	57	0	0	ok
956	m	63	0	0	ok
957	m	63	1	0	ok
958	m	78	2	0	ok
959	f	62	0	0	ok
960	f	64	0	0	ok
961	f	55	0	0	ok
962	m	85	1	0	ok
963	m	61	0	0	ok
964	f	69	2	0	ok
965	m	57	0	0	ok
966	m	67	0	0	ok
967	m	78	1	0	ok
968	f	85	1	0	ok
969	f	68	1	0	ok
970	m	57	0	0	ok
971	m	63	0	0	ok
972	f	60	0	0	ok
973	m	73	0	1	ok
974	f	63	0	0	ok
975	f	82	1	0	ok
976	m	82	1	0	ok
977	m	56	0	0	ok
978	f	68	0	0	ok
979	f	69	0	0	ok
980	f	68	1	0	ok
981	f	75	1	1	ok
982	m	74	0	0	ok
983	m	57	0	0	ok

Tabelle A.1 – Tabelle aller Benutzer mit Status der Sortierten Liste der Interaktionen

Benutzer ID	Geschlecht	Alter	Grad der ASH	kognitive Erkrankung	Liste ist valide
984	m	69	1	0	ok
985	m	57	0	0	ok
986	f	81	0	0	ok
987	f	55	0	0	ok
988	m	60	0	0	ok
989	f	85	0	1	ok
990	f	76	3	1	ok
991	f	73	0	1	ok
992	m	62	0	0	ok
993	f	84	1	1	ok
994	f	60	0	0	ok
995	m	63	0	0	ok
996	m	62	0	0	ok
997	f	57	0	0	ok
998	m	61	0	0	ok
999	m	69	0	0	ok

Tabelle A.2 – Liste aller validen Konfiguration von Interaktionen

Interaktions ID	Wartezeit	Lautstärke	Geschwindigkeit	Tonhöhe	einfache Sprache
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	0	2	0
5	0	0	0	2	1
6	0	0	1	0	0
7	0	0	1	0	1
8	0	0	1	1	0
9	0	0	1	1	1
10	0	0	1	2	0
11	0	0	1	2	1
12	0	0	2	0	0
13	0	0	2	0	1
14	0	0	2	1	0
15	0	0	2	1	1
16	0	0	2	2	0
17	0	0	2	2	1
18	0	1	0	0	0
19	0	1	0	0	1

Tabelle A.2 – Liste aller validen Konfiguration von Interaktionen

Interaktions ID	Wartezeit	Lautstärke	Geschwindigkeit	Tonhöhe	einfache Sprache
20	0	1	0	1	0
21	0	1	0	1	1
22	0	1	0	2	0
23	0	1	0	2	1
24	0	1	1	0	0
25	0	1	1	0	1
26	0	1	1	1	0
27	0	1	1	1	1
28	0	1	1	2	0
29	0	1	1	2	1
30	0	1	2	0	0
31	0	1	2	0	1
32	0	1	2	1	0
33	0	1	2	1	1
34	0	1	2	2	0
35	0	1	2	2	1
36	0	2	0	0	0
37	0	2	0	0	1
38	0	2	0	1	0
39	0	2	0	1	1
40	0	2	0	2	0
41	0	2	0	2	1
42	0	2	1	0	0
43	0	2	1	0	1
44	0	2	1	1	0
45	0	2	1	1	1
46	0	2	1	2	0
47	0	2	1	2	1
48	0	2	2	0	0
49	0	2	2	0	1
50	0	2	2	1	0
51	0	2	2	1	1
52	0	2	2	2	0
53	0	2	2	2	1
54	1	0	0	0	0
55	1	0	0	0	1
56	1	0	0	1	0
57	1	0	0	1	1
58	1	0	0	2	0
59	1	0	0	2	1
60	1	0	1	0	0

