Carl von Ossietzky Universität Oldenburg

German Research Center for Artificial Intelligence

Projektbericht der Projekgruppe Multimodal Multisensor Interaction II

– Projektname –

SmartTableTennis

Carl von Ossietzky Universität Oldenburg
Department für Informatik | Applied Artificial Intelligence

Projekgruppe Multimodal Multisensor Interaction II

(Namen der Studierenden zur Veröffentlichung aus Datenschutzgründen entfernt)
pg-mmi2@lists.uni-oldenburg.de

Zweitprüfer
Prof. Dr.-Ing. Daniel Sonntag
Applied Artificial Intelligence, Oldenburg University
daniel.sonntag@dfki.de

Erstprüfer
Bengt Lüers
Applied Artificial Intelligence, Oldenburg University
bengt.lueers@dfki.de

Stellvertretender Erstprüfer
Michael Barz
Applied Artificial Intelligence, University of Oldenburg

michael.barz@dfki.de

Bengt Lüers	Ort, Datum
Projekgruppe Multimodal Multisensor Interaction II	Ort, Datum

Inhaltsverzeichnis

[1] Dieser Abschnitt ist in Teilen aus dem Review-Papier „Multimodal Learning Analytics und Learning Analytics im Rahmen des Serious Gaming“ übernommen worden.

[2] Dieser Abschnitt wurde größtenteils aus dem Grundlagenkapitel des Proposals der PGMMI2 übernommen und sinngemäß übersetzt.

[3] Dieser Abschnitt wurde in Teilen aus dem Reviewpaper „AI-basierte Dialogsysteme und ChatBots“ übernommen

Kapitel 1

Einleitung

Aus Gründen der Lesbarkeit wird in diesem Dokument der generische Maskulin verwendet und somit auf geschlechtsspezifische Sprache verzichtet.

Dieses Dokument dient als Dokumentation und Bericht der Projektgruppe 2 für Multimodal Multisensor Interaction (PG-MMI2). Besagtes Projekt wurde im Rahmen des Masterstudiums an der Carl von Ossietzky Universität Oldenburg durchgeführt und stellt eine Zusammenarbeit der Gruppe für Multimodal Multisensor Interaction der Universität und des Deutschen Forschungsinstitutes für Künstliche Intelligenz (DFKI) dar. Aufgabenstellung für das Projekt war es, über das Wintersemester 2022/2023 und das Sommersemester 2023 einen Virtual Reality (VR) Demonstrator zu konzipieren, zu entwickeln und im Rahmen einer Benutzerstudie zu bewerten. Bei den Projektteilnehmern handelt es sich um eine Gruppe von initial zehn Masterstudierenden der Fächer Informatik, Wirtschaftsinformatik und Engineering of Socio-Technical Systems. Die Projektteilnehmer konzipierten und implementierten *SmartTableTennis*: ein VR-Tischtennis-Spiel mit intelligenten Assistenzfunktionen. Die ursprüngliche Vision ist in Kapitel 4 zu sehen, während die tatsächliche Umsetzung in Kapitel 5 zu finden ist. Die Auswertung des Systems durch die Nutzerstudie, sowie deren Aufbau und Verlauf, sind in den Kapiteln 6 und 7 dokumentiert.

1.1 Motivation und Forschungsfragen

VR-Technologien bieten dem Nutzer ein hohes Maß an Immersion in die virtuelle Umgebung sowie diverse Interaktionsmöglichkeiten mit dieser [Alcañiz Raya et al., 2020]. Dies führt dazu, dass VR-Technologien in verschiedenen Bereichen eingesetzt werden, ob nun für militärische Zwecke [Checa and Bustillo, 2020], dem Gesundheitssektor [Alcañiz Raya et al., 2020] oder sogar dem Bereich des Sports [Neumann et al., 2018].

Bei der Verwendung von VR-Technologien im Sportsektor handelt es sich um ein noch offenes Forschungsfeld, weshalb hier ein exploratives Vorgehen erfolgt. Hierbei sollen verschiedene Aspekte mit Hinsicht auf ein eigens konstruiertes VR-Environment mit Bezug zum Sportsektor untersucht werden. Ein VR-Environment, mit dem Ziel, den Nutzern Abläufe bestimmter sportlicher Aktivitäten beizubringen, könnte als *Serious Game* (SG) verstanden werden. In [Göbl et al., 2021] wird die Nutzung von Konversationsinterfaces bzw. Chatbots im Rahmen von SGs untersucht, wobei die

SGs in ihrer Einsatzdomäne und dem Zweck des Chatbots unterschieden werden. Aus [Göbl et al., 2021] geht hervor, dass noch keine Chatbot-Anwendung im Rahmen von SGs für den Sportsektor vorlag. Für VR-Anwendungen im Sportsektor liegt nach [Neumann et al., 2018] kein theoretisches Framework vor, nach welchem das System in diesem Projekt konstruiert und evaluiert werden kann, weshalb der Fokus auf der Usability des Systems liegen soll. Daraus ergeben sich die folgenden Forschungsfragen: **RQ1: Wie lässt sich ein VR-Serious-Game für den Sportsektor, am Beispiel von Tischtennis, konstruieren? RQ2: Wie lässt sich ein Konversationsinterface bzw. Chatbot in der Nutzung eines VR Serious Games für den Sportsektor integrieren und welchen Einfluss hat dies auf die Usability des Systems?**.

Der Demonstrator für dieses Projekt ist in Unity VR zu implementieren. Im Jahr 2018 wurde in [Juliani et al., 2018] das Unity ML-Agents Toolkit vorgestellt, das zusammen mit der Unity Engine als Plattform für intelligente Agenten genutzt werden kann. So sollen die Vorteile der Unity Engine, namentlich die Fähigkeit der Engine sensorische, physikalische, aufgabenlogische und soziale Komplexität darzustellen und einfach manipulieren zu können, zugunsten der Konstruktion von Trainingsumgebungen, für intelligente Agenten genutzt werden können [Juliani et al., 2018]. Auf eine Eignung bzw. Nicht-Eignung von Unity ML-Agents[1] für das Training von Agenten, die Anwendung in VR-Environments finden, wird in [Juliani et al., 2018] nicht eingegangen. Daraus lässt sich folgende Forschungsfrage ableiten: **RQ3: Wie lassen sich mit Unity ML-Agents konstruierte Agenten in eine höherdimensionales Umgebung, wie einem Serious Game in VR, integrieren und welche Probleme wirft das ML-Agents Toolkit dabei auf?** Die „höhere Dimensionalität“ bezieht sich hierbei auf die Komplexität und Größe des Observations- und Aktionsraums eines Agenten. So muss sich der KI-Gegenspieler beispielsweise selbst im Raum bewegen können (Bewegungen nach vorne, hinten, links und rechts, sowie Rotation des Körpers) und den Schläger im dreidimensionalen Raum bewegen können (Bewegung nach vorne, hinten, links, rechts, oben und unten, sowie Rotation des Schlägers), um korrekte Schläge auszuführen. Darüber hinaus muss der Bot in der Lage sein, aus diesen Bewegungsrichtungen für Körper und Schläger, diagonale Bewegungen durchzuführen können, um den Ball über das Netz spielen zu.

Unter der Betrachtung des Demonstrators als SG und dem Ziel der Implementierung von Assistenzfunktionen, ist das Feld der *Game Learning Analytics* (GLA) [Freire et al., 2016] beziehungsweise der *Learning Analytics* (LA) [Oviatt et al., 2018], ebenfalls relevant. Grund hierfür ist, dass besagte Assistenzfunktionen auf der Aktivität des Spielers basieren sollen, wofür die Erfassung und Auswertung der vom Spiel generierten Performancedaten nötig ist. In Abschnitt 3.1.1 wird der sogenannte *Learning Analytics Cycle* nach [Clow, 2012] beschrieben, welcher zusammengefasst die Auswertung von Nutzerdaten innerhalb eines Systems und die Unterbreitung von Feedback an den Nutzer vorsieht. Im Falle dieses Projektes kann dies über die Erkennung von Spielerfehlern und der Evaluation der Spielerperformance erfolgen. Herausfordernd ist hierbei die Anforderung einer umfangreichen Datengrundlage, die LA- beziehungsweise *Multimodal Learning Analytics*-Systeme (MLA), stellen [Oviatt et al., 2018]. Um dieser Anforderung zu begegnen, wird die Nutzung synthetischer Daten in Betracht gezogen, welche auf einem zuvor aufgenommenen, kleineren

[1] Dokumentation Unity ML-Agents: `https://unity-technologies.github.io/ml-agents/`, letzter Aufruf: 29.09.2023

Datensatz basieren und mit einem oder mehreren verschiedenen Datengeneratoren, generiert werden sollen.

Im Rahmen von [Hittmeir et al., 2019] wurden fünf Datensätze, mit zwischen fünf und 15 Features und zwei bis drei Klassen, herangezogenen. Auf Basis dieser Datensätze wurden mit drei verschiedenen Verfahren synthetischen Daten generiert, Modelle auf Basis fünf verschiedener Klassifikationsverfahren trainiert und anschließend mit Hinblick auf ihre Genauigkeit getestet. Die drei Generationsverfahren unterschieden sich hierbei in ihrer Verwendung sogenannter *privacy-reserving methods*. Hierbei handelt es sich um Maßnahmen zur Vorbeugung der Nachvollziehbarkeit von personenbezogenen Daten, mit denen ein Netz trainiert wurde, nachdem das Training erfolgt ist. Beispielsweise, wenn ein Modell mit Daten wie dem durchschnittlichen Gehalt von Erwachsenen trainiert wird und in besagtem Trainingsdatensatz einem Namen zugeordnet sind, soll aus den Ausgaben des fertig trainierten Modells nicht nachvollziehbar sein, was das durchschnittliche Gehalt einer bestimmten Person im Trainingsdatensatz ist. Die Ergebnisse von [Hittmeir et al., 2019] ergaben, dass das Training mit synthetischen Daten bei der Klassifikation zu Accuracy-Einbußen führten. Diese lagen zwischen 0,1% und 7,4%, wobei 7,4% nur einmal auftrat und der nächsthöhere Verlust bei 4,9& lag. In wenigen Fällen konnten die mit unbehandelten, synthetischen Daten trainierte Classifier, die, die nur mit dem Trainingsdatensatz trainiert wurden, um bis zu 1,6% out performen. Das Training mit synthetischen Daten unter der Einführung von *privacy-reserving methods* erhöhte die Verluste vereinzelt auf bis zu 50%. [Hittmeir et al., 2019]

[Hittmeir et al., 2019] ziehen daraus, synthetische Daten ohne *privacy-reserving methods* für die Anwendung bei Klassifikationsproblemen im Rahmen von *Supervised Machine Learning* geeignet sind. Die Arbeit von [Hittmeir et al., 2019] hat jedoch keinen Bezug zu VR-Umgebungen oder vergleichbarem, woraus sich im Kontext der bisherigen Forschungsfragen folgende ableiten lässt: **RQ4: Lassen sich mit synthetischen Daten trainierte Machine-Learning-Komponenten, für die Klassifikation von Spielerbewegungen im Rahmen eines VR-Environments nutzen?**

Kapitel 2

Projektmanagement

In diesem Kapitel wird die Organisation der Gruppe über den Verlauf des Projektes thematisiert. Hierbei soll sowohl auf zur Organisation verwendete Tools als auch die Rollenverteilung während des Projektes eingegangen werden. Am Ende dieses Kapitels wird das organisatorische Vorgehen kritisch reflektiert.

2.1 Verwendete Tools

Für die gruppeninterne Absprache wurde von den Projektteilnehmern der Onlinedienst *Discord*[1] genutzt. Dieser ermöglichte schnellen Austausch über Direktnachrichten, die Koordination mit der gesamten Gruppe über geteilte Chatkanäle, sowie Gruppenmeetings über frei zugängliche Sprachkanäle.

Für die Aufgabenorganisation und Versionskontrolle wurde das DevOps Werkzeug *GitLab* genutzt oder spezifischer die vom DFKI zur Verfügung gestellte *GitLab*-Infrastruktur. Issues und die Issue-Boards auf denen diese organisiert sind, auf *GitLab* wurden hierbei nicht nur für Programmieraufgaben, sondern auch organisatorische Aufgaben wie Dokumentationsarbeiten angelegt, um diese zu verteilen und zu organisieren. Im Rahmen von Entwicklungsaufgaben ist mit *Feature Branches* gearbeitet worden. Dies bedeutet, dass für jede separate Entwicklungsaufgabe, ob nun auf Frontend oder Backend, ein eigener Branch angelegt wurde, auf welchem besagte Aufgabe bearbeitet wurde. Der Stand des *Feature Branches* wurde dann nach Abschluss der Arbeiten im Rahmen einer *Merge Request* in den aktuellen lauffähigen Branch eingepflegt, wobei jede *Merge Request* von einem zweiten Gruppenmitglied überprüft wurde.

2.2 Gruppenorganisation und Vorgehensmodell

Die Projektteilnehmer teilten die Gruppe in zwei Teams ein, wobei jeweils ein Team für das Frontend oder das Backend zuständig war. Das Frontend umfasste hierbei das VR-Tischtennis-Spiel selbst und den Bot-Gegenspieler. Das Backend-Team befasste sich mit jeglichen Auswertungsmechanismen und dem Chatbot. Im Verlauf der Implementierung ergab sich eine Rolle, welche sowohl im Front- als auch im Backend

[1] Startseite Discord: `https://discord.com`, letzter Aufruf 27.09.2023

aktiv ist, indem sie sich mit der Bearbeitung der Schnittstelle zwischen beiden Enden beschäftigte.

Die Zuordnung der Projektteilnehmer zu den jeweiligen Teams sowie deren Rollen innerhalb des Projektes sind in Tabelle 2.1 zu sehen.

Teilnehmende	Rollen	
Studentin 0	Back- & Frontend	DevOps
Studentin 1	Backend	PyTorch Enthusiastin
Studentin 2	Backend	Dokumentation
Studentin 3	Frontend	Unity Enthusiastin
Student 0	Backend	Projektmanagement
Student 1	Frontend	DevOps
Student 2	Frontend	Projektmanagement
Student 3	Backend	Studienbeauftragter

Tabelle 2.1: Rollenverteilung und Teamzuordnung innerhalb der Projektgruppe

Geplant wurde das Vorgehen bei der Implementierung zunächst in Anlehnung an das Wasserfallmodell. Hierbei wurde die initiale Meilensteinplanung und Terminierung durch das Projektmanagement erstellt und den anderen Projektmitgliedern sowie dem Betreuer vorgelegt. Das daraus resultierende Feedback wurde aufgenommen, in den Meilensteinplan eingearbeitet und erneut besagten Personen vorgelegt. Im Laufe der Implementierung haben die Projekteilnehmer sich jedoch für ein agiles Modell, in Anlehnung an Scrum, entschieden. Die Begründung hierfür ist in Abschnitt 2.3 zu finden.

Während der Seminarphase wurde ein wöchentliches Gruppenmeeting abgehalten, um den aktuellen Stand der Gruppenmitglieder zu besprechen. Diesen Treffen wohnte alle zwei Wochen Herr Lüers bei, um Feedback zu geben und gegebenenfalls Fragen zu beantworten. Jedes dieser Meetings wurde durch ein wöchentlich wechselndes Gruppenmitglied protokolliert und durch das Projektmanagement moderiert.

Zu Beginn der Implementierungsphase wurde neben den wöchentlichen Gruppenmeetings noch ein wöchentliches Teammeeting, für jeweils Front- und Backend eingeführt. In diesen Meetings sollte der spezifische Stand des jeweiligen Endes besprochen, und über künftige Schritte diskutiert werden. Auch diesen Meetings wohnte Herr Lüers in zweiwöchentlichem Abstand bei, wobei er in jeder ungeraden Kalenderwoche am Gruppenmeeting und in jeder gerade Kalenderwoche an den beiden Teammeetings teilnahm. Die Rolle des Betreuers bei den Teammeetings umfasste die Beantwortung konkreter, technischer Fragen, Empfehlungen von bestimmten Methoden und gegebenenfalls Hilfestellung bei Problemen.

Der primäre Vorteil dieser Planung ist die wöchentliche Möglichkeit zur Rücksprache mit dem Betreuer, ohne auf digitale Kommunikationskanäle wie E-Mail oder Microsoft Teams zurückgreifen zu müssen.

Im Zeitraum der Nutzerstudie wurden jegliche Gruppenmeetings ausgesetzt, da sich jedes Gruppenmitglied, welches nicht aktiv Probanden betreute, sich auf die Doku-

mentation der Implementierung fokussieren sollte und hierfür regelmäßige Absprache als nicht nötig angesehen wurde.

Die Projektteilnehmer entschieden sich gegen die Nutzung eines Zeiterfassungssystems, da dies in den Erfahrungen der Teilnehmer in vorherigen Projekten tendenziell zu Konflikten innerhalb der Gruppen führte. Aufgaben- und zielorientiertes Arbeiten war hierbei für die Teilnehmer von Priorität und Zeitaufwand sollte keine Metrik für Fortschritt sein.

2.3 Kritik am Vorgehensmodell

Mit Hinblick auf das Vorgehensmodell bei der Implementierung entschieden sich die Projektteilnehmer zunächst für eine klassische Projektplanung nach dem Wasserfallmodell. Gründe hierfür waren zum einen die mangelnden Vorkenntnisse mancher Projektmitglieder mit agilen Vorgehensmodellen und zum anderen der von Beginn vorgegebene zeitliche und inhaltliche Rahmen des Projektes. Mit neuen Erkenntnissen während der Implementierungsphase wurden jedoch Probleme erkennbar. So wurde beispielsweise zum Zeitpunkt der initialen Planung, die Komplexität von Aufgaben, wie dem Aufsetzen des neuronalen Netzes zur Evaluation der Performance des Spielers, unterschätzt. Ebenso wurden während der Implementierungsphase bessere Lösungen für vorherige Probleme gefunden, wodurch größere Änderungen am Spiel vorgenommen werden mussten. Dies in Kombination mit dem Verlust des *GitLab*-Zugangs von fünf der acht Gruppenmitglieder führte zur Unbrauchbarkeit des zuvor angelegten Plans und der damit einhergehenden Anpassung des Vorgehensmodells.

Das Arbeiten mit *Feature Branches* führte zu Beginn der Implementierungsphase zu geringfügigen Problemen und verlangsamte die Implementierung etwas. Grund dafür war die nötige Einarbeitung in den Workflow, welcher nicht allen Projektteilnehmern geläufig war. Um hierbei Hilfestellung zu leisten, wurde gruppenintern ein Leitfaden, zu finden in Anhang A, angelegt, welcher die empfohlene Vorgehensweise beschrieb.

Je komplexer das Frontend des Projektes, also das VR-Tischtennis-Spiel, wurde, desto ineffektiver wurde die Arbeit mit *Feature Branches*. Das Lösen von Merge-Konflikten wurde zunehmend komplexer, zeitintensiver und somit zunehmend zu einer Fehlerquelle. Komplexer bedeutet in diesem Fall, dass die Anzahl der Code-Zeilen, welche Konflikte erzeugten, mehr wurden, wodurch die Konsequenzen von Änderungen an Unity-Szenen nicht direkt ersichtlich waren. Bei umfangreicheren Änderungen, somit auch umfangreicheren Branches und Merge-Konflikten, konnte es also häufig zu Fehlern kommen. Ein Ansatz um diesem Problem vorzubeugen, wäre es gewesen, den Umfang von einzelnen zu mergenden Änderungen minimal zu halten, jedoch war dies bei umfangreicheren Features und Mechaniken, so wie dem `GameStateTracker` nicht möglich.