Tabelle A.2 – Liste aller validen Konfiguration von Interaktionen

Interaktions ID	Wartezeit	Lautstärke	Geschwindigkeit	Tonhöhe	einfache Sprache
61	1	0	1	0	1
62	1	0	1	1	0
63	1	0	1	1	1
64	1	0	1	2	0
65	1	0	1	2	1
66	1	0	2	0	0
67	1	0	2	0	1
68	1	0	2	1	0
69	1	0	2	1	1
70	1	0	2	2	0
71	1	0	2	2	1
72	1	1	0	0	0
73	1	1	0	0	1
74	1	1	0	1	0
75	1	1	0	1	1
76	1	1	0	2	0
77	1	1	0	2	1
78	1	1	1	0	0
79	1	1	1	0	1
80	1	1	1	1	0
81	1	1	1	1	1
82	1	1	1	2	0
83	1	1	1	2	1
84	1	1	2	0	0
85	1	1	2	0	1
86	1	1	2	1	0
87	1	1	2	1	1
88	1	1	2	2	0
89	1	1	2	2	1
90	1	2	0	0	0
91	1	2	0	0	1
92	1	2	0	1	0
93	1	2	0	1	1
94	1	2	0	2	0
95	1	2	0	2	1
96	1	2	1	0	0
97	1	2	1	0	1
98	1	2	1	1	0
99	1	2	1	1	1
100	1	2	1	2	0
101	1	2	1	2	1

Tabelle A.2 – Liste aller validen Konfiguration von Interaktionen

Interaktions ID	Wartezeit	Lautstärke	Geschwindigkeit	Tonhöhe	einfache Sprache
102	1	2	2	0	0
103	1	2	2	0	1
104	1	2	2	1	0
105	1	2	2	1	1
106	1	2	2	2	0
107	1	2	2	2	1
108	2	0	0	0	0
109	2	0	0	0	1
110	2	0	0	1	0
111	2	0	0	1	1
112	2	0	0	2	0
113	2	0	0	2	1
114	2	0	1	0	0
115	2	0	1	0	1
116	2	0	1	1	0
117	2	0	1	1	1
118	2	0	1	2	0
119	2	0	1	2	1
120	2	0	2	0	0
121	2	0	2	0	1
122	2	0	2	1	0
123	2	0	2	1	1
124	2	0	2	2	0
125	2	0	2	2	1
126	2	1	0	0	0
127	2	1	0	0	1
128	2	1	0	1	0
129	2	1	0	1	1
130	2	1	0	2	0
131	2	1	0	2	1
132	2	1	1	0	0
133	2	1	1	0	1
134	2	1	1	1	0
135	2	1	1	1	1
136	2	1	1	2	0
137	2	1	1	2	1
138	2	1	2	0	0
139	2	1	2	0	1
140	2	1	2	1	0
141	2	1	2	1	1
142	2	1	2	2	0

Tabelle A.2 – Liste aller validen Konfiguration von Interaktionen

Interaktions ID	Wartezeit	Lautstärke	Geschwindigkeit	Tonhöhe	einfache Sprache
143	2	1	2	2	1
144	2	2	0	0	0
145	2	2	0	0	1
146	2	2	0	1	0
147	2	2	0	1	1
148	2	2	0	2	0
149	2	2	0	2	1
150	2	2	1	0	0
151	2	2	1	0	1
152	2	2	1	1	0
153	2	2	1	1	1
154	2	2	1	2	0
155	2	2	1	2	1
156	2	2	2	0	0
157	2	2	2	0	1
158	2	2	2	1	0
159	2	2	2	1	1
160	2	2	2	2	0
161	2	2	2	2	1

Abkürzungsverzeichnis

ASH	Altersschwerhörigkeit	23
CF	Colaborativ Filtering	6
CNN	Convolutional Neural Network	16
DL	Deep Learning	IV
RS	Empfehlungssystem	16
KI	Künstlichen Intelligenz	14
KNN	künstliches neuronales Netzwerk	14
ML	Machine Learning	2
MLP	Multi-Layer Perceptron	16
MF	Matrixfaktorisierung	41
NCF	Neural Collaborativ Filtering	26
RNN	Recurrent Neural Network	16
ReLU	Rectified Linear Unit	72
SAR	Sozialer Assistenzroboter	1
SV	Softwarevariabilität	11
WHO	World Health Organisation	34

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltexte

Inhalte des Datenträgers

Literatur

[AD20] Mohammed Fadhel Aljunid und Manjaiah Dh. “An Efficient Deep Learning Approach for Collaborative Filtering Recommender System”. en. In: *Procedia Computer Science* 171 (2020), S. 829–836. ISSN: 18770509. DOI: 10.1016/j.procs.2020.04.090 (siehe S. 5, 7).

[Agg16] Charu C. Aggarwal. “Content-Based Recommender Systems”. In: *Recommender Systems*. Cham: Springer International Publishing, 2016, S. 139–166. ISBN: 978-3-319-29657-9 978-3-319-29659-3. DOI: 10.1007/978-3-319-29659-3_4 (siehe S. 16 f.).

[Bau+09] M. Baur, E. Fransen, A. Tropitzsch, L. van Laer, P.S. Mauz, G. Van Camp, N. Blin und M. Pfister. “Einfluss exogener Faktoren auf Altersschwerhörigkeit”. In: *HNO* 57.10 (Okt. 2009), S. 1023–1028. ISSN: 0017-6192, 1433-0458. DOI: 10.1007/s00106-009-1900-9 (siehe S. 23).

[BBI13] Nelly Bencomo, Amel Belaggoun und Valerie Issarny. “Dynamic decision networks for decision-making in self-adaptive systems: A case study”. In: *2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. San Francisco, CA, USA: IEEE, Mai 2013, S. 113–122. ISBN: 978-1-4673-4401-2 978-1-4799-0344-3. DOI: 10.1109/SEAMS.2013.6595498 (siehe S. 6).

[Ben+13] David Benavides, Alexander Felfernig, José A. Galindo und Florian Reinfrank. “Automated Analysis in Feature Modelling and Product Configuration”. In: *Safe and Secure Software Reuse*. Hrsg. von John Favaro und Maurizio Morisio. Bd. 7925. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 160–175. ISBN: 978-3-642-38976-4 978-3-642-38977-1. DOI: 10.1007/978-3-642-38977-1_11 (siehe S. 19).

[Ben20] Nelly Bencomo. “Next steps in variability management due to autonomous behaviour and runtime learning”. en. In: *Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems*. Magdeburg Germany: ACM, Feb. 2020, S. 1–2. ISBN: 978-1-4503-7501-6. DOI: 10.1145/3377024.3380451 (siehe S. 6).

[Bir+16] Melanie Birks, Marie Bodak, Joanna Barlas, June Harwood und Mary Pether. “Robotic Seals as Therapeutic Tools in an Aged Care Facility: A Qualitative Study”. In: *Journal of Aging Research* 2016 (20. Nov. 2016). Publisher: Hindawi, e8569602. ISSN: 2090-2204. DOI: 10.1155/2016/8569602 (siehe S. 12).

[Bis06] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. New York: Springer, 2006. 738 S. ISBN: 978-0-387-31073-2 (siehe S. 14).

[Bro+12] Elizabeth Broadbent, Rie Tamagawa, Anna Patience, Brett Knock, Ngaire Kerse, Karen Day und Bruce A. MacDonald. "Attitudes towards health-care robots in a retirement village". In: *Australasian Journal on Ageing* 31.2 (2012), S. 115–120. ISSN: 1741-6612. DOI: 10.1111/j.1741-6612.2011.00551.x (siehe S. 12).

[Bro+16] Elizabeth Broadbent, Ngaire Kerse, Kathryn Peri, Hayley Robinson, Chandimal Jayawardena, Tony Kuo, Chandan Datta, Rebecca Stafford, Haley Butler, Pratyusha Jawalkar, Maddy Amor, Ben Robins und Bruce MacDonald. "Benefits and problems of healthcare robots in aged care settings: A comparison trial". In: *Australasian Journal on Ageing* 35.1 (2016), S. 23–29. ISSN: 1741-6612. DOI: 10.1111/ajag.12190 (siehe S. 12).

[Car+20] Felix Carros, Johanna Meurer, Diana Löffler, David Unbehaun, Sarah Matthies, Inga Koch, Rainer Wieching, Dave Randall, Marc Hassenzahl und Volker Wulf. "Exploring Human-Robot Interaction with the Elderly: Results from a Ten-Week Case Study in a Care Home". en. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Honolulu HI USA: ACM, Apr. 2020, S. 1–12. ISBN: 978-1-4503-6708-0. DOI: 10.1145/3313831.3376402 (siehe S. 9).

[CE00] Krzysztof Czarnecki und Ulrich W Eisenecker. *Generative Programming: Methods, Tools and Applications*. Addison-Wesley, 2000 (siehe S. 18 f.).

[Che+12] Minmin Chen, Zhixiang Xu, Kilian Weinberger und Fei Sha. *Marginalized Denoising Autoencoders for Domain Adaptation*. 2012. DOI: 10.48550/ARXIV.1206.4683. URL: https://arxiv.org/abs/1206.4683 (siehe S. 16).

[Chi+19] Nayden Chivarov, Denis Chikurtev, Stefan Chivarov, Matus Pleva, Stanislav Ondas, Jozef Juhar und Kaloyan Yovchev. "Case Study on Human-Robot Interaction of the Remote-Controlled Service Robot for Elderly and Disabled Care". In: *Computing and Informatics* 38.5 (2019), S. 1210–1236. ISSN: 2585-8807. DOI: 10.31577/cai_2019_5_1210 (siehe S. 4).