Der Verlust der DFKI-*GitLab*-Zugänge, führte über die zuvor genannten Probleme hinaus ebenfalls dazu, dass das Frontend-Projekt auf die *GitLab*-Plattform der Universität transferiert werden musste. Dies machte die Aufgabenorganisation zwischen Front- und Backend oft unnötig aufwändig. Insbesondere in der zweiten Hälfte der Implementierungsphase wurde die Verwaltung von Issues und Meilensteinen auf *GitLab* weitestgehend vernachlässigt und Absprachen bezüglich Dokumentationsaufgaben und Zuständigkeiten während der Nutzerstudie erfolgten vorwiegend über entsprechende *Discord-Channels*. Durch die mehrfachen Anpassungen des Vorgehensmodells

und der übergreifenden Projektplanung, sowie der Verteilung der Nutzer über zwei separate *GitLab*-Plattformen, wurden die ursprünglich definierten Meilensteine verworfen und eine Neudefinition als nicht nötig erfunden. Zum einen, da gruppenintern angezweifelt wurde, ob eine effektive Definition der Meilensteine mit den vorliegenden Vorkenntnissen möglich sei, und zum anderen hätte aus Sicht der Projektteilnehmer der geringe Mehrwert dieser den hohen Aufwand, dies über zwei separate *GitLab*-Plattformen zu tun, nicht gerechtfertigt. Issues wurden von den jeweiligen Teams (hier: Frontend und Backend) zur integrierten Erstellung der jeweiligen *Feature Branches* genutzt, jedoch nicht zur tatsächlichen Aufgabenverteilung. Über den Projektverlauf ergaben sich klare Zuständigkeiten von einzelnen Projektteilnehmern zu den jeweiligen Komponenten des Projektes. Daher wurde übergreifende Aufgabenverteilung als wenig effektiv angesehen, womit die Aufgabenverteilung über Issues an Wert verlor.

Der Verlust der Zugänge ereignete sich aufgrund eines fehlenden Einverständnisses der Datenschutzerklärung. Besagte Datenschutzerklärung wird erst angezeigt, sobald ein erster Login in das DFKI-Intranet erfolgt, wofür die Verbindung über ein VPN nötig ist. Diese Verbindung konnte nicht von allen Projektteilnehmern erfolgreich aufgesetzt werden, sei es durch nicht identifizierbare Probleme oder unvollständige initiale Zugangsdaten, womit auch die Einverständniserklärung ausblieb. Die Infrastrukturgruppe (ISG) des DFKI übermittelte den Projektteilnehmern eine Erinnerung, diese Einverständniserklärung abzugeben, jedoch erfolgte dies an die DFKI-Mailadressen der Teilnehmer, welche nicht aktiv genutzt wurden. Nachdem die Einverständniserklärung über mehr als 6 Monate nicht gegeben wurde, wurden die Accounts durch die ISG gelöscht. Das Problem bei der Kommunikation der Erinnerung konnte identifiziert und kommuniziert werden, weshalb das DFKI zukünftig Erinnerungen auch an externe E-Mail-Adressen, wie die Universitäts-E-Mail der Projektteilnehmer, schickt, um Wiederholungen dieses Vorfalls zukünftig zu vermeiden.

Kapitel 3

Grundlagen und Begriffsklärung

In diesem Kapitel werden dieses Projekt relevante, theoretische Grundlagen aufbereitet. Darunter sowohl wissenschaftliche als auch technische Grundlagen.

3.1 Wissenschaftliche Grundlagen

Im Folgenden werden die wissenschaftlichen Grundlagen zu *Learning Analytics*, *Serious Gaming*, *Machine Learning* Methoden, *Datengeneratoren*, *Chatbots* und die grundlegenden Tischtennisregeln dargelegt.

3.1.1 Einführung und Abgrenzung von Learning Analytics und Multimodal Learning Analytics[1]

Learning Analytics (LA) beschreibt den ganzheitlichen Prozess der Sammlung, Analyse und Darstellung von Daten in Bezug auf die Verhaltens- und Aktivitätsmuster eines Lernenden. Ziel von LA ist das verbesserte Verständnis und die Ermöglichung besserer Unterstützung während des Lernprozesses. Ein iteratives Vorgehensmodell wurde bereits 2012 von Doug Clow in [Clow, 2012] veröffentlicht: der *Learning Analytics Cycle*.

Abbildung 3.1 zeigt eine grafische Übersicht der Schritte des *Learning Analytics Cycle* nach Clow. Hierbei beginnt der Zyklus mit den Lernenden beziehungsweise *learners* und deren Lernprozess. Darauf folgt die Erzeugung und Sammlung der Daten mithilfe der umgebungsspezifischen Eingabemodalitäten. Der dritte Schritt *metrics* umfasst nicht nur Auswertung und Analyse der Daten anhand zuvor bestimmter Metriken, sondern auch die Darstellung der Daten wie zum Beispiel in Form eines Dashboards. Der vierte Schritt *interventions* beschreibt Maßnahmen, die Einfluss auf den Lernprozess nehmen und diesen in der Regel verbessern sollen. Hierbei fügt Clow hinzu, dass LA nicht immer alle vier Schritte beinhalten, so kann beispielsweise der vierte Schritt

[1] Dieser Abschnitt ist in Teilen aus dem Review-Papier „Multimodal Learning Analytics und Learning Analytics im Rahmen des Serious Gaming“ übernommen worden.

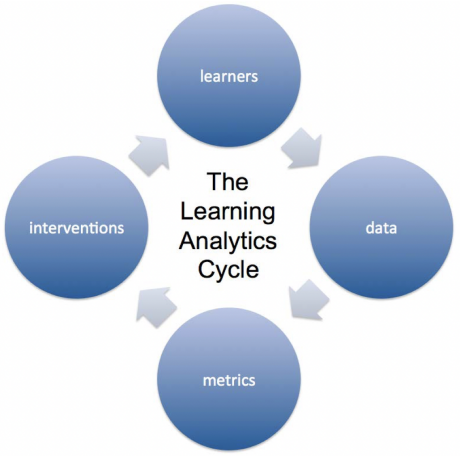


Abbildung 3.1: Learning Analytics Cycle nach Clow aus [Clow, 2012]

fehlen und der Gesamtprozess fällt nach wie vor unter die Definition LA, wobei er jedoch ebenfalls erwähnt, dass dies kein effektives Projekt darstelle. [Clow, 2012]

[Oviatt et al., 2018] beschreiben Multimodal Learning Analytics (MLA) als ein noch emergentes Feld, welches sich auf natürliche Kommunikations-, Handlungs-, sowie physiologische und neuronale Muster eines Lernenden spezialisiert. Mithilfe dieser Spezialisierung seien Methoden der MLA fähig, das Lernverhalten Lernender sowie deren emotionalen Zustand vorherzusagen. [Oviatt et al., 2018]

LA und MLA unterscheiden sich in Hinblick auf die Menge der gesammelten Daten, dem Setting der Praktizierung, der Lernumgebung und den Ursprung der gesammelten Daten. Traditionelle LA Methoden sind auf einen Lerner ausgelegt, der über Tastatur und Maus mit dem Lehrsystem interagiert. Somit handelt es sich bei den generierten Daten rein um Click-Stream-Daten aus einem stationären Setting, die nur dann generiert werden, wenn der Lernende explizit mit dem System interagiert. Die Nutzung mobiler Endgeräte und alternativer Sensoren, wie Mikrofone und Kameras, ermöglichen der MLA hingegen ein mobiles Setting und die Aufzeichnung des Verhaltens während interpersonellen Lernens. Das Ergebnis ist ein umfangreicher, synchronisierter Stream aus verschiedenen Datentypen, die sowohl aktive als auch passive Lernverhalten abbilden können. [Oviatt et al., 2018]

3.1.2 Grundlagen des Serious Gaming[2]

Der Begriff des *Serious Game* (SG) beschreibt jegliche Art von Spiel, deren primärer Zweck nicht die Unterhaltung des Spielers ist. Stattdessen haben sie meist lehrende oder anleitende Funktionen [Alvarez et al., 2011]. Die Prozesse der Extraktion und Verarbeitung von Daten, die in einer solchen Spielumgebung entstehen, werden auch als *Game Analytics* bezeichnet, während das interdisziplinäre Feld aus GL und *Learning Analytics* (LA) als *Game Learning Analytics* (GLA) bekannt ist [Freire et al., 2016].

Einer der einzigartigen Vorteile, den SG im Feld der GLA mit sich bringen, hängt direkt

[2]Dieser Abschnitt wurde größtenteils aus dem Grundlagenkapitel des Proposals der PGMMI2 übernommen und sinngemäß übersetzt.

mit dem Design jeglicher Art Spiel zusammen. Jedes Spiel ist um einen kurzen Loop zwischen Feedback und Interaktion mit der Spielumgebung konstruiert, wodurch eine hohe Menge Interaktionsdaten generiert wird. Dies bringt ebenfalls einzigartige Herausforderungen mit sich: darunter ein hohes Maß an Komplexität beim Design und Betrieb der Daten-Pipeline, was eine Herausforderung ist, die über den gesamten Designprozess des GLA-Systems berücksichtigt werden muss. Eine weitere Komplexitätsebene entsteht durch den Anwendungsfall des Systems. Dieser definiert, wer das System nutzt und somit die vom System generierten Informationen bewertet und Handlungen aus diesen ableitet. So muss beispielsweise in einem Lehrkontext, wie einem Klassenzimmer, die Erfahrung des Lehrenden mit dem System und dessen technisches Know-how berücksichtigt werden. Diese aktuell noch mangelnde Einbringung von Stakeholdern in den Designprozess gilt als einer der am stärksten limitierenden Faktoren bei der Verbreitung von GLA-Systemen. [Petrov et al., 2018]

3.1.3 Grundlagen von Machine-Learning-Methoden

Eine ML-Methode zeichnet sich dadurch aus, dass aus den vorliegenden Daten in Bezug auf eine gegebene Aufgabe gelernt wird. Wird beispielsweise aus den Erfahrungen bei einer bestimmten Aufgabe gelernt, verbessert sich die Performance in der Aufgabe. Es gibt u.A. verschiedene Arten von Aufgaben: Klassifikation, Klassifikation ohne Eingabe, Regression, Transkription, Translation, Aufgaben mit strukturierter Ausgabe, Anomaliedetektion, Synthese und Sampling, Imputation von fehlenden Werten, Rauschunterdrückung und Schätzung der Dichte bzw. Wahrscheinlichkeitsfunktion. [Bengio et al., 2017]

Im Folgenden wird auf den hier relevanten Aufgabentyp, die *Klassifikation*, eingegangen: Bei der Klassifikation ist die Aufgabe zu bestimmen, zu welcher der Kategorien bzw. Klassen eine Eingabe gehört. Dazu wird im Lernalgorithmus eine Funktion verwendet, die für eine Eingabe, auch Features genannt, je nach Klassifikationsaufgabe eine Abschätzung oder eine Wahrscheinlichkeitsverteilung ausgibt, wie sehr die Eingabe zu einer bestimmten Klasse passt. Die ausgegebene Klasse wird auch als Label bezeichnet. [Bengio et al., 2017] Um die Performance des Lernalgorithmus zu bestimmen, gibt es verschiedene Metriken, beispielsweise die Genauigkeit des Modells. Die Genauigkeit (*Accuracy*) wird beispielsweise für Klassifikationsaufgaben verwendet. Mithilfe der Genauigkeit wird der Anteil der korrekten Klassifikationen von allen Klassifikationen angegeben: [Bengio et al., 2017]

$$\text{Genauigkeit} = \frac{\text{Anzahl der korrekten Klassifikationen}}{\text{Anzahl aller Klassifikationen}} \tag{3.1}$$

Bei einem binären Entscheidungsproblem werden Eingaben entweder als positiv oder negativ klassifiziert. Dies lässt sich in einer *Confusion Matrix* darstellen. Dabei existieren vier Kategorien: Korrekt als positiv gelabelte Eingaben, *True Positives* (TP), falsch als positiv gelabelte Eingaben, *False Positives* (FP), korrekt als negativ gelabelte Eingaben, *True Negatives*, und falsch als negativ gelabelte Eingaben, *False Negatives* (FN). Daraus ergeben sich die folgenden Metriken zur Evaluation, Precision und Recall: [Goutte and Gaussier, 2005]

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}$$

Dementsprechend kann mithilfe der Genauigkeit eine Aussage darüber getätigt werden, wie oft die Klassifikationen des Modells korrekt sind. Im Gegensatz dazu können mithilfe von Precision und Recall genauere Aussagen über die Performance des Modells in Bezug auf bestimme Klassen gemacht werden: Precision gibt an, wie oft das Modell korrekte Klassifikationen einer Klasse getätigt hat. Recall zeigt, ob das Modell alle Objekte der betrachteten Klasse klassifizieren konnte. [3] Interessant ist außerdem zu untersuchen, wie der Lernalgorithmus mit unbekannten Daten funktioniert, daher wird üblicherweise die Genauigkeit für ein Testset an unbekannten Daten bestimmt. [Bengio et al., 2017]

ML-Methoden werden in *überwachte* und *unüberwachte* Methoden eingeteilt. Beim überwachten Lernen sind im Datensatz Datenpunkte aus einer Eingabe und einer zugehörigen Ausgabe gegeben, während die zugehörige Ausgabe beim unüberwachten Lernen nicht gegeben ist. Hier werden nützliche Eigenschaften bzw. Muster aus den Daten gelernt und so die Eingaben beispielsweise anhand der gelernten Eigenschaften zu clustern. [Bengio et al., 2017] Ein Beispiel fürs überwachte Lernen ist das Training anhand eines vorgegebenen Datensatzes, in dem die Daten den einzelnen Klassen zugeordnet sind, um im Nachhinein ungesehene Daten in die entsprechenden Klassen klassifizieren zu können. Beim unüberwachten Lernen ist eine klassische Aufgabe, die beste Repräsentation der Daten zu finden. [Bengio et al., 2017]

Ziel von der Anwendung von ML-Methoden ist, dass das Modell auf unbekannten Eingaben, die nicht Teil der Trainingsdaten sind, gut abschneidet. Dies wird als *Generalisierung* bezeichnet. [Bengio et al., 2017]

Mögliche Probleme von ML-Methoden sind *Overfitting* und *Underfitting*. *Underfitting* liegt vor, wenn ein großer *Bias*, also eine große Verzerrung, vorliegt und das Modell keinen kleinen Fehlerwert beim Training erreichen kann. Das kann daran liegen, dass das Training zu frühzeitig beendet wurde oder das Modell eine zu geringe Komplexität aufweist, um das Muster in den Daten zu lernen. Um *Underfitting* zu lösen, kann das Training verlängert oder das Model komplexer gestaltet werden. Ebenfalls können weitere Features zu den Daten hinzugefügt werden. [Vandeput, 2021] Beim Overfitting hingegen ist der Abstand zwischen Test- und Trainingsfehler zu groß [Bengio et al., 2017]. Um *Overfitting* zu vermeiden, werden Regulierungsmethoden eingesetzt. Ein Beispiel für neuronale Netze ist die L1- und L2-Regulierung, bei denen die Gewichte der Netze reduziert werden, um diese einfacher zu halten und diese somit zu generalisieren. [Moore and DeNero, 2011]

Um das Verhalten von Lernalgorithmen zu beeinflussen, werden *Hyperparameter* verwendet. Diese werden vor dem Training gesetzt und nicht direkt durch den Lernalgorithmus selbst angepasst. Je nach Lernalgorithmus gibt es verschiedene Hyperparameter, auf die im Folgenden noch weiter eingegangen werden soll. [Bengio et al., 2017]

Deep Learning gehört zu den ML-Methoden [Bengio et al., 2017], darunter fallen die hier relevanten *neuronalen Netze CNN*, *RNN* und *LSTM*, die im folgenden genauer beschrieben werden:

[3] Accuracy vs. precision vs. recall in machine learning: what's the difference?: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#:~:text=Accuracy%20shows%20how%20often%20a,objects%20of%20the%20target%20class., letzter Aufruf 28.09.2023

Neuronale Netze

Künstliche neuronale Netze (KNN) zeichnen sich hinsichtlich ihrer Architektur dadurch aus, dass sie gewichtete, gerichtete Graphen sind, deren Knoten Neuronen und die Verbindungen gerichtete, gewichtete Kanten sind. Außerdem gibt es Eingabe- und Ausgabeneuronen. Ein Neuron berechnet die gewichtete Summe der eingehenden Signale und gibt je nach *Aktivierungsfunktion* einen Wert aus. Bei der Aktivierungsfunktion *ReLU*, die vermehrt im maschinellen Lernen verwendet wird, wird beispielsweise bei negativen Werten eine 0 ausgegeben und bei positiven Werten werden linear die positiven Werte ausgegeben. [Sonnet, 2022] Durch *ReLU* kommt es zu "toten Neuronen", wenn diese 0 ausgeben, weil es durch kleinere Eingabeänderungen selten zur Änderung der Ausgabe kommt. *LeakyReLU* umgeht dieses Problem: Bei positiven Werten funktioniert *LeakyReLU* genauso wie *ReLU*. Bei negativen Werten wird der negative Wert multipliziert mit einem Parameter ausgegeben. [Masetti and Di Giandomenico, 2020] Dieser Parameter ist bei PyTorch bspw. mit 0.01 festgelegt und kann beliebig angepasst werden[4]. Bei der *Aktivierungsfunktion tanh* wird die Ausgabe, wie in Gleichung 3.4 von [Sonnet, 2022] dargestellt, berechnet: Bei der *Aktivierungsfunktion* wird der *Tangens hyperbolicus* von x abgebildet. Dabei konvergiert die Funktion bei positiven Werten gegen 1 und bei negativen Werten gegen -1.

$$f(net_j) = tanh(net_j) = 1 - \frac{2}{e^{2*net_j} + 1} \tag{3.4}$$

Für neuronale Netze, deren Ziel die Klassifizierung ist, wird in der Ausgabeschicht oftmals die Aktivierungsfunktion *Softmax* verwendet. Bei *Softmax* geben die Neuronen der Ausgabeschicht einen Wert zwischen 0 und 1 aus, egal wie groß der Eingabewert ist. Die Werte aller Neuronen der Ausgabeschicht ergeben durch Normierung in Summe 1. Jedes Neuron soll dabei eine Klasse repräsentieren und der jeweilige Ausgabewert kann als jeweilige Wahrscheinlichkeit für die Klasse interpretiert werden. Der Input wird dann derjenigen Klasse zugeordnet, dessen Neuron die höchste Wahrscheinlichkeit aufweist. [Werner, 2020]

Für diese Arbeit sind zwei Netzwerkarten relevant: *Feed-Forward Netzwerke*, in denen keine Schleifen vorkommen, oder *Rekurrente neuronale Netze* (RNN), die durch eine Feedbackschleife charakterisierbar sind. Die *Feed-Forwand Netzwerke* werden alternativ auch *Multi-Layer Perzeptron* genannt, in dem die Neuronen in Schichten mit unidirektionalen Verbindungen dazwischen angeordnet sind. Diese werden auch als *Dense-Layer* bezeichnet. Anzumerken ist, dass ihre Ausgabe zu einer Eingabe nicht von der zeitlich vorigen Eingabe abhängt. Anders ist dies bei *RNN*s: Hier dienen die Rückkopplungsschleifen dazu, dass vorige Ergebnisse bei der aktuellen Eingabe berücksichtigt werden. [Jain et al., 1996]

Der Lernprozess eines *KNN*s besteht darin, die Architektur des Netzes und die Gewichte der Kanten so zu aktualisieren, dass die Performance für die bestehende Aufgabe verbessert wird. [Jain et al., 1996] Wie das Training genau aussieht, wird im Folgenden genauer beschrieben.