[Dar+17] Margot Darragh, Ho Seok Ahn, Bruce MacDonald, Amy Liang, Kathryn Peri, Ngaire Kerse und Elizabeth Broadbent. "Homecare Robots to Improve Health and Well-Being in Mild Cognitive Impairment and Early Stage Dementia: Results From a Scoping Study". In: *Journal of the American Medical Directors Association* 18.12 (1. Dez. 2017), 1099.e1–1099.e4. ISSN: 1525-8610. DOI: 10.1016/j.jamda.2017.08.019 (siehe S. 12).

[Das+13] Biswajit Das, Sandipan Mandal, Pabitra Mitra und Anupam Basu. "Effect of aging on speech features and phoneme recognition: a study on Bengali voicing vowels". In: *International Journal of Speech Technology* 16.1 (März 2013), S. 19–31. ISSN: 1381-2416, 1572-8110. DOI: 10.1007/s10772-012-9147-3 (siehe S. 24).

[Den+19] Zhi-Hong Deng, Ling Huang, Chang-Dong Wang, Jian-Huang Lai und Philip S. Yu. "DeepCF: A Unified Framework of Representation Learning and Matching Function Learning in Recommender System". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (Juli 2019), S. 61–68. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v33i01.330161 (siehe S. 8).

[DN20] Matthieu Deru und Alassane Ndiaye. *Deep Learning mit TensorFlow, Keras und TensorFlow.js*. 2. Aufl. Rheinwerk Computing, 28. Apr. 2020. 496 S. ISBN: 978-3-8362-7427-2 (siehe S. 27).

[Fie97] Reinhard Fiehler. “Kommunikation im Alter und ihre sprachwissenschaftliche Analyse. Gibt es einen Kommunikationsstil des Alters?” In: *Sprech- und Gesprächsstile*. Hrsg. von Margret Selting und Barbara Sandig. Berlin, Boston: DE GRUYTER, 31. Jan. 1997. ISBN: 978-3-11-082044-7. DOI: 10.1515/9783110820447.345 (siehe S. 25).

[Fis+19] Arthur D. Fisk, Walter R Boot, Neil H Charness, Wendy A Rogers und Sara J Czaja. *Designing for Older Adults: principles and creative human factors approaches*. OCLC: 1101430677. 2019. ISBN: 978-1-138-05366-3 978-0-367-13818-9 (siehe S. 23 f., 39).

[Fis+20] Marcus Fischer, David Heim, Adrian Hofmann, Christian Janiesch, Christoph Klima und Axel Winkelmann. “A taxonomy and archetypes of smart services for smart living”. In: *Electronic Markets* 30.1 (März 2020), S. 131–149. ISSN: 1019-6781, 1422-8890. DOI: 10.1007/s12525-019-00384-5 (siehe S. 14).

[GB19] Luis Hernan Garcia Paucar und Nelly Bencomo. “Knowledge Base K Models to Support Trade-Offs for Self-Adaptation using Markov Processes”. In: *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems* (SASO). Umea, Sweden: IEEE, Juni 2019, S. 11–16. ISBN: 978-1-72812-731-6. DOI: 10.1109/SASO.2019.00011 (siehe S. 8).

[GBC16] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep learning*. OCLC: 1224542700. Cambridge, Massachusetts: The MIT Press, 2016. ISBN: 978-0-262-33737-3 (siehe S. 14, 16).

[Gra+19] Florenz Graf, Cagatay Odabasi, Theo Jacobs, Birgit Graf und Thomas Fodisch. “MobiKa - Low-Cost Mobile Robot for Human-Robot Interaction”. In: *2019 28th IEEE International Conference on Robot and Human Interactive Communication* (RO-MAN). New Delhi, India: IEEE, Okt. 2019, S. 1–6. ISBN: 978-1-72812-622-7. DOI: 10.1109/RO-MAN46459.2019.8956405 (siehe S. 9).

[Grö21] Tim Gröger. “Konzeption eines Konfigurationsprozesses zur adaptiven Interaktion mit Assistenzrobotern”. Großer Beleg. Dresden: Technische Universität Dresden, Feb. 2021 (siehe S. 1).

[GS15] Przemyslaw Gilski und Jacek Stefanski. “Android os: a review”. In: *Tem Journal* 4.1 (2015). Publisher: UIKTEN-Association for Information Communication Technology Education and Science, S. 116 (siehe S. 51).

[Hau20] Yasmin Haustein. “Erfolgsmethoden für Sprachbenutzungsoberflächen von Assistenzrobotern für Senioren”. Bachelorarbeit. Dresden: Technische Universität Dresden, Mai 2020 (siehe S. 24 f., 31, 70, 72).

[He+17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu und Tat-Seng Chua. “Neural Collaborative Filtering”. In: *Proceedings of the 26th International Conference on World Wide Web*. WWW ’17: 26th International World Wide Web Conference. Perth Australia: International World Wide Web Conferences Steering Committee, Apr. 2017, S. 173–182. ISBN: 978-1-4503-4913-0. DOI: 10.1145/3038912.3052569 (siehe S. 28, 41 f., 71).

[Hee+10] Marcel Heerink, Ben Kröse, Vanessa Evers und Bob Wielinga. “Assessing Acceptance of Assistive Social Agent Technology by Older Adults: the Almere Model”. In: *International Journal of Social Robotics* 2.4 (Dez. 2010), S. 361–375. ISSN: 1875-4791, 1875-4805. DOI: 10.1007/s12369-010-0068-5 (siehe S. 11 f., 14, 69).

[HH10] Denise Heger und Inga Holube. “Wie viele Menschen sind schwerhörig”. In: *Zeitschrift für Audiologie Audiological Acoustics* 49.2 (Mai 2010), S. 61–70 (siehe S. 34, 36, 38 f., 71).

[JZH21] Christian Janiesch, Patrick Zschech und Kai Heinrich. “Machine learning and deep learning”. In: *Electronic Markets* 31.3 (Sep. 2021), S. 685–695. ISSN: 1019-6781, 1422-8890. DOI: 10.1007/s12525-021-00475-2 (siehe S. 14–16).

[KAK08] Christian Kästner, Sven Apel und Martin Kuhlemann. “Granularity in software product lines”. In: *Proceedings of the 13th international conference on Software engineering - ICSE ’08*. the 13th international conference. Leipzig, Germany: ACM Press, 2008, S. 311. ISBN: 978-1-60558-079-1. DOI: 10.1145/1368088.1368131 (siehe S. 19 f.).

[Kan+90] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak und A Spencer Peterson. *Feature-oriented domain analysis (FODA) feasibility study*. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990 (siehe S. 19).

[Kit+15] Ralf Kittmann, Tim Fröhlich, Johannes Schäfer, Ulrich Reiser, Florian Weißhardt und Andreas Haug. “Let me Introduce Myself: I am Care-O-bot 4, a Gentleman Robot”. In: *Mensch und Computer 2015 – Proceedings*. Hrsg. von Sarah Diefenbach, Niels Henze und Martin Pielot. Berlin: De Gruyter Oldenbourg, 2015, S. 223–232 (siehe S. 14).

[LAG10] Akeem O. Lasisi, Taiwo Abiona und Oye Gureje. “Tinnitus in the elderly: Profile, correlates, and impact in the Nigerian Study of Ageing”. In: *Otolaryngology–Head and Neck Surgery* 143.4 (Okt. 2010), S. 510–515. ISSN: 0194-5998, 1097-6817. DOI: 10.1016/j.otohns.2010.06.817 (siehe S. 23).

[LBH15] Yann LeCun, Yoshua Bengio und Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (Mai 2015), S. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539 (siehe S. 14, 16).

[Lee15] Ji Young Lee. “Aging and Speech Understanding”. In: *Journal of Audiology & Otology* 19.1 (Apr. 2015), S. 7–13. ISSN: 2384-1621. DOI: 10.7874/jao.2015.19.1.7 (siehe S. 39).

[Lee21] Jaeryoung Lee. “Generating Robotic Speech Prosody for Human Robot Interaction: A Preliminary Study”. en. In: *Applied Sciences* 11.8 (Apr. 2021), S. 3468. ISSN: 2076-3417. DOI: 10.3390/app11083468 (siehe S. 3).

[Mis+20] Justinas Miseikis, Pietro Caroni, Patricia Duchamp, Alina Gasser, Rastislav Marko, Nelija Miseikiene, Frederik Zwilling, Charles de Castelbajac, Lucas Eicher, Michael Fruh und Hansruedi Fruh. “Lio-A Personal Robot Assistant for Human-Robot Interaction and Care Applications”. In: *IEEE Robotics and Automation Letters* 5.4 (Okt. 2020), S. 5339–5346. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2020.3007462 (siehe S. 9).