Ein *Convolutional Neural Network* (CNN) ist eine Art von tiefen neuronalen Netzwerken, in der beim Verarbeiten der Eingabedaten eine Faltung stattfindet. Die Faltungs-

[4] PyTorch-Dokumentation LeakyReLU: `https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html`, letzter Aufruf: 27.09.2023

operation (Convolution) beschreibt, in welcher Art und Weise das Netzwerk Informationen aus den Daten extrahiert und verarbeitet. Die Faltungsoperation ermöglicht es, die Ähnlichkeit oder Unterschiedlichkeit von Merkmalen an verschiedenen Positionen der Eingabe zu analysieren. Die erste der beiden gefalteten Signale ist die Eingabe, während das zweite Signal der *Kernel* ist, der der Durchführung der Faltung nützt. Der *Kernel* hat eine spezifische Größe, die je nach Dimension und Größe der Eingabedaten variieren kann. Er durchläuft während der Faltung die Features der Eingabe und bildet den gewichteten Durchschnitt aus allen Features, die er mit seiner Größe umfasst. Mit der Einführung des *Strides* kann der Kernel Daten überspringen. Mit dem *Zero-Padding* werden die Daten mit 0 erweitert, um ein neues Datenformat zu erhalten. Die Daten der Größe $9 * 9$ können dann beispielsweise so erweitert werden, dass sie durch Hinzufügen von Nullen auf eine Größe von $10 * 10$ verändert werden. [Vakalopoulou et al., 2023] Bei Daten, die dreidimensional aufgebaut sind, werden bei Python für eine Dimension *Channel* eingesetzt [5]. *Pooling Operationen* sind dafür da, die Größe der Feature Maps zu verringern und dadurch den Rechenaufwand zu verkleinern [Vakalopoulou et al., 2023].

Abbildung 3.2 aus [Vakalopoulou et al., 2023] zeigt eine beispielhafte Architektur von *CNNs*. Diese besteht hauptsächlich aus dem Teil zum Lernen der Merkmale und dem Klassifikationsteil. Das Lernen der Merkmale basiert auf den Faltungsoperationen, nicht-linearen Aktivierungsfunktionen, wie der *ReLU*-Funktion, und einer Pooling-Operation, während die gelernten Merkmale im Klassifikationsteil geglättet und dann in ein Netz aus komplett vernetzten Schichten, beispielsweise in ein *Multilayer Perzeptron*, gegeben wird. Dort findet dann die Klassifikation statt. [Vakalopoulou et al., 2023]

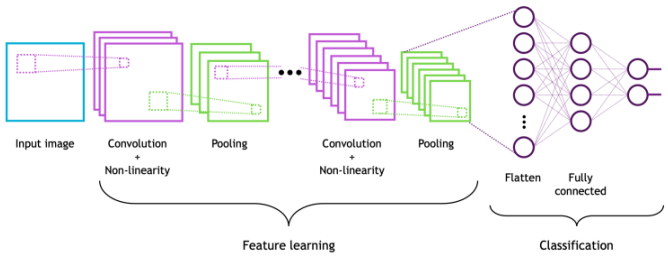


Abbildung 3.2: Beispielhafte Architektur eines CNNs aus [Vakalopoulou et al., 2023]

Rekurrente neuronale Netze (RNN) sind eine Art von KNNs, die besonders für sequentielle Eingabedaten geeignet sind. Dabei können die Eingabe und Ausgabe hinsichtlich ihrer Länge variabel sein. Besonders ist hier, dass nicht nur die aktuelle Eingabe, sondern auch vorige Eingaben berücksichtigt werden. [Bengio et al., 2017] Um dies zu erreichen, werden Feedbackschleifen verwendet [Jain et al., 1996], die zwischen Knoten und Schichten bestehen und Eingabesequenzen von beliebiger Länge verarbeiten können. Die Algorithmen zum Aktualisieren der Gewichte basieren hauptsächlich auf Gradienten. Hier können Probleme wie *vanishing* oder *exploding* Gradienten auftreten. [Smagulova and James, 2019] Das Problem der *exploding* Gradienten liegt vor, wenn die Norm des Gradienten während des Trainings stark ansteigen. Beim Problem der *vanishing* Gradienten wird die Norm des Gradienten während der Backpro-

[5] PyTorch-Dokumentation Conv3d: `https://pytorch.org/docs/stable/generated/torch.nn.Conv3d.html`, letzter Aufruf: 21.09.2023

pagation verschwindend klein, was es dem Modell schwierig macht, die Gewichte zu aktualisieren. [Pascanu et al., 2013] [Bengio et al., 1994]

Die Historie der Sequenz wird in einem rekurrenten, versteckten Vektor zwischengespeichert, der vom vorigen versteckten Vektor abhängt. [Tan et al., 2015] Abbildung 3.3 aus [Bengio et al., 2017] zeigt eine beispielhafte Architektur eines RNNs.

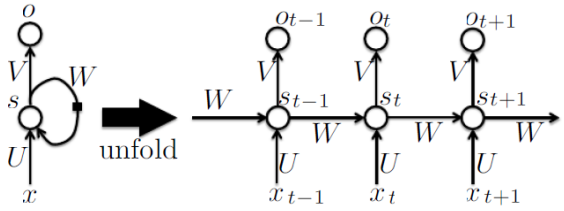


Abbildung 3.3: Beispielhafte Architektur eines RNNs aus [Bengio et al., 2017]. Links ist der rekurrente Netzkreislauf mit versteckter Rekursion zu sehen. Rechts ist die Architektur als Flussgraph dargestellt, der nicht zeitlich gefaltet ist. Dort ist jeder Knoten mit einer spezifischen Zeitinstanz verbunden. [Bengio et al., 2017]

Long Short-Term Memory (LSTM) ist eine Art von RNNs, die das Problem der verschwindenden Gradienten von RNNs behebt. [Tan et al., 2015] Um dies zu erreichen, verfügen LSTMs über einen Speicher und multiplikative Gatter.
[Smagulova and James, 2019]

Abbildung 3.4 aus [Van Houdt et al., 2020] zeigt eine Architektur eines LSTM-Blocks, der gebräuchlichsten Variante eines LSTM-Blocks. Insgesamt besteht ein LSTM aus mehreren rekurrent vernetzten Netzwerken, den Speicherblöcken. Ein einzelner Speicherblock, ein LSTM-Block, besteht aus einer Zelle, einem Eingabegatter, einem Ausgabegatter und einem *forget*-Gatter. Die Zelle ist dafür zuständig, Werte über beliebige Zeitintervalle zu speichern, während die Gatter für die Regulation des Informationsflusses zuständig sind. Das *forget*-Gatter sorgt für die Bestimmung, welche Information aus dem vorigen Zellzustand vergessen werden soll. [Van Houdt et al., 2020]

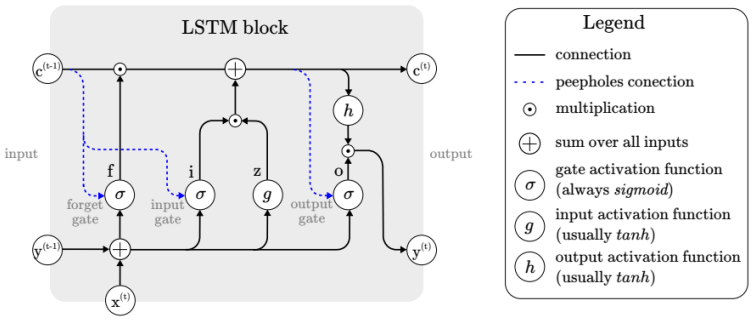


Abbildung 3.4: Beispielhafte Architektur eines LSTM-Blocks aus [Van Houdt et al., 2020].

Training der neuronalen Netze und dessen Hyperparameter

Wie bereits in der Einführung dieses Abschnittes erläutert, können verschiedene Hyperparameter im Training eingesetzt werden, um dieses zu verbessern. Ein neuronales Netz „lernt“, indem die Gewichte der einzelnen Neuronen im Training angepasst werden. Das geschieht, indem die sogenannte *Backpropagation* während des Trainings in jeder Epoche durchgeführt wird. Während einer Epoche werden zunächst die Features in das Netz gegeben. Bei der Klassifizierung berechnet das Netz dann die entsprechende Klasse. Danach wird eine sogenannte *Loss-Funktion* auf der vom Netz berechneten Klasse und der tatsächlichen Klasse berechnet, um den Fehler zu bestimmen. Ein Beispiel einer Loss-Funktion für die Klassifikation ist die sogenannte *Cross-Entropy* oder auch *Kreuzentropie*, bei der die Logarithmen der Wahrscheinlichkeiten der korrekt klassifizierten Klassen aufsummiert wird. Je kleiner die berechnete Wahrscheinlichkeit der korrekt klassifizierten Klasse ist, desto höher ist der Fehlerwert. [Werner, 2020]

Nach Berechnung des Fehlerwertes wird der Fehler zurück durch das Netzwerk propagiert. Dabei wird auf der Fehlerfunktion ein *Gradientenabstieg* durchgeführt, um die Gewichte der Neuronen dementsprechend anzupassen. Beim Gradientenabstieg wird dabei versucht, das globale Minimum in der Fehlerlandschaft zu erreichen, wozu *Optimierer* und *Lernraten* verwendet werden. *Optimierer* sind dazu da, dass lokale Minima in der Fehlerlandschaft übergangen werden können. Als Schrittweite, wie weit die *Optimierer* innerhalb des *Gradientenabstiegs* gehen, wird die *Lernrate* bezeichnet. Ist diese zu hoch gewählt, kann es dazu kommen, dass globale Minima übersprungen werden. Bei einer zu niedrigen *Lernrate* kann es zu einem längeren Training kommen. Beispiele für *Optimierer* sind *SGD mit Momentum* oder *Adam*. Der *Optimierer SGD mit Momentum* betrachtet beim *Gradientenabstieg* den Durchschnitt des Gradienten über die Zeit, um das Oszillieren um globale Minima zu vermeiden. Beim *Adam* kann zusätzlich die Lernrate über die Zeit adaptiert werden, indem vorherige Ereignisse in Betracht gezogen werden. Damit kann der Optimierer schneller das Optimum erreichen. [Günther and Fritsch, 2010], [Haji and Abdulazeez, 2021], [Bengio et al., 2017]

3.1.4 Grundlagen für Datengeneratoren

Das Training von künstlichen neuronalen Netzen erfordert eine große Menge an Daten. Je variabler die Daten sind, desto besser kann die KI lernen, da die KI durch die variablen Daten robuster wird und in kritischen Situationen wie beispielsweise Unfällen bessere Entscheidungen treffen kann. Jedoch scheitern vermehrt Anwendungen, da unzureichend Daten „wegen einer zu teuren Datenerfassung oder aus Datenschutzgründen“ [Hecker et al., 2023] zur Verfügung stehen. Aus diesem Grund werden vermehrt generierte Daten, auch als *synthetische Daten* bezeichnet, eingesetzt. Für die Generierung von Daten werden Datengeneratoren, die auf künstlicher Intelligenz basieren, verwendet. Datengeneratoren haben den Vorteil, dass sie eine große Variation von Daten erstellen und beliebig viele Daten erzeugen können. Ebenfalls können mit Datengeneratoren Trainingsdaten gezielt erweitert werden, um einen ausbalancierten Datensatz für das Training bereitzustellen. Die Daten bleiben dabei anonym, sodass Probleme mit dem Datenschutz vermieden werden können. [Hecker et al., 2023]

Beispiele für Datengeneratoren sind die Python-Pakete *synloc* und *ctgan*. Synloc implementiert ein Paper zur synthetischen Datengenerierung über den *k-nearest-neighbor-Algorithmus* [Kalay, 2022]. Der Local Resampler, der in dem Python-Paket zur Gene-

rierung genutzt wird, berechnet zu gegebenen Daten zu jedem Datensatz die k nächsten Nachbarn und speichert diese in einer Liste. Bei der Generierung der synthetischen Daten wird aus der Liste ein zufälliger Datenpunkt ausgewählt. [Kalay, 2022]

Bei dem Python-Paket ctgan[6] wird die synthetische Datengenerierung mittels neuronaler Netze durchgeführt, die versuchen den Aufbau der Daten zu lernen, um darauf basierend synthetische Daten zu generieren [Xu et al., 2019].

3.1.5 Einführung in Chatbots und Dialogsysteme[7]

Es ist grundlegend zwischen *Dialogsystemen* und *Chatbots* zu unterscheiden. Dialogsysteme, auch *conversational agents* genannt, zeichnen sich dadurch aus, dass sie eine Unterhaltung mit Nutzern ermöglichen. Diese Unterhaltung kann in Form von natürlicher Sprache, also Text oder Sprache, stattfinden. [Jurafsky and Martin, 2000] Drei wichtige Komponenten von Dialogsystemen sind **Sprachverständnis**, **Dialogmanagement** und **Spracherzeugung**. [Su et al., 2018]

Es gibt zwei Arten von Dialogsystemen: **aufgabenorientierte Dialogagenten** und **Chatbots**. Aufgabenorientierte Dialogagenten geben dem Nutzer in der jeweiligen Konversation Hilfestellungen bei bestimmten Aufgaben. [Jurafsky and Martin, 2000]

Grundlegend basieren Dialogagenten auf *Natural Language Understanding* (NLU), einem Zustandstracker, einer Dialogpolitik (DP) und einem Generator natürlicher Sprache (NLG). Mithilfe des NLU kann die Absicht aus Äußerungen des Nutzers identifiziert werden und der Zustandstracker dient zum Verfolgen des Zustands der Konversation. Die DP wählt die nächste Aktion aus, auf Grundlage des aktuellen Zustands. Der NLG ist dafür zuständig, die Aktion des Agenten in eine Antwort natürlicher Sprache umzuwandeln. [Gao et al., 2019]

Die Bestandteile von intelligenten Benutzerschnittstellen, wie in Abbildung 3.5 dargestellt, sind die Eingabeverarbeitung, Medienanalyse, Interaction Management, Output Rendering, Media Design und Backend Services. Komponenten der Eingabeverarbeitung und der Medienanalyse sorgen dafür, dass die Eingabe des Nutzers analysiert und verstanden wird. Dabei können unterschiedliche Modalitäten betrachtet und miteinander kombiniert werden. So kann eine NLU-Komponente verwendet werden, um semantische Information aus der Nutzereingabe zu gewinnen, die Nutzerabsicht zu klassifizieren und Entitäten zu extrahieren. Dafür kann beispielsweise die RASA_NLU verwendet werden. [Zacharias et al., 2018] Zusammengefasst zeichnen sie sich dadurch aus, dass sie die Nutzereingabe verarbeiten und im Backend je nach Anwendungskontext analysieren. [Zacharias et al., 2018]

Das Interaktionsmanagement, die Ausgabedarstellung und die Mediengestaltung werden für ein zentrales Dialogmanagement genutzt und ermöglichen, dass zukünftige Aktionen auf Basis des aktuellen Zustands der Nutzereingabe generiert werden können. Dabei kann die Ausgabegenerierung mit NLG stattfinden. Ein Beispiel für das Dialogmanagement ist RASA_CORE. In den Backendservices können beispielsweise Daten analysiert werden oder Klassifizierungen mit neuronalen Netzen stattfinden. [Zacharias et al., 2018]

[6] CTGAN-Repository: `https://github.com/sdv-dev/CTGAN`, letzter Aufruf 26.06.2023

[7] Dieser Abschnitt wurde in Teilen aus dem Reviewpaper „AI-basierte Dialogsysteme und ChatBots“ übernommen

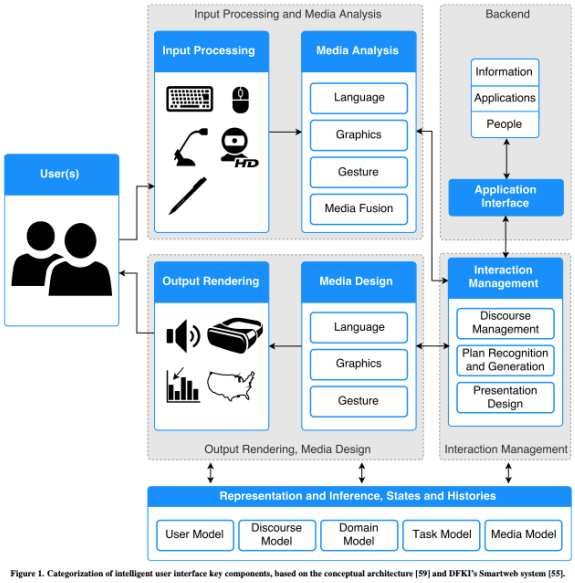


Figure 1. Categorization of intelligent user interface key components, based on the conceptual architecture [59] and DFKI's Smartweb system [55].

Abbildung 3.5: Systemarchitektur mit Komponenten von intelligenten Benutzerschnittstellen. Dazu zählen auch Dialogsysteme. [Zacharias et al., 2018]. Bildquelle: [Zacharias et al., 2018]

Intelligente Dialogsysteme können die Eingabe des Nutzers interpretieren. Dafür gibt es verschiedene Möglichkeiten wie Machine Learning (ML), Natural Language Processing (NLP), Neural Networks (NN) und Data-Mining. [Jadeja and Varia, 2017]

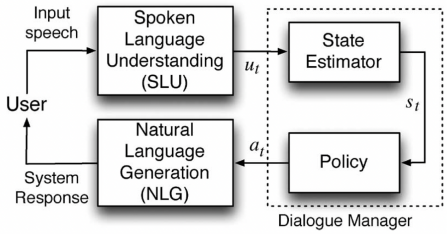


Abbildung 3.6: Komponenten von zustandsbasierten Dialogsystemen. Jede Nutzereingabe wird in eine abstrakte Repräsentation der Absicht des Nutzers umgewandelt. Der Zustand wird entsprechend der Absicht aktualisiert und auf Grundlage der Dialogpolitik wird dieser Zustand in eine Aktion bzw. Antwort des Systems überführt [Young et al., 2013]. Bildquelle: [Young et al., 2013]

Chatbots hingegen sollen eine Konversation mit einem menschlichen Nutzer ermöglichen, die der Konversation zwischen zwei Menschen möglichst ähnlich ist. Anwendungen sind hier, wie bei Dialogsystemen, eine bestimmte Aufgabe zu verfolgen, Unterhaltungszwecke [Jurafsky and Martin, 2000] oder die Bereitstellung von Informationen [Jadeja and Varia, 2017]. Auch Chatbots können auf NLP, ML und NN basieren, denn sie sind eine Art von Dialogsystemen. [Jurafsky and Martin, 2000]

Chatbots basieren entweder auf einem bestimmten Wissensbereich, auf den sie trainiert sind oder auf dessen Daten sie Zugriff haben (**geschlossenes Wissensgebiet**), oder sie können allgemeine Fragen beantworten (**offenes Wissensgebiet**).
[Adamopoulou and Moussiades, 2020] nach [Nimavat and Champaneria, 2017]

Weiterhin können Chatbots entweder **interpersonell** oder **intrapersonell** sein: interpersonelle Chatbots sind nur auf eine bestimmte Domäne spezialisiert, in der seine Aufgabe ausgeführt werden kann, nicht aber auf ein persönliches Gespräch mit dem Nutzer. Intrapersonelle Chatbots sind auf die Domäne des Nutzers spezialisiert und sollen ein gewisses Verständnis des Nutzers aufweisen.
[Adamopoulou and Moussiades, 2020] nach [Nimavat and Champaneria, 2017]

3.1.6 Grundlegende Tischtennisregeln

Tischtennis ist eine Rückschlagsportart, bei dem (im Einzel) zwei Spieler an einer Tischtennis-Platte in einer Sporthalle oder im Freien abwechselnd jeweils mit ihrem Tischtennisschläger den Ball übers Netz spielen. Hier ist die Hand-Auge-Koordination der Spieler wichtig. Besonders ist die hohe Geschwindigkeit der Bälle, die schnelle Reaktionen seitens der Spieler erfordert. Dies ist vergleichbar zu Sportarten wie Badminton, Tennis und Squash, bei denen das Spielgerät ebenfalls schnell hin- und hergespielt wird. Unterschiedlich ist das Spielgerät, in Anbetracht seiner Größe und Art, und die Größe des Spielfelds. In allen drei genannten Sportarten wird ebenfalls ein Schläger verwendet, der je nach Spielgerät unterschiedlich groß ist.

Die offiziellen Tischtennis-Regeln[8] lauten wie folgt:
Ein Tischtennisspiel wird so lange gespielt, bis ein Spieler eine vorher festgelegte Anzahl Sätze gewonnen hat. Im Regelfall handelt es sich bei der Anzahl der zu gewinnenden Sätze um drei. Möglich ist es aber auch, zwei oder vier Gewinnsätze zu spielen. Den Satz gewinnt, welcher Spieler zuerst elf Punkte gewonnen hat. Wenn aber beide Spieler zehn Punkte gesammelt haben, geht es in die Verlängerung. Wer in der Verlängerung zuerst mit zwei Punkten Vorsprung führt, gewinnt in diesem Fall den Satz. Nach jedem Satz werden die Seiten gewechselt. Einen Spezialfall des Seitenwechsels gibt es im Verlängerungssatz, beispielsweise im fünften Satz bei drei Gewinnsätzen: hier findet ein zusätzlicher Seitenwechsel während des Satzes statt, sobald einer der Spieler fünf Punkte gewonnen hat.[8]

Für den Ballwechsel gibt es folgende Regeln, wobei der Aufschlag ein Spezialfall ist und nachfolgend beschrieben wird: Im Normalfall muss der Spieler den Ball während des Ballwechsels so spielen, dass der Ball auf gegnerischen Tischhälfte aufkommt. Dabei ist zu beachten, dass der Ball auf der eigenen Tischhälfte nur einmal aufkommen darf – danach muss der Ball herüber gespielt werden. Erlaubt ist, dass der Ball vorm Auftreffen auf der gegnerischen Tischhälfte das Netz oder den Netzpfosten berührt und dann auf der gegnerischen Tischhälfte aufkommt. Andere Gegenstände wie Wände, die Decke oder auch die Kleidung des Spielers dürfen nicht berührt werden. Ansonsten handelt es sich um einen Fehler und der Gegner erhält den Punkt. Der Ball muss außerdem die Tischoberfläche berühren, wird die Tischseite getroffen, handelt es sich ebenfalls um einen Fehler. Eine Ausnahme ist die Tischkante; kommt der Ball auf dieser auf, handelt es sich um einen gültigen Schlag. Des Weiteren ist es nicht erlaubt, sich mit der freien Hand auf dem Tisch aufzustützen.[8]

[8]Wichtigste Spielregeln - Auf Spiel, Satz und Sieg: `https://www.tischtennis.de/mein-sport/spielen/wichtigste-spielregeln.html`, letzter Aufruf 26.06.2023

Eine Besonderheit des Ballwechsels ist der Aufschlag: Zu Beginn des Spiels wird zunächst ausgelost, wer mit dem Aufschlag beginnt und wer auf welcher Seite spielt. Wer die Auslosung gewinnt, entscheidet sich entweder für den Aufschlag oder eine Seite. Der Aufschlag selbst wird wie folgt durchgeführt: Der Ball liegt zunächst frei auf der geöffneten Hand und wird dann mindestens 16 cm hinter dem Tisch und oberhalb des Tisches hochgeworfen. Der Ball darf den Körper oder den Boden nicht berühren, bevor er geschlagen wird. Der Aufschlag kann im Einzel mit Vorhand oder Rückhand durchgeführt werden. Außerdem darf der Ball nicht vom Körper verdeckt sein. Damit der Aufschlag gültig ist, muss der Ball zunächst auf der eigenen und dann auf der gegnerischen Tischhälfte aufkommen. Ist der Aufschlag dementsprechend gültig und berührt aber das Netz, wird der Aufschlag wiederholt. Kommt der Ball nicht, wie gerade beschrieben, zuerst auf der eigenen und dann auf der gegnerischen Tischhälfte auf und berührt dazu noch das Netz, handelt es sich um einen Fehler und der Gegner erhält den Punkt. Außerdem ist zu beachten, dass der Gegenspieler bereit ist. Ist dies nicht der Fall, wird der Aufschlag wiederholt. Das Aufschlagrecht wechselt, nachdem ein Spieler zweimal aufgeschlagen hat. Eine Ausnahme ist die bereits erwähnte Satzverlängerung: hier wird das Aufschlagrecht nach jedem Aufschlag gewechselt.[8]

Demnach handelt es sich bei Tischtennis um eine Sportart, die schnelle Reaktionen und eine gute Hand-Auge-Koordination erfordert. Tischtennis ist geprägt von Taktik, denn durch Platzierung, Variation des Spins und Geschwindigkeit des Balls kann der Gegner zu Fehlern gezwungen werden. In der Realität ist das Tischtennisspiel darüber hinaus noch geprägt vom Material des Spielers. Denn durch besondere Beläge, die beispielsweise außen Noppen aufweisen, kann Spin anders als mit glatten Belägen ohne Noppen erzeugt werden. Auf den ersten Blick scheint Tischtennis physisch nicht so anstrengend zu sein. Aber die Sidesteps, die ein Spieler an der Platte macht, erfordern Agilität und auch ein gewisses Maß an Kraft. Mit diesen kleinen, seitlichen Schritten kann der Spieler sich korrekt zum Ball stellen. Daher kann Tischtennis als ein laufintensiver Sport angesehen werden, bei dem viele kleine Schritte nötig sind. Weiterhin erfordert Tischtennis aufgrund der Variabilität in Spin, Platzierung und Geschwindigkeit der Bälle ein hohes Maß an Konzentration.

3.2 Technische Grundlagen

3.2.1 Grundlagen von Unity und Unity ML-Agents

Grundlegende Informationen zu Unity

Unity ist eine Spiel-Engine, die verschiedene Modulen zur Konstruktion von Spielen bereitstellt, die sich mit der Physik oder Dynamik der Spielwelt, der Eingabe und Ausgabe in Form von 3D-Darstellung, 2D-Zeichnung und Tonausgabe befassen. Die Struktur der Engine kann durch die folgenden Module beschrieben werden und ist in Abb. 3.7 zu sehen: In der obersten Schicht befindet sich die virtuelle Welt oder das Szenario, mit dem der Spieler interagieren kann. Hier können verschiedene Arten von simulierter Physik oder Interaktionen angewendet werden. Unterhalb dieser Ebene der virtuellen Welt befindet sich der Spielcode. Sein Zweck ist es, die Spielmechanik zu steuern, einschließlich einfacher Physik, der Anzeige von Parametern, der Netzwerkfunktionalitäten, der Animationen als autonome Agenten oder das Verhalten des Avatars des Spielers. Dieser Spielcode wird normalerweise in einer spielspezifischen Skriptsprache geschrieben. Die *Rendering Engine* kümmert sich um die Identifizierung

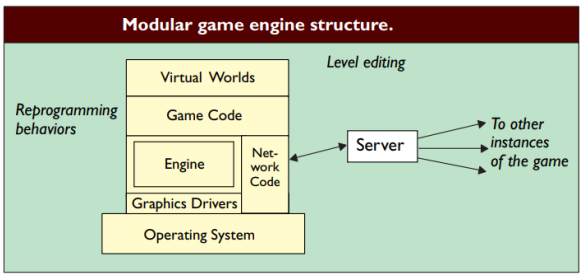


Abbildung 3.7: Darstellung aus [Lewis and Jacobson, 2002] von der modularen Architektur einer Spiel-Engine wie Unity.

und das Darstellen der Sicht des Spielers aus einem 3D-Modell der Umgebung. Die Netzwerkschicht liefert die Grundlagen für die Server-Client-Kommunikation. Der Server kommuniziert mit einem oder mehreren Client(s), um synchronisierte Informationen oder Daten über das Spiel oder den Spieler auszutauschen. Die nicht zu Unity gehörenden Grafiktreiber sorgen für die Übersetzung der von der *Rendering Engine* kommenden Anfragen an die Grafikbibliothek. [Lewis and Jacobson, 2002]

Unity ist eine Engine, aber auch die führende Plattform, wenn es um die Erstellung von Spielen und interaktiven 3D-Anwendungen in Echtzeit geht. [9] *Unity* hat bereits eine Rendering-Engine integriert. Der *Unity*-Editor ist einfach zu bedienen, da er auf „drag & drop“ basiert. Objekte können C#-Skripten zugewiesen werden, um das Verhalten des Objekts zu definieren oder um Informationen zu verwalten. *Unity* bietet auch die Möglichkeit, physikalische Eigenschaften von Objekten zu definieren, wie Masse, Sprungkraft, Widerstand, Federung und Kollisionserkennung. Darüber hinaus ist es vorteilhaft, dass es eine große Entwicklergemeinschaft gibt und eine umfangreiche Dokumentation für die gesamte API von *Unity* vorliegt. [Craighead et al., 2008] Der *Unity*-Editor ist für die Anwendungsentwicklung unerlässlich, weil er Menüelemente hinzufügt, Assets verwendet, Szenen erstellt oder Builds veröffentlicht. [10] Eine Szene enthält die Umgebungen und Menüs des Spiels und kann mit `GameObjects` und Komponenten erstellt werden. In einer Szene können Eingaben gelesen oder Interaktivität hinzugefügt werden, um das Spielgeschehen zu definieren. Verschiedene Assets und Pakete, die von anderen oder von *Unity* erstellt wurden, sind im Asset Store oder im Package Manager zu finden. [11] Assets sind Komponenten, die importiert und für die Erstellung einer Anwendung verwendet werden können. Das können visuelle oder Audio-Elemente oder auch Animationen sein. Neben extern entwickelten Assets können auch eigene Assets in *Unity* erstellt werden, wie z.B. einen Animations-Controller, Audio-Mixer oder „Render Texture“. [12] Darüber hinaus bietet *Unity* Werkzeuge für die Erstellung von Echtzeitprojekten. Außerdem enthält *Unity* integrierte Dienste (*Unity Gaming Services*) zur Skalierung des Entwicklungszyklus. Dieser Service umfasst Build- und Entwicklungstools sowie Analysetools. [10] Auch die VR-Entwicklung ist mit

[9] Unity-Dokumentation Get Started with Unity: `https://unity.com/de/learn/get-started`, letzter Aufruf 19.01.2023

[10] Unity-Dokumentation: `https://docs.unity.com/`, letzter Aufruf 19.01.2023

[11] Unity-Dokumentation Working in Unity: `https://docs.unity3d.com/Manual/UnityOverview.html`, letzter Aufruf 19.01.2023

[12] Unity-Dokumentation Asset Workflow: `https://docs.unity3d.com/Manual/AssetWorkflow.html`, letzter Aufruf 19.01.2023

Unity möglich. Unterstützte Plattformen von Unity sind z.B. Oculus oder OpenXR. Mit dem XR Interaction Toolkit kann der Entwickler Interaktivität in die VR-Anwendung einbauen. [13]

Grundlegende Informationen zum Aufbau eines Unity Projektes

Unity bietet diverse Templates für verschiedene Projektarten. Dabei kann beispielsweise zwischen zweidimensionalen oder dreidimensionalen Umgebungen unterschieden werden. Diese Templates beinhalten oftmals eine Reihe von Features und Anwendungsbeispielen, um dem Nutzer den Einstieg in das Arbeiten mit der Unity-Umgebung zu vereinfachen. Besagte Templates enthalten nicht nur die zuvor erwähnten Assets und Skripte für ein jeweiliges Environment, sondern auch projektinterne Informationen über die Spielwelt selbst [14]. Es werden also auch Informationen über die physikalischen Gesetzlichkeiten der Spielwelt innerhalb der Projektinformationen verwaltet. Darunter befinden sich zum Beispiel die „Gravitation“, oder technisch gesehen die Beschleunigung entlang der globalen y-Achse der Spielwelt [15]. Andere dort enthaltene Informationen sind beispielsweise die Interaktionsregeln verschiedener *Layer*[16]. Insgesamt können Objekte in Unity mit sogenannten *Tags*[17] versehen werden, welche dann später zur Identifikation von Gruppen von Objekten verwendet werden können. Weiterhin können auch über Tags das Kollisionsverhalten verschiedener Objektgruppen mit Hinblick auf physikalischen Kontakt sowie Interaktionen mit Licht verwaltet werden. So ist es also möglich dort Gruppen von Objekten für andere Gruppen in allen Belangen unsichtbar werden zu lassen.

Des Weiteren beinhaltet ein Unity-Projekt Informationen über verwendete Drittanbieter-Pakete, welche sich in Erweiterungen des Unity-Editors, Assets oder sonstigen Funktionalitäten äußern können [18]. Darüber hinaus beinhaltet ein Unity-Projekt auch Informationen über die lokalen Einstellungen und externe Tools, die in Verbindung mit Unity verwendet werden. Darunter befinden sich beispielsweise Einstellungsmöglichkeiten für den verwendeten Code Editor (für die PGMMI2: JetBrains Rider) sowie unter anderem auch Optionen zur verwendeten Versionskontrolle [19]. Es lassen sich also innerhalb der Unity Projekteinstellungen kritische Ressourcen verwalten, welche nicht nur einen Einfluss auf die Spielwelt, sondern auch auf den Workflow der Nutzer ausüben.

Zunächst wird durch Unity keine strikte Ordnerstruktur über verwendete Assets, Skripte oder Ähnliches vorgegeben. Jedoch bietet es sich aufgrund der Parität mit weiteren Drittanbieter-Assets an, eigene Projektdateien nicht in im `root`-Ordner des Projektes zu verorten, sondern diese stattdessen in einem weiteren Ordner unterzu-

[13] Unity Dokumentation VR development in Unity: `https://docs.unity3d.com/Manual/VROverview.html`, letzter Aufruf 19.01.2023

[14] Unity - Manuals: Project Templates `https://docs.unity3d.com/2021.1/Documentation/Manual/ProjectTemplates.html`, letzter Aufruf 29.09.2023

[15] Unity-Dokumentation Physics.gravity: `https://docs.unity3d.com/ScriptReference/Physics-gravity.html`, letzter Aufruf 29.08.2023

[16] Unity-Dokumentation Layers: `https://docs.unity3d.com/Manual/Layers.html`, letzter Aufruf 29.08.2023

[17] Unity-Dokumentation Tags: `https://docs.unity3d.com/Manual/Tags.html`, letzter Aufruf 29.08.2023

[18] Unity - Manual: Unity's Package Manager `https://docs.unity3d.com/2021.1/Documentation/Manual/Packages.html`, letzter Aufruf 29.08.2023

[19] Unity - Manual: Preferences `https://docs.unity3d.com/Manual/Preferences.html`, letzter Aufruf 29.08.2023

ordnen. So wird eine klare Trennung der Assets von verschiedenen Anbieter sichergestellt.

Grundlegende Funktionsweisen des ML-Agents Toolkits

Das Open-Source Toolkit *Unity ML-Agents* ermöglicht es künstliche Intelligenz für Nicht-Spieler-Charaktere (Non-Player Characters, NPCs) in ein existierendes Unity Projekt zu integrieren. Darüber hinaus kann das besagte Unity Projekt auch als Trainingsenvironment für die KI-Komponente genutzt werden. Die Methoden, die das Toolkit anbietet, stammen hierbei primär aus dem Bereich des Reinforcement Learnings, es stehen jedoch auch Imitation-Learning und seit Veröffentlichung der Version 2.0 auch des Multi-Agent-Reinforcement-Learning zur Verfügung. Da die letzteren für dieses Projekt jedoch weniger relevant sind, wird im Folgenden auf diese nicht näher eingegangen.[20] [Juliani et al., 2018]

Ein Agent und dessen *Behavior*, also sein Verhalten innerhalb seiner Umgebung, sind immer an ein bestehendes `GameObject` gekoppelt. In der Abbildung 3.8 ist ein beispielhafter Aufbau eines Trainingsenvironments mit *ML-Agents* schematisch dargestellt. *Learning Environment* beschreibt hierbei das Unity Projekt und besteht aus mindestens einer `Scene`. Ein *Agent* befindet sich hierbei innerhalb dieser Umgebung und nimmt über vordefinierte Sensoren sein Environment in Form von Observationen wahr. Diese Observationen bestimmen, wie der Agent basierend von der Verhaltensfunktion, hier das *Behavior*, handelt. Die Grundlage dieses *Behavior* kann unterschiedlich sein: auf Basis eines vorgegebenen Regelsatzes (*Heuristic*) oder eines vorhandenen neuronalen Netzes (*Inference*). Die dritte Variante ist, dass das Verhalten im *Learning Environment* trainiert werden soll. Für dieses Training müssen zwei externe Python-Komponenten vorliegen: die *Python API* und der *Python Trainer*. Die *Python API* ist ein ausgelagertes Interface, welches von außerhalb des Unity Environments dieses manipulieren und anderweitig mit diesem interagieren kann. Darüber hinaus interagiert nur dieses Interface mit dem *Python Trainer*, welcher alle nötigen Machine-Learning-Algorithmen für das Training des Agenten enthält.[20]

Für das Training mit *ML-Agents* stehen verschiedene Algorithmen zur Verfügung. Als Standardlösung empfehlen die Autoren die Nutzung der *Proximal Policy Optimization* (PPO)[21] und für langsamere Environments als off-policy Lösung den *Soft Actor Critic* (SAC) Algorithmus [Haarnoja et al., 2018]. Als „langsameres“ Environment klassifizieren die Autoren Environments mit mehr als 0,1 Sekunden pro Trainingsschritt.[20]

Potenziell relevant für dieses Projekt schien das *Intrinsic Curiosity Module* für Agenten. Dieses liefert dem Agenten intrinsische Reward-Signale abhängig von dessen Fähigkeit, die Auswirkungen seiner Aktionen vorherzusagen. Je höher dabei die Abweichung zwischen Vorhersage und beobachteter Auswirkung der Aktion, desto höher der Reward. Das Ziel ist es dabei, die Exploration des Handlungsspielraumes durch den Agenten zu fördern, sollte das Environment nur geringfügige oder nur selten Rewards zurückgeben.[20]

Im Rahmen kompetitiver Environments bietet das *ML-Agents* Toolkit die Möglichkeit, Trainings mit PPO oder SAC um *Self-Play*[Bansal et al., 2018] zu erweitern. Hierfür

[20] ML-Agents Toolkit Overview: `https://unity-technologies.github.io/ml-agents/ML-Agents-Overview/`, letzter Aufruf 29.09.2023

[21] OpenAI PPO-Dokumentation: `https://spinningup.openai.com/en/latest/algorithms/ppo.html`, letzter Aufruf: 29.09.2023

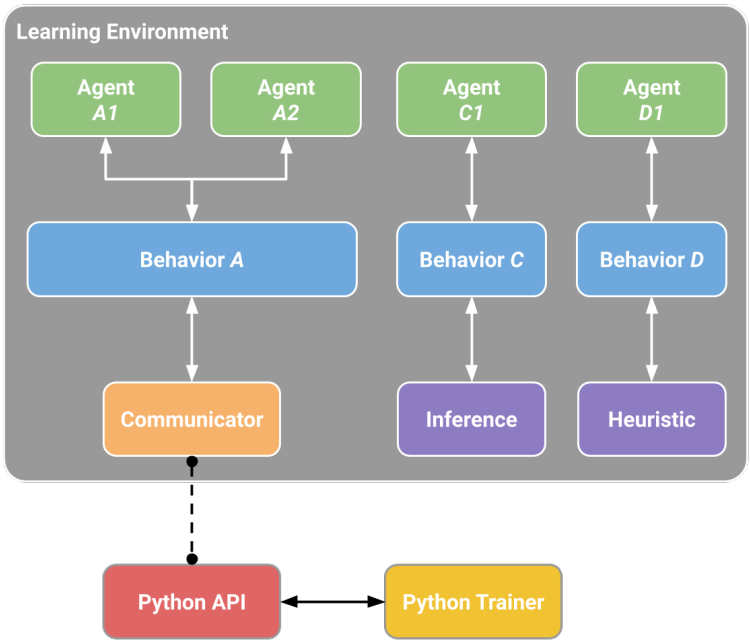


Abbildung 3.8: Exemplarische Lernumgebung mit Unity ML-Agents[20]

müssen zunächst die Agenten in unterschiedliche Teams aufgeteilt werden, wobei dies innerhalb des Unity-Editors erfolgen kann. Zu jedem Zeitpunkt während des Trainings fungiert eins der Teams als lernende Instanz. Das Team, welches nicht lernt, verwendet eine veraltete Version der Policy, wobei diese in festgelegten Intervallen ausgetauscht bzw. aktualisiert wird. Darüber hinaus wird in regelmäßigen Abständen gewechselt, welches Team als lernende Instanz dient und welches die veraltete Policy verwendet. Über die *Self-Play*-Parameter in den Trainingskonfigurationen lässt sich ebenfalls einstellen, wie viele veraltete Versionen der Policy, aus denen für das nichtlernende Team gewählt wird, zwischengespeichert werden und wie wahrscheinlich es ist, dass das lernende Team gegen die aktuellste Version der Policy spielt.[20]