[Mül16] Steffen Müller. “Realisierung nutzeradaptiven Interaktionsverhaltens für mobile Assistenzroboter”. Diss. Technische Universität Ilmenau, 2016 (siehe S. 1).

[Mun11] Jennifer S Mundorff. “Effects of speech signal type and attention on acceptable noise level in elderly, hearing-impaired listeners”. Diss. James Madison University, 2011 (siehe S. 24).

[PBL05] Klaus Pohl, Günter Böckle und Frank van der Linden. *Software product line engineering: foundations, principles, and techniques*. 1st ed. New York, NY: Springer, 2005. 467 S. ISBN: 978-3-540-24372-4 (siehe S. 17 f.).

[PG18] Amit Kumar Pandey und Rodolphe Gelin. “A Mass-Produced Sociable Humanoid Robot: Pepper: The First Machine of Its Kind”. In: *IEEE Robotics & Automation Magazine* 25.3 (Sep. 2018), S. 40–48. ISSN: 1070-9932, 1558-223X. DOI: 10.1109/MRA.2018.2833157 (siehe S. 14).

[PKA18] Irena Papadopoulos, Christina Koulouglioti und Sheila Ali. “Views of nurses and other health and social care workers on the use of assistive humanoid and animal-like robots in health and social care: a scoping review”. In: *Contemporary Nurse* 54.4 (4. Juli 2018), S. 425–442. ISSN: 1037-6178, 1839-3535. DOI: 10.1080/10376178.2018.1519374 (siehe S. 12).

[PLF20] Alisha Pradhan, Amanda Lazar und Leah Findlater. “Use of Intelligent Voice Assistants by Older Adults with Low Technology Use”. en. In: *ACM Transactions on Computer-Human Interaction* 27.4 (Aug. 2020), S. 1–27. ISSN: 1073-0516, 1557-7325. DOI: 10.1145/3373759 (siehe S. 5).

[Qui+21] Clément Quinton, Michael Vierhauser, Rick Rabiser, Luciano Baresi, Paul Grünbacher und Christian Schuhmayer. “Evolution in dynamic software product lines”. In: *Journal of Software: Evolution and Process* 33.2 (Feb. 2021). ISSN: 2047-7473, 2047-7481. DOI: 10.1002/smr.2293 (siehe S. 20–22).

[Ram+16] Hassan Ramchoun, Mohammed Amine, Janati Idrissi, Youssef Ghanou und Mohamed Ettaouil. “Multilayer Perceptron: Architecture Optimization and Training”. In: *International Journal of Interactive Multimedia and Artificial Intelligence* 4.1 (2016), S. 26. ISSN: 1989-1660. DOI: 10.9781/ijimai.2016.415 (siehe S. 26).

[Ren10] Steffen Rendle. “Factorization Machines”. In: *2010 IEEE International Conference on Data Mining*. 2010 IEEE 10th International Conference on Data Mining (ICDM). Sydney, Australia: IEEE, Dez. 2010, S. 995–1000. ISBN: 978-1-4244-9131-5. DOI: 10.1109/ICDM.2010.127 (siehe S. 42).

[Rus+19] Alessandro Russo, Grazia D’Onofrio, Aldo Gangemi, Francesco Giuliani, Misael Mongiovi, Francesco Ricciardi, Francesca Greco, Filippo Cavallo, Paolo Dario, Daniele Sancarlo, Valentina Presutti und Antonio Greco. “Dialogue Systems and Conversational Agents for Patients with Dementia: The Human–Robot Interaction”. en. In: *Rejuvenation Research* 22.2 (Apr. 2019), S. 109–120. ISSN: 1549-1684, 1557-8577. DOI: 10.1089/rej.2018.2075 (siehe S. 4).

[Sch+12] Ina Schaefer, Rick Rabiser, Dave Clarke, Lorenzo Bettini, David Benavides, Goetz Botterweck, Animesh Pathak, Salvador Trujillo und Karina Villela. “Software diversity: state of the art and perspectives”. In: *International Journal on Software Tools for Technology Transfer* 14.5 (Okt. 2012), S. 477–495. ISSN: 1433-2779, 1433-2787. DOI: 10.1007/s10009-012-0253-y (siehe S. 19 f.).

[Sha21] Qusai Shambour. “A deep learning based algorithm for multi-criteria recommender systems”. en. In: *Knowledge-Based Systems* 211 (Jan. 2021), S. 106545. ISSN: 09507051. DOI: 10.1016/j.knosys.2020.106545 (siehe S. 8).