3.2.2 Grundlagen von Virtual Reality

Virtual Reality (VR) Anwendungen ermöglichen dem Nutzer in eine stereoskopische Welt einzutauchen, welche selbigem ein einzigartiges Erlebnis einer gegebenen Umgebung ermöglicht. Dies wird hierbei durch ein sogenanntes VR-Headset ermöglicht, welches ein Gerät nicht unähnlich einer Brille darstellt, das dem Nutzer zwei leicht unterschiedliche Bilder gleichzeitig vor dessen Augen präsentiert, sodass hierdurch ein dreidimensionaler Eindruck erschaffen wird [Rubio-Tamayo et al., 2017]. Zusätzlich zur Präsentation einer virtuellen Welt ermöglichen viele VR-Anwendungen die Interaktion mit der gleichen durch entsprechende Controller, welche je nach Gerät dem Nutzer einen unterschiedlichen Interaktionsumfang bieten kann. Das in PG-MMI2 verwendete VR-Headset, die *HTC VIVE Pro Eye*, liefert mittels der dazugehörigen Controller, dem *Controller 2.0*, dem Nutzer ein vergleichsweise kompromissloses VR-Erlebnis. Da sich die Interaktionsmöglichkeiten auf wenige Knöpfe und ein multifunktionales Trackpad beschränken, werden dementsprechend nur ganzheitli-

che Bewegungen wie bspw. vollständiges Greifen eines Objektes erfasst und in der VR-Umgebung wiedergegeben. Teilbewegungen einzelner Finger können nicht erkannt werden. Insgesamt erfolgt die Verortung der Controller sowie des Headsets mittels des sogenannten *Lighthouse*-Systems von Valve, welches auf zwei, drei oder vier *Basisstationen* aufbaut. Durch diese ist es möglich mittels vorherigen Ausmessens des Anwendungsbereiches und der Kommunikation der Geräte durch einen verbundenen Rechner eine exakte Positionierung der VR-Objekte im echten Raum zu ermitteln.[22]

Im Kontext eines SG kann die Verbindung von spielerischen Ansätzen und der Verwendung von VR-Umgebungen das Lernen, die Benutzererfahrung und die Wissensvermittlung messbar verbessern. Da VR-Anwendungen einen hohen Grad an Immersion und Realismus vermitteln, verwischt sich so die Grenze zwischen realer und digitaler Welt, was wiederum die Übertragbarkeit dem erlangten Wissen verbessern kann. Hierbei steht also die Interaktivität und Benutzerfreundlichkeit eines gegebenen Systems im direkten Zusammenhang mit der vermittelten Lernrate.
[Checa and Bustillo, 2020] Daher besteht in VR-Anwendungen eines SG die Herausforderung, reale Situationen so genau wie möglich zu simulieren. Da jedoch dieser Bereich durch bisher nur wenige Studien abgedeckt wurde, gilt es also herauszufinden, welche Faktoren für Sportsimulationen wie Tischtennisspiele in VR-Anwendungen besonders entscheidend sind. [Miles et al., 2012]

Innerhalb der VR-Hardware-Landschaft können HMDs in eigenständige, kabelgebundene und mobile VR-Geräte unterteilt werden, die alle ihre eigenen Merkmale aufweisen. Eigenständige HMDs zeichnen sich durch ihre Autarkie aus, da sie alle erforderlichen Datenverarbeitungskomponenten bereits besitzen. Diese Geräte zeichnen sich durch Mobilität und Benutzerkomfort aus, sind jedoch aufgrund interner Hardwarebeschränkungen in ihrer Leistung eingeschränkt und erfordern ein regelmäßiges Aufladen der Batterien. Im Gegensatz dazu sind kabelgebundene HMDs für die Datenverarbeitung auf externe Computer angewiesen, die komplexe VR-Erlebnisse ermöglichen, mit dem Nachteil, dass eine ständige Kabelverbindung erforderlich ist. Mobile VR-Geräte, eine Abwandlung der eigenständigen HMDs, nutzen Smartphones zur Berechnung und Erzeugung dreidimensionaler Umgebungen und sind dadurch erschwinglich und mobil. Allerdings haben sie die gleichen Einschränkungen bei der Datenverarbeitung wie eigenständige Geräte. [Angelov et al., 2020]

Die Qualität der Displays in Virtual Reality (VR)-Headsets spielt eine entscheidende Rolle bei der Gestaltung der Erfahrung des Benutzers in der dreidimensionalen VR-Umgebung. [Dinh et al., 1999] [Bowman and McMahan, 2007]
[Parsons and Cobb, 2016] Die Displaytechnologie ist ein entscheidender Faktor bei der Bestimmung der Bildqualität, wobei die beiden vorherrschenden Typen LCD- und AMOLED-Displays sind. Während mit der LCD-Technologie eine hohe Bildqualität erreicht werden kann, sind AMOLED-Displays für ihre überlegene Farbwiedergabe und Schwarzdarstellung bekannt, was besonders in Weltraumsimulationen oder Horrorspielen zu realistischeren Bildern und einem verbesserten VR-Erlebnis führt. Abgesehen von der Display-Technologie wird die Bildqualität von VR-Headsets durch das Screen-Door-Artefakt beeinflusst, das die Pixelkonturen umfasst und zu einem spürbaren Gittereffekt führt. Dieses Problem wird in VR durch die Nähe der Augen des Benutzers zu den Displays noch verstärkt. Die Stärke des Screen-Door-Effekts hängt in erster Linie von der Pixeldichte und der Bildschirmauflösung ab, wobei dich-

[22] HTC VIVE user guide: `https://developer.vive.com/documents/720/Vive_User_Guide.pdf`, letzter Aufruf 01.09.2023

tere Pixelanordnungen die Sichtbarkeit einzelner Pixel verringern und die allgemeine Bildqualität in VR verbessern. [Dinh et al., 1999] [Bowman and McMahan, 2007] [Parsons and Cobb, 2016]

Der Eckpfeiler moderner VR-Systeme sind ihre Tracking-Systeme, die für die Bestimmung der Position und Ausrichtung des Blickpunkts des Benutzers verantwortlich sind. [Cummings and Bailenson, 2016] Die Genauigkeit dieses Trackings hat einen großen Einfluss auf die Qualität der Immersion in VR. Studien, wie die erwähnte Meta-Analyse in [Bekele and Champion, 2019], zeigen, dass das Präsenzgefühl des Benutzers stark von der Geschwindigkeit und Präzision der Positionsbestimmung beeinflusst wird, zusätzlich zu Faktoren wie dem Sichtfeld der Anzeige. Das Tracking lässt sich grob in zwei Kategorien einteilen: Orientierungs- und Positionsverfolgung, die die Freiheitsgrade (DoF) definieren, innerhalb derer die Positionsänderungen der Zielgeräte überwacht werden. Beim Orientierungs-Tracking wird die Ausrichtung der Geräte im dreidimensionalen Raum mithilfe von Sensoren wie Beschleunigungsmessern, Gyroskopen und Kameras ermittelt. Im Gegensatz dazu wird bei der Positionsverfolgung nicht nur die Orientierung, sondern auch die räumliche Position bestimmt. Dieser Ansatz erhöht das Potenzial für realistische Benutzerinteraktionen mit der VR-Umgebung und ist somit ideal, um eine Erfahrung voller Immersion zu erreichen. [Cummings and Bailenson, 2016] [Bekele and Champion, 2019]

Controller sind ein wichtiger Bestandteil von VR-Headsets, da sie die direkte Interaktion des Benutzers mit der virtuellen Umgebung ermöglichen und durch die über sie mögliche Repräsentation der Hände zu einem umfassenden Erlebnis beitragen. Sie ermöglichen nicht nur Eingaben, sondern geben auch taktiles Feedback als Reaktion auf Benutzeraktionen. Die Art des verwendeten Tracking-Systems spielt eine entscheidende Rolle für die Fähigkeiten des Controllers, wobei das Positions-Tracking, das für immersive VR bevorzugt wird, zu zwei verschiedenen Kategorien führt: 3DoF- und 6DoF-Controller. Bemerkenswert ist, dass alle hier betrachteten Geräte mit 6DoF-Controllern kompatibel sind. Eine besondere Erwähnung verdient der Valve Index mit seinen „Knuckles"-Controllern, die sich durch ein fortschrittliches Finger-Tracking-System auszeichnen. Diese Controller verfügen über zahlreiche Sensoren, die die Positionen der einzelnen Finger präzise erfassen und verarbeiten, wodurch die Interaktionsgenauigkeit und -komplexität in der virtuellen Umgebung erhöht wird, was sich positiv auf das Präsenzgefühl des Benutzers auswirken kann. [Angelov et al., 2020]

3.2.3 Grundlagen von *REST* APIs

Eine Möglichkeit, Schnittstellen (APIs) für webbasierte Anwendungen zu erstellen, sind sogenannte *REST*ful Webservices.

REST steht hierbei für *Representional State Transfer* und bezeichnet Architekturprinzipien, mit denen Webdienste erstellt werden können. Diese Webdienste fokussieren sich auf Systemressourcen und wie Ressourcenzustände angesprochen und über HTTP an einen Client, egal in welcher Programmiersprache dieser implementiert worden ist, transferiert werden können. *REST* sei einfacher zu verwenden als andere Schnittstellendesigns, die auf SOAP oder WSDL basieren. Folgende Gestaltungsprinzipien sind bei der Implementierung eines *REST*-Webdienstes zu beachten:
[Rodriguez, 2008]

Explizite Verwendung von HTTP-Methoden

Die HTTP-Methoden sollen explizit verwendet werden:
Die POST-Methode ermöglicht, eine Ressource zu erstellen, während die GET-Methode dafür verwendet wird, eine Ressource abzurufen. Zur Erstellen einer neuen Ressource oder zum Ersetzen einer Repräsentation der Zielressource wird die PUT-Methode[23] verwendet. Der Unterschied zur POST-Methode liegt darin, dass der mehrfache Aufruf der gleichen PUT-Methode die gleiche Wirkung, aber keine Nebenwirkungen hat. Zur Löschung einer Ressource wird die DELETE-Methode verwendet. So existiert eine Eins-zu-Eins-Zuordnung von den HTTP-Methoden zu den CRUD-Operationen *Erstellen*, *Lesen*, *Aktualisieren* und *Löschen*. [Rodriguez, 2008]

Zustandsloses Design

Eine *REST* Webdienstanwendung stellt in den HTTP-Headern und Body einer Anfrage alle notwendigen Parameter, Kontext und Daten bereit, die der Server benötigt zur Erzeugung einer Antwort. Dies wird als „Zustandslosigkeit" bezeichnet und sorgt für eine bessere Leistung des Webdienstes und vereinfacht die Implementierung und das Design des Servers bzw. der serverseitigen Komponente. Ohne diesen Zustand muss der Server nicht noch die Sitzungsdaten mit einer externen Anwendung synchronisieren. [Rodriguez, 2008]

Diese Zustandslosigkeit eines *REST* Webdienstes äußert sich in den Eigenschaften bzw. Zuständigkeiten von Server und Client:
Der **Server** erzeugt Antworten, die Links zu anderen Ressourcen enthalten können. So können Anwendungen zwischen den Ressourcen wechseln. Außerdem kann der Server Antworten erstellen, die eine Information enthalten, ob sie cachefähig sind oder nicht. So kann die Leistung der Anwendung durch gezieltes Caching verbessert und unnötige Anfragen vermieden werden. Der **Client** verwendet einen besonderen Header, den „Cache-Control Header„, zur Bestimmung, ob eine Ressource eine lokale Kopie gespeichert werden soll oder nicht. Weiterhin kann der Client über den „Last-Modified-Header„ der Antwort mittels einer Conditional Get-Anfrage den Server fragen, ob sich eine Ressource geändert hat und die Ressource nur dann zu senden, wenn diese sich geändert hat. Außerdem kann der Client vollständige Anfragen senden, die unabhängig von anderen Anfragen bearbeitet werden können. [Rodriguez, 2008]

Verzeichnisstruktur ähnlich zu URLs

Die verwendeten URLs sollten intuitiv und leicht verständlich sein, sodass sie als Schnittstelle fungieren, die sich selbst dokumentiert. Es soll nur durch den Namen der URL erkennbar sein, worauf verwiesen wird und welche Ressourcen abzufragen sind. Um eine hohe Verständlichkeit zu gewährleisten, kann die Definition einer URL einer Verzeichnisstruktur ähneln: In dieser hierarchischen Struktur ist ein Pfad als Wurzel definiert, mit weiteren, abzweigenden Unterpfaden. Weitere Leitlinien für die Definition der URL-Struktur lauten wie folgt: Es sollen stets Kleinbuchstaben und der Plural verwendet und Leerzeichen durch Unter- oder Bindestriche ersetzt werden. Auf Abfrage-Strings soll so weit wie möglich verzichtet werden und stattdessen die Request-Parameter verwendet werden. Außerdem soll statt einer 404-„not found" Ressource stets eine Standardressource als Antwort definiert sein. Zuletzt sollen URLs

[23] PUT method: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT`, letzter Aufruf 28.09.2023

statisch sein, sodass der Link auch bei Ressourcenänderungen stets gleich und verfügbar bleibt. Hinsichtlich der Definition der URLs sollen sie nach Ressourcen organisiert sein. [Rodriguez, 2008]

Transfer von Daten der Typen XML oder JSON

Tabelle 3.1 zeigt die verschiedenen Datentypen, die über einen *REST*ful-Dienst angefragt werden können. Dabei handelt es sich um JSON und HTML. Dieses Prinzip, die MIME-Typen und HTTP Accept zu verwenden, wird *Content Negotiation* genannt und ermöglicht dem Client, das passende Datenformat zu wählen und die Datenkopplung zwischen dem *REST*ful Dienst und der Anwendungen zu minimieren. [Rodriguez, 2008]

JSON	application/json
HTML	application/xml

Tabelle 3.1: MIME Typen nach [Rodriguez, 2008], mit denen bestimmte Content-Typen, wie auch CSV, ini, yaml oder toml, über *REST*ful-Dienste angefragt werden können.

3.2.4 Wahl des Frameworks und Sprachmodells für den Chatbot

Mögliche, verfügbare Tools zur Implementierung eines Chatbots samt Vor- und Nachteilen[24] sind in Tabelle 3.2 dargestellt.

Wenn die Tools aus Tabelle 3.2 verglichen werden, sind begrenzte Gestaltungsmöglichkeiten und hohe oder unabsehbare Kosten für die PG schwerwiegende Nachteile. Denn zum einen verfügt die PG über begrenzte Mittel, zum anderen soll ein Chatbot erstellt werden, der konkret für unsere Anwendung des VR-Tischtennisspiels nutzbar ist. Daher fiel in Betrachtung der vorgestellten Tools die Wahl auf **Rasa**[25].

Als Alternative zur Implementierung des Chatbots ist im Rahmen von Recherche und Diskussionen neben **Rasa**[25] auch **ChatGPT** in der engeren Auswahl.

ChatGPT-3 ist ein Transformer-Sprachmodell von OpenAI, das aufgrund der großen Menge an Trainingsdaten universell nutzbar ist.[26] In dem Artikel[26] vergleicht der Autor Rasa[25] und ChatGPT-3 miteinander und kommt zu dem Schluss, dass Rasa[25] zu bevorzugen ist. Denn der große Vorteil von Rasa[25] ist die Flexibilität, dass die eigenen Trainingsdaten für den speziellen Anwendungsfall des Chatbots definiert werden können.[26] Diese Argumente sind aber überholt, denn seit kurzem besteht die Möglichkeit, ein ChatGPT-Sprachmodell mit eigenen Trainingsdaten zu trainieren. Jedoch sei das Feintuning mit eigenen Daten für das aktuelle ChatGPT-3.5-turbo Sprachmo-

[24] ChatGPT vs. the World: A Comprehensive Guide to Conversational AI Rivals: `https://medium.com/@siva.desetti27/chatgpt-vs-the-world-a-comprehensive-guide-to-conversational-ai-rivals-9d279874e50`, letzter Aufruf 06.07.2023

[25] Rasa-Dokumentation: `https://rasa.com/docs/rasa/`, letzter Aufruf 20.09.2023

[26] GPT-3 vs. Rasa chatbots - Comparing the performance of GPT-3 and a custom-trained Rasa chatbot: `https://towardsdatascience.com/gpt-3-vs-rasa-chatbots-8b3041daf91d`, letzter Aufruf 14.07.2023

Tool	Vorteile	Nachteile
Google Dialogflow	nutzerfreundliches Interface zur Chatboterstellung, sehr gute NLP	schwierige Abschätzung der Kosten
IBM Watson Assistant	sehr gute NLP, nutzerfreundliches Interface zur Chatboterstellung ohne Programmierkenntnisse	hohe Kosten, begrenzte Gestaltungs-möglichkeiten
Rasa[25]	open source, geringe Kosten, sehr gute NLP	komplexes Nutzerinterface, begrenzte Dokumentation und Support von Rasa[25]
Amazon Lex	sehr gute NLP, Integration mit anderen Amazon Services	hohe Kosten, begrenzte Gestaltungsmöglichkeiten

Tabelle 3.2: Tools zur Implementierung eines Chatbots samt Vor- und Nachteilen[24]

dell momentan nicht möglich.[27] Weiterhin stellt ChatGPT TTS-Funktionalität über ihr Open-Source Whisper-Modell bereit.[28]

Letztendlich fiel die Wahl des Sprachmodells auf Rasa[25], da es zum Zeitpunkt des Implementierungsbeginns hinsichtlich des Trainierens mit eigenen Daten etwas etablierter ist.

3.2.5 Rasa[25] Grundlagen

Nach [Jiao, 2020] enthält die Rasa[25] Chatbot-Pipeline **Rasa NLU**. Dabei handelt es sich um ein Modul zur Entitätssuche. Konkret ist die **Rasa NLU** dafür zuständig, Entitäten aus der Nachricht des Nutzers zu abstrahieren und Intents (Absichten) zu klassifizieren. Dafür basiert **Rasa NLU** auf der Pipeline `spacy_sklearn`, die einige NLP-Funktionalitäten enthält. Je nach abstrahierter Entität und damit verknüpfter Antworten bzw. Aktionen kann der Chatbot dann entsprechend antworten. Damit der Chatbot die verschiedenen Entitäten und Intents zu abstrahieren erlernen kann, werden diese, wie nachfolgend beschrieben, in einer Datei definiert. [Jiao, 2020]

[Sharma and Joshi, 2020] nennen neben **Rasa NLU** noch **Rasa Core** als wichtige Komponente, die dafür zuständig ist, basierend auf der Eingabe des Nutzers Entscheidungen zu treffen. Abbildung 3.9 aus [Sharma and Joshi, 2020] zeigt die Architektur des Rasa[25]-Frameworks und den Informationsfluss. Zuerst wird die Nutzereingabe im Interpreter (**Rasa NLU**) analysiert, die Ausgabe enthält den ursprünglichen Text, Absicht und Entitäten. Der Tracker empfängt diese Informationen und enthält den

[27] OpenAI-API-Dokumentation Fine-tuning: `https://platform.openai.com/docs/guides/fine-tuning`, letzter Aufruf 14.07.2023

[28] OpenAI-API-Dokumentation Speech to text: `https://platform.openai.com/docs/guides/speech-to-text`, letzter Aufruf 14.07.2023

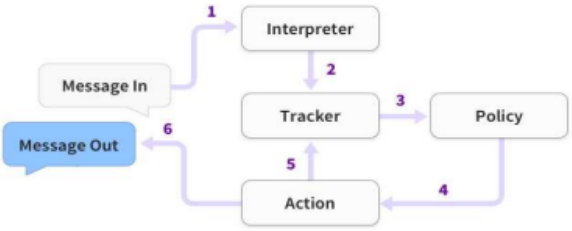


Abbildung 3.9: Darstellung aus [Sharma and Joshi, 2020] von der Architektur von Rasa[25] samt Informationsfluss.

momentanen Zustand der Konversation. Die Ausgabe des Trackers enthält seinen aktuellen Zustand und wird der Policy übergeben. Auf dieser Grundlage wird die nächste Aktion bestimmt, worüber der Tracker informiert wird. Zuletzt wird die bestimmte Aktion ausgeführt. Die Policy ist eine Art Entscheidungsinstanz und dafür zuständig, die nächste Aktion des Chatbots zu bestimmen. Rasa Core stellt verschiedene Policies bereit. [Sharma and Joshi, 2020]

Damit ein Rasa[25]-Sprachmodell trainiert werden kann, müssen die Trainingsdaten definiert werden. Zunächst lassen sich die Trainingsdaten in NLU- und Konversations-Trainingsdaten aufteilen. Alle Trainingsdaten werden im Format YAML definiert.[29]

Die NLU-Trainingsdaten sind Beispiele der Äußerungen, die der Nutzer machen kann. Die Äußerungen sind in einer YAML-Datei unter dem Key `nlu` aufgelistet und verschiedenen Absichten (*Intents*) zugeordnet. Außerdem können Entitäten dazu verwendet werden, Informationen aus der Äußerung des Nutzers zu extrahieren. Des Weiteren können auch Lookup-Tabellen oder reguläre Ausdrücke in den NLU-Trainingsdaten definiert werden.[29]

Mit den Konversations-Trainingsdaten wird das Dialogmanagement-Modell trainiert. Die Storys werden genutzt, um Muster in Konversationen zwischen dem Nutzer und des Chatbots zu erkennen und auf unbekannte Gesprächsverläufe verallgemeinert. Storys bestehen aus dem Namen der Story und einer Liste an Steps, die die Nutzeräußerungen (Intents aus den NLU-Trainingsdaten) und die Aktionen enthalten und den Verlauf der Konversation definieren. Aktionen können simple Antworten, die dem Nutzer ausgegeben werden, oder Custom Actions sein. Diese Actions können dafür genutzt werden, um beispielsweise beliebigen Code auszuführen. Im Gegensatz dazu werden noch Rules verwendet. Mit diesen können kurze Bestandteile von Konversationen definiert werden, die immer den gleichen Ablauf haben, um die `RulePolicy` zu trainieren.[29]

Die in den Trainingsdaten Intents, Chatbot-Antworten und, wenn definiert, auch die Entitäten, Slots, Forms und Actions werden ebenfalls in der Domain YAML-Datei definiert. Die Domäne definiert die Umgebung des Chatbots und ermöglicht, die Konfiguration für die Konversations-Sessions zu definieren. So kann beispielsweise eine Zeit angegeben werden, nach der die Session abläuft.[30]

[29] Rasa-Dokumentation Training Data Format: `https://rasa.com/docs/rasa/training-data-format/`, letzter Aufruf 16.07.2023

[30] Rasa-Dokumentation Domain: `https://rasa.com/docs/rasa/domain/`, letzter Aufruf 16.07.2023

3.2.6 Verwendete Bibliotheken und Pakete

Im Folgenden werden die im Backend verwendeten Bibliotheken aufgeführt:

- Flask[31]: Framework zur Erstellung von Webapps
- gTTS[32]: Bibliothek und CLI Werkzeug zur Anbindung an die Text-to-Speech API von Google Translate
- matplotlib[33]: Bibliothek zur Erstellung von statischen, animierten und interaktiven Visualisierungen
- NumPy[34]: Bibliothek zur Bereitstellung von mehrdimensionalen Array-Objekten, Matrizen und jegliche Operationen auf Arrays
- pandas[35]: Bibliothek zur Datenanalyse und -manipulation
- pandas-stubs[36]: Typ-Annotationen für Pandas
- Pillow[37]: Python Bildbearbeitungsbibliothek
- pip[38]: Package-Installer für Python
- PyAudio[39]: Bibliothek zum Abspielen und Aufnehmen von Audio mithilfe von Python
- pygame[40]: Bibliothek zur Erstellung von Videospielen und Multimedia-Anwendungen in Python. Es umfasst auch Funktionen zum Laden und Abspielen von Audios
- pytest[41]: Bibliothek zum Schreiben von Tests, auch für komplexe Tests von Applikationen
- pytest-cov[42]: Pytest-Plugin zum Erfassen der Testabdeckung
- PyTorch[43]: Tensor-Bibliothek für Deep Learning mithilfe von GPUs und CPUs
- PyYAML[44]: YAML-Parser und -Emitter für Python
- Rasa[25]: Erstellung von Chat- und sprachbasierten AI-Assistenten

[31] Flask-Dokumentation: `https://flask.palletsprojects.com/en/2.3.x/`, letzter Aufruf 20.09.2023

[32] gTTS-Dokumentation: `https://gtts.readthedocs.io/en/latest/`, letzter Aufruf 20.09.2023

[33] matplotlib-Dokumentation: `https://matplotlib.org/`, letzter Aufruf 20.09.2023

[34] NumPy-Dokumentation: `https://numpy.org/`, letzter Aufruf 20.09.2023

[35] pandas-Dokumentation: `https://pandas.pydata.org/`, letzter Aufruf 20.09.2023

[36] pandas-stubs-Dokumentation: `https://pypi.org/project/pandas-stubs/1.2.0.58/`, letzter Aufruf 20.09.2023

[37] Pillow-Dokumentation: `https://pillow.readthedocs.io/en/stable/`, letzter Aufruf 20.09.2023

[38] pip-Dokumentation: `https://pip.pypa.io/en/stable/`, letzter Aufruf 20.09.2023

[39] PyAudio-Dokumentation: `https://people.csail.mit.edu/hubert/pyaudio/docs/`, letzter Aufruf 20.09.2023

[40] pygame-Dokumentation: `https://www.pygame.org/wiki/about`, letzter Aufruf 20.09.2023

[41] pytest-Dokumentation: `https://docs.pytest.org/en/7.4.x/`, letzter Aufruf 20.09.2023

[42] pytest-conv-Dokumentation: `https://pytest-cov.readthedocs.io/en/latest/`, letzter Aufruf 20.09.2023

[43] PyTorch-Dokumentation: `https://pytorch.org/docs/stable/index.html`, letzter Aufruf 20.09.2023

[44] PyYAML-Dokumentation: `https://pyyaml.org/wiki/PyYAMLDocumentation`, letzter Aufruf 20.09.2023

- requests[45]: HTTP-Client-Bibliothek für Python
- scikit-learn[46]: Klassisches Machine Learning in Python
- setuptools[47]: Bibliothek zum Packaging von Python-Projekten
- SpeechRecognition[48]: Bibliothek für Spracherkennung, die verschiedene APIs und Engines wie Google Speech Recognition oder WhisperAPI unterstützt
- sympy[49]: Bibliothek für symbolische Mathematik

Nachfolgend werden die im Frontend verwendeten Bibliotheken aufgeführt:

- Animation Rigging[50]: Erweiterte Animationstechniken und Kontrollen in Unity.
- Burst[51]: Compiler-Technologie-Paket zur Verbesserung der Leistung von C#-Code in Unity
- Code Coverage[52]: Überprüfung der Code-Abdeckung
- Custom NUnit[53]: Anpassung von NUnit-Tests
- Editor Coroutines[54]: Verwaltung von Coroutines
- Input System[55]: Einfachere Handhabung von Eingaben und Eingabegeräten
- JetBrains Rider Editor[56]: Integration der Rider-IDE in Unity
- Mathematics[57]: Bereitstellung mathematischer Funktionen und Algorithmen
- Newtonsoft Json[58]: Verarbeitung von JSON-Daten
- Oculus XR Plugin[59]: Bereitstellung von Anzeige- und Eingabeunterstützung für Oculus-Geräte

[45] requests-Dokumentation: `https://requests.readthedocs.io/en/latest/`, letzter Aufruf 20.09.2023

[46] scikit-learn-Dokumentation: `https://scikit-learn.org/stable/`, letzter Aufruf 20.09.2023

[47] setuptools-Dokumentation: `https://setuptools.pypa.io/en/latest/`, letzter Aufruf 20.09.2023

[48] SpeechRecognition-Dokumentation: `https://pypi.org/project/SpeechRecognition/`, letzter Aufruf 20.09.2023

[49] sympy-Dokumentation: `https://www.sympy.org/en/index.html`, letzter Aufruf 20.09.2023

[50] Unity-Dokumentation Animation-Rigging:`https://docs.unity3d.com/Packages/com.unity.animation.rigging@1.1/manual/index.html`, letzter Aufruf: 26.09.2023

[51] Unity-Dokumentation Burst: `https://docs.unity3d.com/Packages/com.unity.burst@1.6/manual/index.html`, letzter Aufruf: 26.09.2023

[52] Unity-Dokumentation Code-Coverage:`https://docs.unity3d.com/Packages/com.unity.testtools.codecoverage@1.2/manual/index.html`, letzter Aufruf: 26.09.2023

[53] Unity-Dokumentation Custom-NUnit:`https://docs.unity3d.com/Packages/com.unity.ext.nunit@1.0/manual/index.html`, letzter Aufruf: 26.09.2023

[54] Unity-Dokumentation Editor-Coroutines:`https://docs.unity3d.com/Packages/com.unity.editorcoroutines@1.0/manual/index.html`, letzter Aufruf: 26.09.2023

[55] Unity-Dokumentation Input-System:`https://docs.unity3d.com/Packages/com.unity.inputsystem@1.4/manual/index.html`, letzter Aufruf: 26.09.2023

[56] Unity-Dokumentation JetBrains-Rider-Editor:`https://docs.unity3d.com/Packages/com.unity.ide.rider@3.0/manual/index.html`, letzter Aufruf: 26.09.2023

[57] Unity-Dokumentation Mathematics:`https://docs.unity3d.com/Packages/com.unity.mathematics@1.2/manual/index.html`, letzter Aufruf: 26.09.2023

[58] Unity-Dokumentation Newtonsoft-Json:`https://docs.unity3d.com/Packages/com.unity.nuget.newtonsoft-json@3.0/manual/index.html`, letzter Aufruf: 26.09.2023

[59] Unity-Dokumentation Oculus-XR Plugin:`https://docs.unity3d.com/Packages/com.unity.xr.oculus@3.2/manual/index.html`, letzter Aufruf: 26.09.2023

- OpenXR Plugin[60]: Vereinfachung der AR/VR-Entwicklung für eine Vielzahl von Geräten
- Profile Analyzer[61]: Analyse von CPU-Daten im Profiler über mehrere Frames und Datensätze
- Services Core[62]: Definition von gemeinsamen Komponenten, die von mehreren Game-Service-Paketen verwendet werden, um die Gesamterfahrung bei der Arbeit mit dem Game Services SDK zu vereinheitlichen
- Settings Manager[63]: Umwandlung jedes serialisierbaren Felds in eine Einstellung mit einer vordefinierten Einstellungsoberfläche
- Subsystem Registration[64]: API zur Registrierung eines Subsystems, das mit der Subsystem API implementiert wurde
- Test Framework[65]: Ausführung von Edit-Mode- und Play-Mode-Tests in Unity
- TextMeshPro[66]: Bereitstellung erweiterter Textfunktionen in Unity
- Timeline[67]: Erstellung von cinematic Inhalten, Spielabläufen, Audiosequenzen und komplexen Partikeleffekten
- Tutorial Framework[68]: Anzeige von interaktiven In-Editor-Tutorials in Tutorialprojekten und Projektvorlagen
- Unity UI[69]: Entwicklung von Benutzeroberflächen für Spiele und Anwendungen
- UnityEngine.Networking[70]: Funktionalitäten zur Kommunikation zwischen Server und Client
- Version Control[71]: Nutzung von Collaborate oder Plastic SCM (beta) direkt im Unity-Editor

[60]Unity-Dokumentation OpenXR-Plugin:https://docs.unity3d.com/Manual/com.unity.xr.openxr.html, letzter Aufruf: 26.09.2023
[61]Unity-Dokumentation Profile-Analyzer:https://docs.unity3d.com/Packages/com.unity.performance.profile-analyzer@1.1/manual/index.html, letzter Aufruf: 26.09.2023
[62]Unity-Dokumentation Services-Core:https://docs.unity.com/ugs/manual/overview/manual/services-core-api, letzter Aufruf: 26.09.2023
[63]Unity-Dokumentation Settings-Manager:https://docs.unity3d.com/Packages/com.unity.settings-manager@1.0/manual/index.html, letzter Aufruf: 26.09.2023
[64]Unity-Dokumentation Subsystem-Registration:https://docs.unity3d.com/Packages/com.unity.subsystemregistration@1.1/manual/index.html, letzter Aufruf: 26.09.2023
[65]Unity-Dokumentation Test-Framework:https://docs.unity3d.com/Packages/com.unity.test-framework@1.1/manual/index.html, letzter Aufruf: 26.09.2023
[66]Unity-Dokumentation TextMeshPro:https://docs.unity3d.com/Packages/com.unity.textmeshpro@3.0/manual/index.html, letzter Aufruf: 26.09.2023
[67]Unity-Dokumentation Timeline:https://docs.unity3d.com/Packages/com.unity.timeline@1.6/manual/index.html, letzter Aufruf: 26.09.2023
[68]Unity-Dokumentation Tutorial-Framework:https://docs.unity3d.com/Packages/com.unity.learn.iet-framework@2.2/manual/index.html, letzter Aufruf: 26.09.2023
[69]Unity-Dokumentation Unity-UI:https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/index.html, letzter Aufruf: 26.09.2023
[70]Unity-Dokumentation UnityEngine.Networking:https://docs.unity3d.com/Packages/com.unity.multiplayer-hlapi@1.0/api/UnityEngine.Networking.html, letzter Aufruf: 27.09.2023
[71]Unity-Dokumentation Version-Control:https://docs.unity3d.com/Packages/com.unity.collab-proxy@1.17/manual/index.html, letzter Aufruf: 26.09.2023

- Visual Scripting[72]: Visuelle Skripterstellung
- Visual Studio Code Editor[73]: Integration des Visual Studio Code-Editors in Unity
- Visual Studio Editor[74]: Integration des Visual Studio-Editors in Unity
- XR Core Utilities[75]: Vielfältige XR-Dienstprogramme und Tools
- XR Interaction Toolkit[76]: Implementierung von XR-Interaktionen in Unity
- XR Legacy Input Helpers[77]: Tracked Pose Driver, XR Bindings, Input Bindings
- XR Plugin Management[78]: Einfache Verwaltung von XR-Plugins

[72]Unity-Dokumentation Visual-Scripting:https://docs.unity3d.com/Packages/com.unity.visualscripting@1.7/manual/index.html, letzter Aufruf: 26.09.2023

[73]Unity-Dokumentation Visual-Studio-Code-Editor:https://docs.unity3d.com/Packages/com.unity.ide.vscode@1.2/manual/index.html, letzter Aufruf: 26.09.2023

[74]Unity-Dokumentation Visual-Studio-Editor:https://docs.unity3d.com/Packages/com.unity.ide.visualstudio@2.0/manual/index.html, letzter Aufruf: 26.09.2023

[75]Unity-Dokumentation XR-Core-Utilities:https://docs.unity3d.com/Packages/com.unity.xr.core-utils@2.0/manual/index.html, letzter Aufruf: 26.09.2023

[76]Unity-Dokumentation XR-Interaction-Toolkit:https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/index.html, letzter Aufruf: 26.09.2023

[77]Unity-Dokumentation XR-Legacy-Input-Helpers:https://docs.unity3d.com/Packages/com.unity.xr.legacyinputhelpers@2.1/manual/index.html, letzter Aufruf: 26.09.2023

[78]Unity-Dokumentation XR-Plugin-Management:https://docs.unity3d.com/Packages/com.unity.xr.management@4.2/manual/index.html, letzter Aufruf: 26.09.2023

Kapitel 4

Konzept

In diesem Kapitel wird das Konzept der Projektgruppe MMI2 vorgestellt. Hierbei wird die Herangehensweise an die Konstruktion eines VR-Tischtennis-Spiels mit Instruktorfunktionen, autonomem Gegenspieler und intelligenter Assistenz erläutert. Hierfür wird zu Beginn die grundlegende Architektur erläutert und anschließend in separaten Abschnitten die vorgesehene Funktionsweise der größeren Komponenten, wie dem automatischen Gegenspieler, dargelegt. Abweichungen von diesem ursprünglichen Konzept, die sich während der Durchführung des Projekts ergeben haben, werden im Rahmen der Implementierungskapitel und des Fazits besprochen.

4.1 System-Architektur

Mit der in Abbildung 4.1 gezeigten, geplanten Architektur des Systems soll **RQ1** beantwortet werden. Der Spieler soll über das *VR-Setup*, also VR-Brille, Controller und Lighthouse-System mit der *Table-Tennis-Scene* interagieren. Diese umfasst das Tischtennisspiel in Unity VR. Das *Chatbot-Frontend* beschreibt hierbei das Interface, in der Nachrichten des Chatbots dargestellt werden und über welches der Spieler Anfragen an diesen schicken kann. Die Implementierung dieses Chatbots soll zur Beantwortung von **RQ2** beitragen. Der *AI-Opponent*, also der KI-Gegenspieler, umfasst hierbei die Funktionalitäten zur Observation des Environments, Entscheidungsfindung und zur Interaktion mit dem Environment, ungeachtet dessen, ob diese Funktionalitäten sich direkt innerhalb der *Table-Tennis-Scene* befinden oder nicht. Beispielsweise interagiert der Agent über ein Ingame-Modell mit der Szene, jedoch wird dieses nicht als separate Komponente dargestellt. *Observations* sind die Daten, welche der Agent aus dem Environment bezieht und verarbeitet, während *Actions* die Handlungen sind, die der Agent innerhalb des Environments vornimmt. Das *API Frontend* übernimmt hierbei den Versand von Spieler- und Spiel-Daten sowie Positionsdaten von Objekten an das Backend. Darüber hinaus empfängt es Antworten des Chatbots und übergibt diese an das *Chatbot Frontend* zur Ausgabe in der Szene.