[SK09] Xiaoyuan Su und Taghi M. Khoshgoftaar. “A Survey of Collaborative Filtering Techniques”. In: *Advances in Artificial Intelligence* 2009 (27. Okt. 2009), S. 1–19. ISSN: 1687-7470, 1687-7489. DOI: 10.1155/2009/421425 (siehe S. 17).

[SNC19] Sergio Sayago, Barbara Barbosa Neves und Benjamin R Cowan. “Voice assistants and older people: some open issues”. en. In: *Proceedings of the 1st International Conference on Conversational User Interfaces - CUI '19*. Dublin, Ireland: ACM Press, 2019, S. 1–3. ISBN: 978-1-4503-7187-2. DOI: 10.1145/3342775.3342803 (siehe S. 5).

[SRR17] P Sivaseshu, D Ravikanth und K Rajagopal. “Finite element analysis of silo surface cleaning robot”. In: *International Journal of Scientific Engineering Technology Research* (2017), S. 4249–4252 (siehe S. 11).

[SSA14] Christoph Seidl, Ina Schaefer und Uwe Aßmann. “Integrated management of variability in space and time in software families”. In: *Proceedings of the 18th International Software Product Line Conference - Volume 1*. SPLC '14: 18th International Software Product Line Conference. Florence Italy: ACM, 15. Sep. 2014, S. 22–31. ISBN: 978-1-4503-2740-4. DOI: 10.1145/2648511.2648514 (siehe S. 17–20).

[Tan+20] Ana Tanevska, Francesco Rea, Giulio Sandini, Lola Cañamero und Alessandra Sciutti. “A Socially Adaptable Framework for Human-Robot Interaction”. In: *Frontiers in Robotics and AI* 7 (Okt. 2020), S. 121. ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00121 (siehe S. 3).

[Tem+17] Paul Temple, Mathieu Acher, Jean-Marc Jezequel und Olivier Barais. “Learning Contextual-Variability Models”. In: *IEEE Software* 34.6 (Nov. 2017), S. 64–70. ISSN: 0740-7459, 1937-4194. DOI: 10.1109/MS.2017.4121211 (siehe S. 5 f.).

[Tre+18] Pascale Tremblay, Isabelle Deschamps, Pascale Bédard, Marie-Hélène Tessier, Micaël Carrier und Mélanie Thibeault. “Aging of speech production, from articulatory accuracy to motor timing.” In: *Psychology and Aging* 33.7 (Nov. 2018), S. 1022–1034. ISSN: 1939-1498, 0882-7974. DOI: 10.1037/pag0000306 (siehe S. 24).

[TS15] Philippe Thomas und Marie-Christine Suhner. “A New Multilayer Perceptron Pruning Algorithm for Classification and Regression Applications”. In: *Neural Processing Letters* 42.2 (Okt. 2015), S. 437–458. ISSN: 1370-4621, 1573-773X. DOI: 10.1007/s11063-014-9366-5 (siehe S. 26).

[UDP20] United Nations, Department of Economic and Social Affairs und Population Division. *World population ageing 2020 Highlights: living arrangements of older persons*. en. OCLC: 1231958835. 2020. ISBN: 978-92-1-148347-5 (siehe S. 1).

[UDP22] United Nations, Department of Economic and Social Affairs und Population Division. *World population prospects 2022: summary of results*. OCLC: 1340426535. New York: United Nations, 2022. ISBN: 978-92-1-001438-0 (siehe S. 1, 37).

[Vdi90] Verein Deutscher Ingenieure e. V. *VDI-Richtlinie-2860: Montage-und Handhabungstechnik*. VDI, Düsseldorf, 1990 (siehe S. 11).

[WFG11] Nele Wild-Wall, Michael Falkenstein und Patrick D. Gajewski. “Age-Related Differences in Working Memory Performance in A 2-Back Task”. In: *Frontiers in Psychology* 2 (2011). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2011.00186 (siehe S. 24).

[WHO21] *World report on hearing*. OCLC: 1295473345. Geneva: World Health Organization, 2021. ISBN: 978-92-4-002048-1 (siehe S. 34).

[Wil+08] Terry L. Wiley, Rick Chappell, Lakeesha Carmichael, David M. Nondahl und Karen J. Cruickshanks. “Changes in Hearing Thresholds over 10 Years in Older Adults”. In: *Journal of the American Academy of Audiology* 19.4 (Apr. 2008), S. 281–292. ISSN: 1050-0545, 2157-3107. DOI: 10.3766/jaaa.19.4.2 (siehe S. 23, 35, 39).

[Wis96] Karl Heinz Wisotzki. *Altersschwerhörigkeit: Grundlagen - Symptome - Hilfen*. Stuttgart: Kohlhammer, 1996. 263 S. ISBN: 978-3-17-014019-6 (siehe S. 23).

[Yua+16] Jianbo Yuan, Walid Shalaby, Mohammed Korayem, David Lin, Khalifeh AlJadda und Jiebo Luo. “Solving cold-start problem in large-scale recommendation engines: A deep learning approach”. In: *2016 IEEE International Conference on Big Data (Big Data)*. 2016 IEEE International Conference on Big Data (Big Data). Washington DC,USA: IEEE, Dez. 2016, S. 1901–1910. ISBN: 978-1-4673-9005-7. DOI: 10.1109/BigData.2016.7840810 (siehe S. 16 f.).

[Zha+20] Shuai Zhang, Lina Yao, Aixin Sun und Yi Tay. “Deep Learning Based Recommender System: A Survey and New Perspectives”. en. In: *ACM Computing Surveys* 52.1 (Jan. 2020), S. 1–38. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3285029 (siehe S. 8, 16).

[ZL18] Ying Zhang und Chen Ling. “A strategy to apply machine learning to small datasets in materials science”. In: *npj Computational Materials* 4.1 (Dez. 2018), S. 25. ISSN: 2057-3960. DOI: 10.1038/s41524-018-0081-z (siehe S. 16).

Onlinequellen

[Ald20] Aldebaran United Robotics Group. *Pepper in Healthcare*. SoftBank Robotics. 10. März 2020. URL: https://www.softbankrobotics.com/emea/en/pepper-healthcare-ga (besucht am 31. 08. 2022) (siehe S. 14).

[Cic20] Christiane Cichy. *Helden am Limit: Warum Pflegekräfte in Deutschland chronisch überlastet sind*. archive.org: https://web.archive.org/web/20200411114818/https://www.daserste.de/information/wirtschaft-boerse/plusminus/sendung/ueberlastetes-pflegepersonal-in-deutschland-100.html (besucht am 14.12.2022). 2020. URL: https://www.daserste.de/information/wirtschaft-boerse/plusminus/sendung/ueberlastetes-pflegepersonal-in-deutschland-100.html (besucht am 16. 10. 2020) (siehe S. 1).

[Deu22] Deutsche Alzheimer Gesellschaft e. V. *Prävalenzrate von Demenz nach Alter und Geschlecht*. Publication Title: Statista. Aug. 2022. URL: https://de.statista.com/statistik/daten/studie/246021/umfrage/praevalenzrate-von-demenzerkrankungen-in-deutschland-nach-alter-und-geschlecht/ (besucht am 01. 09. 2022) (siehe S. 36, 39, 71).

[MR16] Rolf Müller und Heinz Rothgang. *Bedarfs– und Angebotsanalyse und –prognose über Ausbildungsplätze in der Altenpflegeausbildung*. 2016. URL: https://www.socium.uni-bremen.de/uploads/News/2015/150212_Gutachten_Altenpflegeausbildung_Bericht.pdf (besucht am 07. 06. 2021) (siehe S. 1).

[Pro15] Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA. *Roboter als vielseitiger Gentleman*. Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA. 15. Jan. 2015. URL: https://www.ipa.fraunhofer.de/de/presse/presseinformationen/2015-01-15_roboter-als-vielseitiger-gentleman.html (besucht am 31. 08. 2022) (siehe S. 14).

[Ras22] Rasa Technologies GmbH. *Introduction to Rasa Open Source — rasa.com*. 2022. URL: https://rasa.com/docs/rasa/ (besucht am 10. 07. 2022) (siehe S. 52).

[Rei22] Kathy Reid. *Mark II Media Kit - Mycroft — mycroft.ai*. 2022. URL: https://mycroft.ai/media/ (besucht am 10. 07. 2022) (siehe S. 52).

[Sch18] Torsten Schmitt. *Loomo von Segway Robotics - das Hoverboard, das dich verfolgt*. techkrams.de. 2018. URL: https://techkrams.de/loomo-von-segway-robotics-das-hoverboard-das-dich-verfolgt/ (besucht am 20. 07. 2022) (siehe S. 51).

[Sta22] Statistisches Bundesamt. *Altersstruktur der Bevölkerung in Deutschland 2021*. Statistisches Bundesamt. Tabelle: 12411-0005. Dez. 2021. URL: https://www-genesis.destatis.de/genesis/online (besucht am 22. 06. 2022) (siehe S. 37 f., 71).