Das Backend besteht aus dem *API Backend*, welches die Daten vom Frontend empfängt und an den *Aggregator* weitergibt, sowie die Antworten des Chatbots an das Frontend sendet. Der *Aggregator* handhabt die Formatierung der eingegangenen Daten für verarbeitenden Komponenten sowie die Zuordnung von Antworten zu den

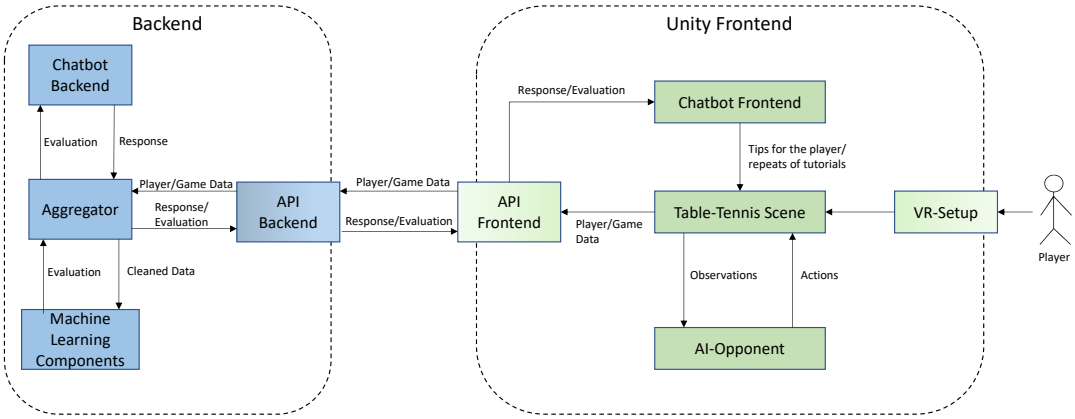


Abbildung 4.1: Eigene Darstellung der geplanten Systemarchitektur[1]

eingegangenen Datensätzen. Die in der Abbildung nur als *Machine Learning Components* betitelten Komponenten sind die verschiedenen Mechanismen zur Klassifizierung und Evaluation Spielerbewegungen auf Basis diverser Positionsdaten. Diese Evaluationsergebnisse werden über den *Aggregator* an das *Chatbot Backend* gegeben, welches eine angemessene Antwort bzw. Empfehlung für den Spieler generiert. Diese wird über den *Aggregator* und das *API Backend* zurück an das Frontend geschickt.

4.2 Konzept des Spiels

Bei dem Spiel soll es sich um eine VR-Umsetzung von Tischtennis handeln. Um die Komplexität des Spiels selbst den Anforderungen entsprechend skalieren zu können und um die Verfügbarkeit von Spieldaten zu gewährleisten, wird von der Nutzung eines existierenden Spiels als Basissystem abgesehen. Dementsprechend erfolgt die Implementierung des Spiels durch die Projektteilnehmer. Die Verwendung von existierenden Assets für Spielkomponenten, wie Tischtennisplatte oder Schläger, wird aus Gründen der Zeitersparnis als sehr vorteilhaft angesehen. Ziel ist ein einfacher, geschlossener, beleuchteter Raum mit einer Tischtennisplatte, in welchem dem Spieler der Schläger und Ball präsentiert wird, sodass dieser beide aufnehmen kann und gegen einen KI-Gegenspieler Tischtennis spielen kann. Hierbei soll dem Spieler die Option gegeben werden, zwischen der eigentlichen Spielumgebung und einer Übungsumgebung wechseln zu können. Besagte Übungsumgebung soll vom grundlegenden Setup analog zur Spielumgebung sein, nur dass hier der Spieler Tutorials hören kann, in denen ihm die Funktionalitäten der Umgebung und Spielzüge wie der Aufschlag erklärt werden. Diese Erklärungen sollen vom Chatbot unterstützt werden. Während eines

[1]Diese Abbildung ist ebenfalls im Proposal der PG MMI2 vorzufinden

laufenden Spiels soll der Punktestand verfolgt und für den Spieler sichtbar angezeigt werden. Die Implementierung der Spiellogik und aller damit zusammenhängenden Grafikelemente erfolgt in Unity sowie mithilfe von *SteamVR*.

Während des Spiels sollen Schläger-Bewegungen des Spielers evaluiert werden und, sollten Schläge zu einem Fehler führen, soll dem Spieler die Option geboten werden, Tutorials für diese Bewegungen zu wiederholen. Außerdem soll dem Spieler die gesamte Zeit über die Möglichkeit gegeben sein, einen Chatbot um Hilfe zu fragen. Um den *cognitive load* des Spielers zu reduzieren, soll der Aufruf des Chatbots nicht über Knöpfe, sondern alternative Mechaniken, wie einem Sprachbefehl erfolgen.

Die Implementierung des KI-Gegenspielers (im Folgenden oftmals auch der*Bot*) soll mithilfe des *Unity ML-Agents* Toolkits erfolgen. Der Aktionsraum des Bots soll die Bewegung entlang der Tischtennisplatte und die Bewegung des Schlägers im dreidimensionalen Raum nicht überschreiten. Es wird also von Rotationen des Körpers und komplexen Rotationen des Schlägers, also Rotationen des Schlägers entlang aller drei Achsen, abgesehen. Sollte dieser Aktionsraum nicht ausreichen, um den Bot erfolgreich den Ball an den Spieler zurückspielen lassen zu können, so sollen entsprechende Hilfsfunktionalitäten implementiert werden. Besagte Hilfsfunktionalitäten sollen ebenfalls eine Handlungsoption zur Reduktion der Komplexität und somit Dauer und Ressourcenintensität des Trainings darstellen. Unter Hilfsfunktionalitäten sind Methoden wie beispielsweise eine statische Beschleunigung des Balls in Richtung fixer Punkte, zur Approximation eines Aufschlags durch den Bot, zu verstehen. Ziel des Bots soll es sein, den Ball erfolgreich über das Netz zu spielen, ohne unzulässige Spielzustände herbei zu führen. Der Bot soll keine besonders hohe Herausforderung für den Spieler darstellen. Der Bot soll in der Lage sein, mithilfe von Sensoren Daten über seine eigenen Positionsdaten, die seines Schlägers und die des Balls zu erhalten. Um die Komplexität dieser Observationen möglichst niedrig zu halten, sollen Vektor-Sensoren verwendet werden, die Positionsvektoren als direkte Inputs erhalten. Die Reward-Verteilung soll eventbasiert erfolgen und sich an den Regeln des Tischtennis orientieren. Genaue Beschreibungen des Handlungs- und Observationsraumes, sowie der Reward-Verteilung, erfolgt in Abschnitt 5.1.5. Ein KI-Gegenspieler nach diesem Konzept soll zu einer Antwort auf **RQ3** beitragen.

4.3 Konzept des Backends

Das Backend soll primär aus zwei Microservices und einer aggregierenden Instanz bestehen. Dieser Aggregator soll eintreffende Daten synchronisieren und Antworten den Daten zuordnen. Ebenso soll er die Daten reinigen. Bei den beiden Microservices handelt es sich um die Komponenten zur Evaluation der Spiel- und Spielerdaten und den Chatbot. Der Chatbot soll dabei mit dem Tool *Rasa*[25] aufgesetzt und auf die Nutzung von Tischtennis-spezifischem Vokabular trainiert werden. Für die Evaluationskomponenten sollen Toolkits wie *PyTorch*[43] oder *TensorFlow*[2] für das Training von ML-Methoden genutzt werden. Hierbei wird von der Komplexität der Evaluationsprozesse abhängig gemacht, welche Methoden genutzt werden. Ziel soll es sein, beim Schlag eines Spielers zu erkennen, um was für einen Schlag es sich handelt und ob dieser eine Fehlerquelle darstellt. Als Fehler gilt hierbei zunächst, ob ein Schlag zu einem Punkt für den Gegner geführt hat.

[2]Tensorflow:`https://www.tensorflow.org/`, letzter Aufruf: 28.09.2023

Für das Training der ML-Komponenten zur Klassifikation und Evaluation von Schlägen wird ein umfangreicher Trainings- und Testdatensatz vonnöten sein. Daher wird die Konstruktion dieser Komponenten zur Beantwortung von **RQ4** genutzt, indem dieser Datensatz zum größten Teil aus synthetischen Daten besteht.

4.4 Konzept für die API und den Datenaustausch zwischen Front- und Backend

Beim Design der Dateninfrastruktur zwischen Front- und Backend ist der Kommunikationstyp essenziell. Es gilt also abzuwägen, ob die Kommunikation synchron oder asynchron erfolgen soll und ob einer oder mehrere Empfänger vorliegen. Bereits während der Konzeptionierung ist davon auszugehen, dass nur zwischen Client und Server kommuniziert wird, womit also jede Nachricht nur einen Empfänger hat. Über den Aspekt der Synchronität hinaus werden zwei Toolkits zum Aufsetzen der Architektur in Betracht gezogen: *ZeroMQ* in Kombination mit dessen C#-Port *NetMQ* oder *Flask*. Welches Toolkit im Rahmen der Implementierung genutzt wird, steht zum Zeitpunkt der Konzeptionierung noch nicht fest.

Kapitel 5

Implementierung

In diesem Kapitel ist dokumentiert, wie die Projektteilnehmer das in Kapitel 4 vorgestellte Konzept umgesetzt haben, wobei Abweichungen von besagtem Konzept aufgezeigt und erläutert werden. Hierzu wird in Abschnitt 5.1 erläutert, wie das Tischtennis-Spiel selbst aufgebaut ist. Danach wird in Abschnitt 5.3 auf den Aufbau der Schnittstelle zwischen Front- und Backend eingegangen, wobei sich auf den Prozess Datenaustauschs fokussiert wird. Letztlich wird in Abschnitt 5.4 die Architektur und Aufmachung der Machine Learning Komponenten des Backends beleuchtet.

5.1 Implementierung des VR-Tischtennis-Spiels

Im Folgenden wird erläutert, wie das Tischtennisspiel aufgebaut ist und mithilfe von Unity implementiert wurde. Dafür wird zunächst darauf eingegangen, wie die Tischtennisregeln umgesetzt sind (Abschnitt 5.1.1). Danach wird in Abschnitt 5.1.2 die Gestaltung des Trainingsraumes beschrieben. Es folgt eine Erläuterung des Hitbox-Problemes (Abschnitt 5.1.3) und der Interaktion mit virtuellen Objekten (Abschnitt 5.1.4). Abschließend werden die beiden Implementierungsansätze des Bots beschrieben (Abschnitt 5.1.5 und 5.1.6), welche im darauf folgenden Kapitel verglichen und diskutiert werden.

5.1.1 Umsetzung der Tischtennisregeln

Die Implementierung von Tischtennisregeln in einem VR-Tischtennis-Spiel erweist sich aufgrund der Beschränkungen von VR-Anwendungen als herausfordernde Aufgabe. In der realen Welt sind die Tischtennisregeln äußerst detailliert und komplex, wobei Aspekte wie die Aufschlagvorschriften oder Netzberührungen präzise festgelegt sind. In einer VR-Umgebung, in der die physische Realität virtuell nachgebildet wird, ergeben sich jedoch Hindernisse bei der exakten Umsetzung dieser Regeln. Die Schwierigkeit besteht darin, dass VR-Anwendungen zwar eine immersive Erfahrung bieten, jedoch einige physikalische Eigenschaften der realen Welt nicht vollständig replizieren können. Aus diesem Grund sind häufig Anpassungen oder sogar die Abstraktion von bestimmten Tischtennisregeln erforderlich, um ein reibungsloses und unterhaltsames Spielerlebnis zu gewährleisten. Solche Anpassungen können die Behandlung von Netz- und Tischberührungen oder andere komplexe Regelaspekte um-

fassen. Im Laufe des folgenden Abschnittes sollen die nuancierten Abstraktionen der Regeln des Deutschen Tischtennis Bundes, wie sie in Kapitel 3.1.6 beschrieben wurden, näher erläutert werden. Insbesondere wurden die folgenden Aspekte der realen Tischtennisregeln abstrahiert und ausgelassen:

1. Der Seitenwechsel nach jedem Satz und im Verlängerungssatz wurde nicht berücksichtigt. Ebenso entfällt die Auslosung des Aufschlagrechts und der Seitenwahl zu Beginn eines Spiels. Diese Regeln, obwohl integraler Bestandteil des traditionellen Tischtennisspiels, wurden in der Projektumsetzung in Kapitel 1.1 nicht als essenziell erachtet, da sie nicht direkt in den Aktions-Feedback-Loop des Spiels eingreifen.

2. Die Regel, den Ball beim Aufschlag aus der geöffneten Hand hochzuwerfen, wurde in der Spielentwicklung zunächst berücksichtigt, jedoch später abstrahiert, um das Gameplay zu vereinfachen. Im VR-Spiel wird der Ball nun ohne den Zwischenschritt des Hochwerfens direkt aus der Hand gespielt. Hierbei schwebt der Ball versetz neben der Hand, um Kollisionen der Controller zu vermeiden.

3. Der implementierte Bot führt den Aufschlag aus, ohne auf die Rückmeldung des Spielers zu warten, was aus zeitlichen Gründen und aufgrund der Priorisierung von anderen Aspekten so umgesetzt wurde. Diese Abweichung von der realen Regel wird als nicht spielkritisch betrachtet.

4. Die Regel, dass der Aufschlag nicht verdeckt sein darf, wird im VR-Tischtennis-Spiel nicht beachtet, da eine praktikable Lösung für die Umsetzung dieser Regel fehlt. Zudem wird dieser Aspekt als eine seltene Ausnahme betrachtet, die keinen wesentlichen Einfluss auf die Benutzerfreundlichkeit des Systems hat.

5. Ebenso wird die Regel, dass der Ball den Körper nicht berühren darf, bevor er gespielt wird, nicht in der VR-Implementierung berücksichtigt. Aufgrund der potenziell geringen Nutzererfahrung mit VR Systemen, ist die Wahrscheinlichkeit einer Kollision des Körpers und Balls nicht gering. Daher wurde im Zuge der Nutzerfreundlichkeit die diese Regel entschärft. Stattdessen darf der Ball lediglich den Boden nicht berühren.

Ansonsten werden die Schläge entsprechend der in Kapitel 3.1.6 beschriebenen Regeln als legal und illegal definiert, wie es in Abbildung 5.1 dargestellt ist. So können die essenziellen Spielregeln, die einen Ballwechsel ausmachen, umgesetzt werden.

Abbildung 5.1 zeigt die finale Version des sogenannten `GameStateTrackers`. Der GamestateTracker dient den derzeitigen Fluss des Spieles nachzuvollziehen und eventuelle Regelverstöße zu ermitteln und entsprechend zu behandeln. Hierbei ist der GamestateTracker als Singleton umgesetzt, um einen einfachen Zugang auf spiel-relevante Daten wie den derzeitigen Punktestand oder bspw. das Aufschlagrecht von beliebiger Stelle im Code zu behalten. Um die Ermittlung von illegalen Spielzügen zu vereinfachen, ist der GamestateTracker als leicht abgeänderter Mealey-Automat aufgebaut, in welchem jeder legale Spielzug einen dedizierten Zustandswechsel erzeugt. Illegale Spielzüge sind im Zuge der vereinfachten Darstellung hier nicht näher dargestellt. Hierbei definiert ein Spielzug ein Auftreffen des Balls auf eine bestimmte Oberfläche wie dem Schläger des Spielers, dessen Seite der Tischtennisplatte oder bspw. den Boden. Sollte in einem Zustand eine illegale Eingabe getätigt werden, wird der Automat auf einen seiner beiden Initialzustände zurückgesetzt. Hierbei entscheidet das derzei-

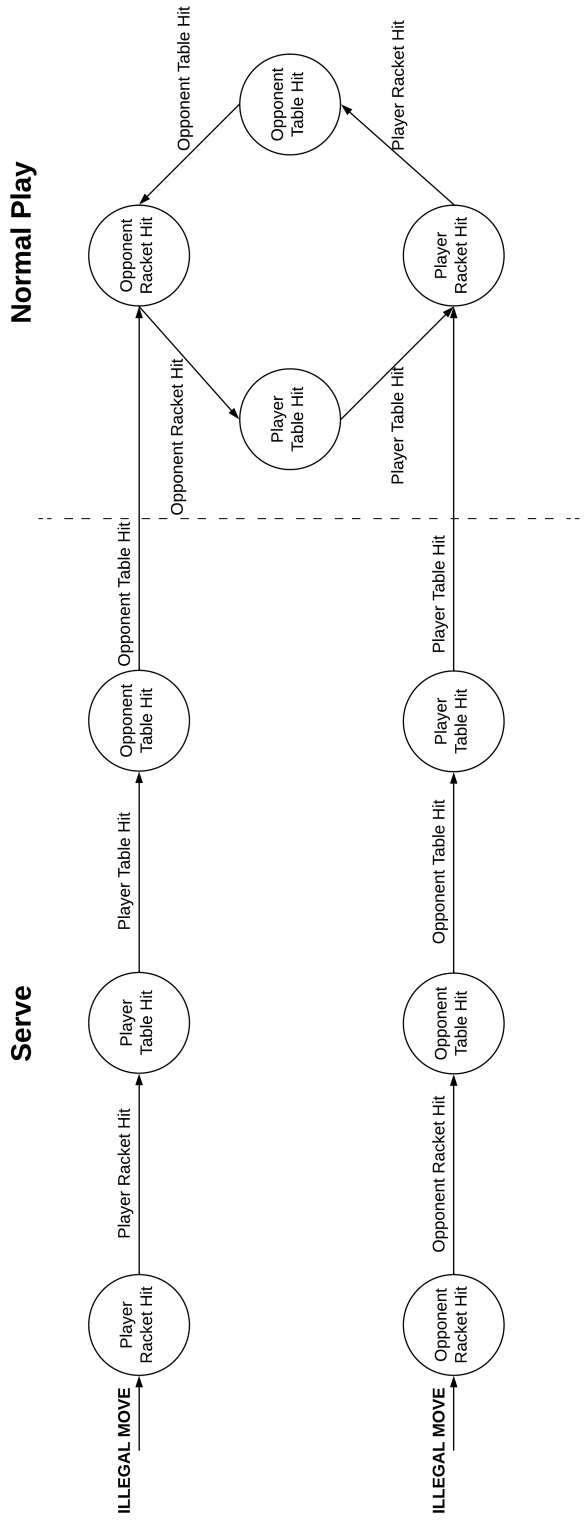


Abbildung 5.1: GamestateTracking zur Modellierung des Spielverlaufs und zur Definition von legalen und illegalen Schlägen. Eigene Darstellung

tige Aufschlagrecht, welcher der jeweiligen Initialzustände gewählt wird. Aufgrund der zuvor genannten Komplexität der umzusetzenden Tischtennisregeln ist der GamestateTracker jedoch mehrere Iterationen durchlaufen, welche aufgrund von diversen Schwächen und Unzulänglichkeiten der Modelle weiterentwickelt wurden.

Anhang F.1 zeigt den ersten Entwurf des GamestateTracker, welcher aber aufgrund seiner hohen Komplexität und Fehleranfälligkeit in seiner Umsetzung verworfen wurde.

Anhang F.2 zeigt den zweiten Entwurf des GameStateTrackers, welcher im Gegensatz zu seiner ursprünglichen Implementierung eine starke Reduktion der vorliegenden Komplexität aufweist. Aufgrund der bisher immer noch fehlenden klaren Abtrennung zwischen dem Aufschlag des Spielers und dem des Gegenspielers ergaben sich auch hier große Fehlerpotenziale in der Umsetzung, welche zu der bereits in Abbildung 5.1 vorgestellten finalen Version des GameStateTrackers geführt haben.

5.1.2 Gestaltung des Trainingsraums

Im Spiel gibt es zwei verschiedene Räumlichkeiten: den „Spielraum“ und den „Trainingsraum“ (siehe 5.2). Im Spielraum spielt der Nutzer gegen den Bot, während der Trainingsraum dem Spieler die Möglichkeit bietet, alleine gegen eine Tischtennisplatte zu spielen und damit die Tischtennisfähigkeiten zu verbessern. Außerdem können Spieler im Trainingsraum mit dem Chatbot interagieren und ihre Fragen zu den Herausforderungen verschiedener Schläge stellen und alleine gegen die Tischtennisplatte das Gelernte aus den Gesprächen mit dem Chatbot üben.

Für die Implementierung des Trainingsraums wurde ein Button erstellt, der es den Benutzern ermöglicht, zwischen den beiden Räumen zu wechseln. Da der Trainingsraum sowohl gemeinsame Objekte mit dem Spielraum als auch Objekte, die ausschließlich zum Trainingsraum gehören, hat, wurden alle Objekte in Unity zur effizienten Verwaltung in drei Hauptgruppen unterteilt: Die erste Gruppe enthält Objekte, die ausschließlich dem Spielraum zugeordnet sind, die zweite Gruppe beinhaltet Objekte, die ausschließlich für den Trainingsraum gedacht sind, und die dritte Gruppe enthält Objekte, die in beiden Räumen benötigt werden. Beim Drücken des Buttons werden die Gruppen „Trainingsraum“ und „Spielraum“ aktiviert bzw. deaktiviert. Befindet sich der Spieler im Spielraum und drückt den Button „Spielraum/Trainingsraum“, werden die Objekte, die ausschließlich zum Spielraum gehören, deaktiviert, während die Objekte der Gruppe Trainingsraum aktiviert werden. Entsprechend werden, wenn sich der Spieler im Trainingsraum befindet und denselben Button drückt, die Objekte des Trainingsraums deaktiviert und die Objekte des Spielraums aktiviert. Die Gruppe der „common items“ mit den Objekten, die in beiden Räumen benötigt werden, bleibt nach dem Raumwechsel immer aktiviert. Die Objekte in den drei Gruppen sind entsprechend unterteilt:

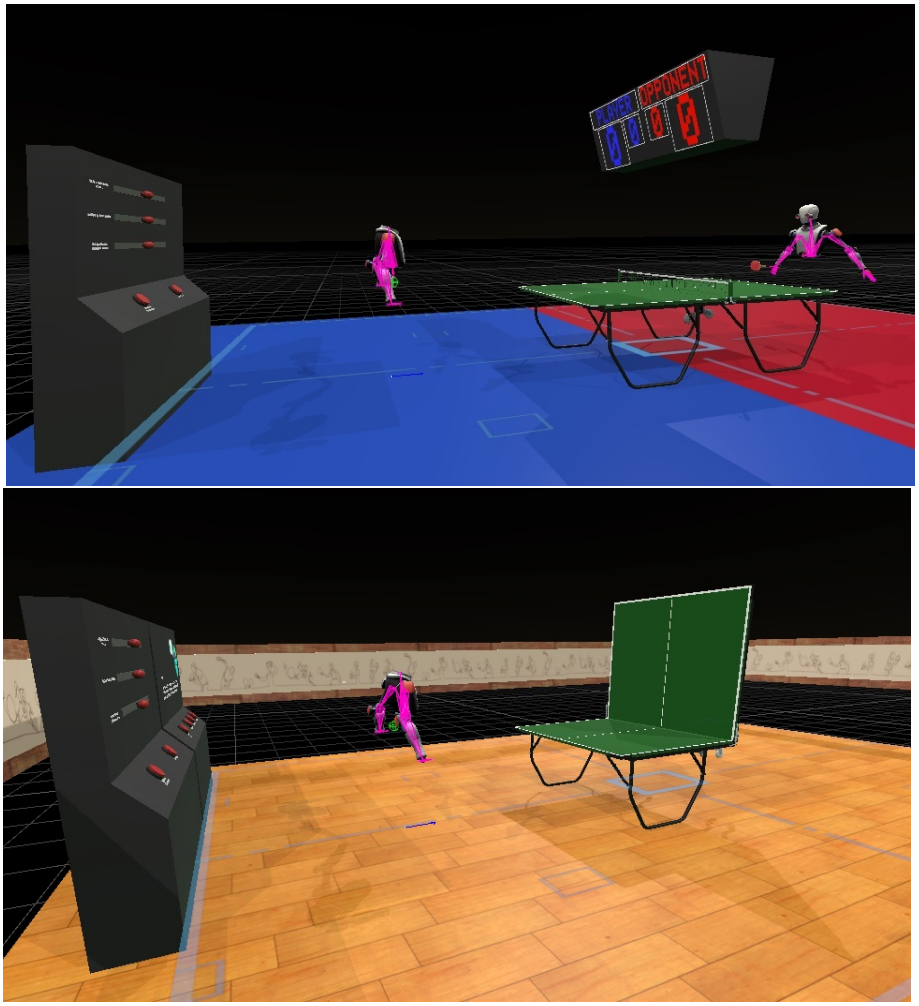


Abbildung 5.2: Oben: Spielraum – Unten: Trainingsraum

Gemeinsame Objekte in beiden Räumen:

- Spieler
- Umgebung
- Kollisionsdetektoren
- Interaktive Elemente: Buttons, die es dem Spieler ermöglichen, zwischen den beiden Räumen (Spielraum und Trainingsraum) zu wechseln, das Spiel neu zu starten oder die Schläger-Rotation anzupassen.
- Beobachter: Kameras, die den Spielverlauf verfolgen und für die Studie relevante Daten aus verschiedenen Perspektiven sammeln.
- Game Management: Elemente wie `GameTracker`, `GameDataSender`, `BallPickUpManager`, `AudioManager` und `GameModifierManager`
- Tischtennistisch auf der Seite des Spielers

Nur im Trainingsraum:

- Chatbot-Schnittstelle
- Einseitig hochgeklappte Tischtennisplatte
- Individuelle Dekoration und Bodenbelag: Der Trainingsraum hat eine andere Atmosphäre mit individueller Dekoration und Bodenbelag, die sich vom Spielraum unterscheiden (siehe Abbildung 5.3). Für den Boden wurde Holz verwendet, und zur Wanddekoration wurden handgezeichnete „Line Art"-Darstellungen von VR und Tischtennisschlägen hinzugefügt.

Nur im Spielraum:

- KI-Gegenspieler
- Punkteanzeige
- Boden des Spielraums
- Tischtennistisch auf der Seite des Bots

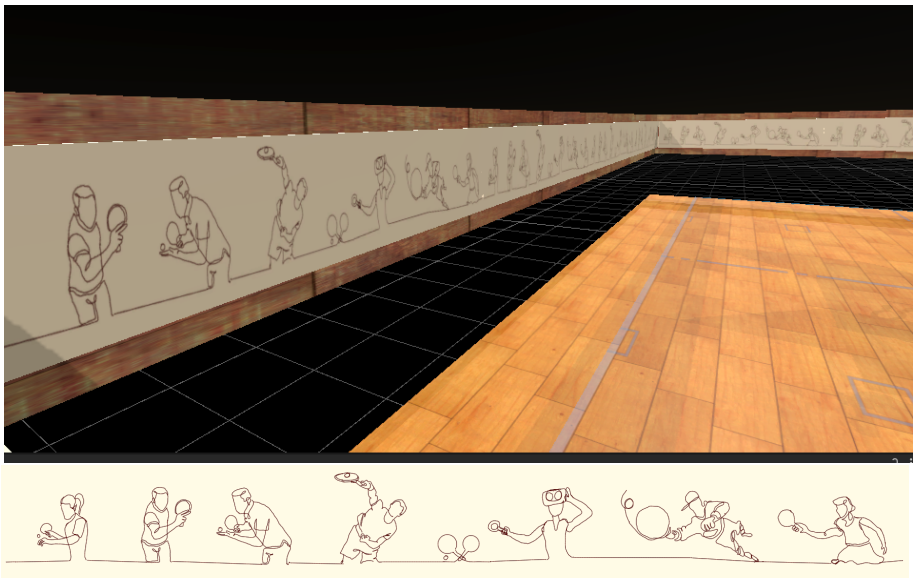


Abbildung 5.3: „Line Art“ Zeichnung zur Wanddekoration im Trainingsraum

Chatbot-Schnittstelle im Trainingsraum

Die Chatbot-Schnittstelle (siehe 5.4) ist das primäre Element, welches den Trainingsraum vom Spielraum unterscheidet. Diese Schnittstelle besteht aus verschiedenen Buttons, die jeweils eine bestimmte Funktion haben:

- **„Mit Chatbot sprechen“-Button:** Dieser Button ermöglicht es den Nutzern, ein Gespräch mit dem Chatbot zu beginnen. Hier können Fragen gestellt und Ratschläge zur Verbesserung der Tischtennisfähigkeiten erhalten werden.
- **Tutorials für Rückhand, Vorhand und Aufschlag:** Neben dem „Mit Chatbot sprechen“-Button stehen im Trainingsraum drei weitere Buttons zur Verfügung, die Tutorials für verschiedene Schlagtechniken anbieten. Bei diesen Tutorials handelt es sich um vom Chatbot vorgetragene Anleitung und dienen dazu, die Schlagtechniken zu erlernen und zu verfeinern. Es gibt separate Buttons für die Rückhand, die Vorhand und den Aufschlag, mit denen die Nutzer auf spezifische Tutorials zugreifen können.

Darüber hinaus wurde ein Text zur Vorstellung des Chatbots sowie ein Chatbot-Logo[1] verwendet, um auf den ersten Blick einen Eindruck von Interaktion zu erzeugen.

Zur Implementierung der Chatbot-Schnittstelle musste nach dem Drücken jedes Buttons eine Anfrage an den Chatbot im Backend gesendet werden. Diese Anfrage enthält eine Nachricht und eine URL, wie im Kapital 5.3 genauer erklärt ist.

- Nach dem Drücken des Buttons „Mit Chatbot sprechen“ wird eine Nachricht vom Typ String, „Hallo“, an die Route `/chatbot` der API gesendet.
- Für die Tutorials Buttons wird eine Nachricht an die `/tutorial` API Route ge-

[1] FREEP!K:`https://www.freepik.com/search?format=search&query=chatbot`, letzter Aufruf 25.09.2023

Abbildung 5.4: Chatbot-Schnittstelle (ausschließlich im Trainingsraum)

sendet. Diese Nachricht enthält Schlusswörter wie „Vorhand“, „Rückhand“ oder „Aufschlag“, um das gewünschte Tutorial zu starten.

Nach der Implementierung der Chatbot-Schnittstelle ergab sich ein unvorhergesehenes Problem: Nachdem ein Button, beispielsweise für das „Service“-Tutorial, gedrückt und das Tutorial gestartet wurde, treten Konflikte auf, wenn ein weiteres Tutorial ausgewählt oder der Button zum ‚Mit Chatbot sprechen“ gedrückt wird. Infolgedessen werden die entsprechenden Audiodateien gleichzeitig wiedergegeben und der Benutzer kann sich aufgrund der Audiokonflikte nicht auf eine einzelne konzentrieren.

Um dieses Problem zu lösen, wurden zwei Lösungsansätze für jeden Teil, Tutorials und das Gespräch mit dem Chatbot, in Betracht gezogen:

1. Statt Anfragen an das Backend für Tutorials zu senden, wurden die Tutorials als MP3-Dateien in Unity importiert, um deren Start und Stopp direkt zu steuern.
2. Um die Gespräche mit dem Chatbot zu starten und zu stoppen, wurde auf der Backend-Seite eine neue Funktionalität für den Chatbot implementiert, die es

ermöglicht, den Chatbot stoppen, wenn der Benutzer ein neues Tutorial startet oder den Trainingsraum verlässt.

Diese zwei Lösungen ermöglichten es, den Konflikt bei dem Abspielen der Tutorials und des Chatbot-Dialogs zu lösen. Bei jedem Button-Druck wird geprüft, ob bereits ein Tutorial abgespielt wurde oder ein Dialog aktiv ist. Ist das der Fall, werden alle abgespielten Tutorials und auch der Chatbot gestoppt, um anschließend das gewünschte Tutorial zu starten. Außerdem werden alle Tutorials und Gespräche mit dem Chatbot beendet, sobald der Trainingsraum verlassen wird. Dies ist erforderlich, da im Spielraum weder Tutorials noch Chatbot-Gespräche durchgeführt werden müssen.

5.1.3 Hitbox Issues

Im Laufe der Entwicklung haben sich eine Vielzahl an Hürden in Bezug zur Verwendung der Unity eigenen Physik-Simulation offenbart, deren Überwindung besonders viel Zeit und Arbeit in Anspruch nahm. Aufgrund der Natur eines SGs ist es erstrebenswert, eine möglichst realitätsnahe Simulation der Spielobjekte eines Tischtennisspiels nachzuahmen. Hierzu zählen beispielsweise eine realistische und vorhersagbare Flugbahn des Balls, eine glaubwürdige Reibungsinteraktion der verschieden griffigen Seiten des Schlägers mit dem Ball, sowie die Erstellung eines robusten und zuverlässigem Restsystem von Unity. Dies ist jedoch nicht zur Zufriedenstellung der Projektgruppe erfolgt, da sich der Tischtennisball durch den Schläger des Spielers bewegen konnte. Um diese Probleme technisch besser aufarbeiten zu können, ist zunächst eine gewisse Begriffs- und Technologieklärung erforderlich.

Collision Mode

Unity verwendet zur Verwaltung von physikalischen Eigenschaften eines Objektes die *Rigidbody*-Komponente, welche jedem Objekt zugewiesen werden kann. Ein Rigidbody enthält unter anderem die grundlegenden Informationen über die Masse eines Objektes, dessen Kollisionskörper, Luftwiderstand, sowie auch dessen Position und Geschwindigkeitsvektoren. Außerdem enthält ein Rigidbody auch tiefergehende Datenpunkte wie den Kollisionsmodus (Collision Mode), der bestimmt, wie die Kollision zweier Rigidbody innerhalb Unitys Physicsengine behandelt werden muss. Der gewählte Kollisionsmodus bestimmt die Kosten und Genauigkeit einer Kollisionsbehandlung und kann das Verhalten selbiger drastisch beeinflussen. Hierbei bestimmt also der Anwendungsfall die Verwendung des entsprechenden Modus. [2]

Initial wird jede Kollision in Unity sogenannt *diskret* behandelt. Das heißt, dass, sobald die Physicsengine einen weiteren Zeitschritt berechnet, geprüft wird, ob in diesem Moment zwei Rigidbody-Objekte miteinander kollidiert sind. Diese Art der Kollisionsberechnung ist einfach und daher kosteneffektiv. Jedoch wird hier die Zeitspanne zwischen zwei Zeitschritten der Physicsengine außer Acht gelassen, sodass ein Körper, der sich schnell genug bewegt, durch einen anderen statischen oder sich bewegenden Körper hindurchbewegen könnte, ohne miteinander zu kollidieren. Dieses Verhalten wird umgänglich als *Tunneling* bezeichnet und ist für den hier beschriebenen Anwendungsfall eines SGs nicht wünschenswert, da er nicht die Realität wider-

[2] Unity-Dokumentation Rigidbody.collisionDetectionMode: `https://docs.unity3d.com/ScriptReference/Rigidbody-collisionDetectionMode.html`, letzter Aufruf 26.09.2023

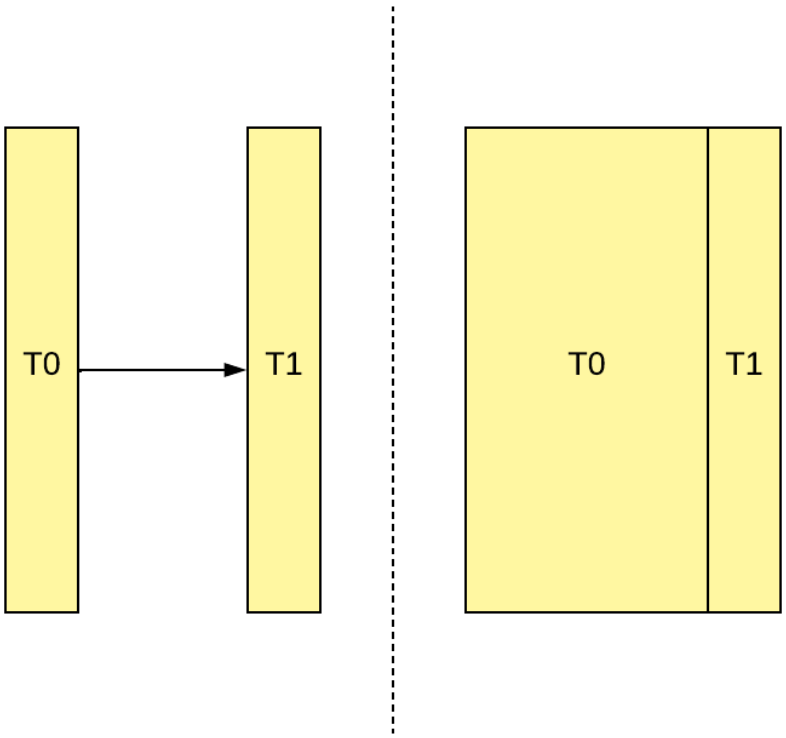


Abbildung 5.5: Darstellung der Ausdehnung eines Kollisionskörper über zwei Zeitschritte hinweg

spiegelt [3]. Da so schnell gespielte Tischtennisbälle durch einen Schläger oder Tisch fallen könnten, obliegt es hier dem Entwickler, einen passenderen Kollisionsmodus auszuwählen und dessen Kosten und Nutzen in Anbetracht des restlichen Projektes abzuwägen.

Für den hier beschriebenen Anwendungsfall eines SGs bietet es sich an, eine kontinuierliche Kollisionserkennung zu verwenden, um ein solches Tunneling zu verhindern. Unity bietet verschiedene kontinuierliche Kollisionsmodi an, welche unterschiedliche Vor- und Nachteile bieten. Für diese Wahl spielt nicht nur die Geschwindigkeit, mit der sich ein Objekt durch den Raum bewegt, eine Rolle, sondern zusätzlich dazu auch die Rotationsgeschwindigkeit des Objektes.

Eine der kontinuierlichen Kollisionsbehandlungsmethoden ist die sogenannte *Sweep-based CCD*. Hierbei wird die Kollisionsbox eines Objektes zur Zeit $T0$ auf die Position des Objektes zur Zeit $T1$ ausgedehnt, sodass diese erweitert wird. Abbildung 5.5 veranschaulicht dieses Phänomen anhand eines zweidimensionalen Kollisionskörpers. Hier ist zu beachten, dass nur die Geschwindigkeiten der kollidierenden Objekte beachtet werden. Die Rotationsgeschwindigkeiten der Objekte werden hierbei vernach-

[3] Unity - Manual: Continuous collision detection (CCD): `https://docs.unity3d.com/Manual/ContinuousCollisionDetection.html`, letzter Aufruf 29.09.2023

lässigt. Dies führt dazu, dass Objekte, welche mit einer hohen Winkelgeschwindigkeit rotieren, durch andere Objekte tunneln können. In Abbildung 5.6 ist dieser Sachverhalt dargestellt.[4]

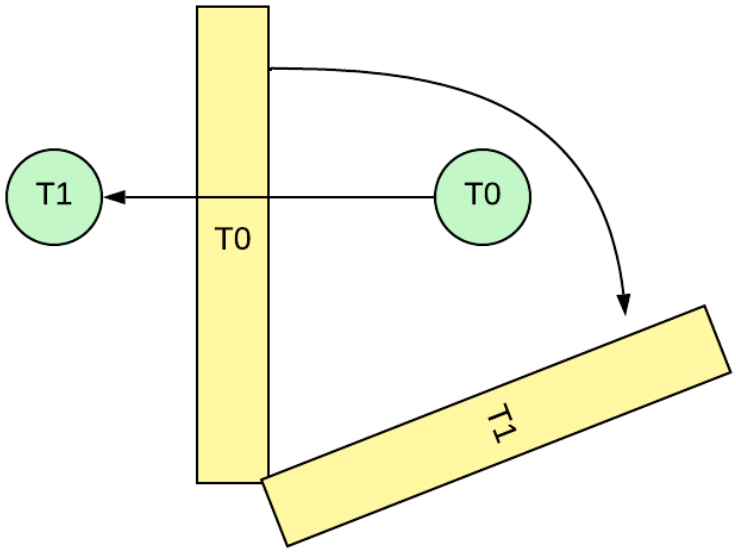


Abbildung 5.6: Beispiel von Nachteilen von Sweep Based CCD. Eigene Darstellung

Eine weitere Kollisionsbehandlungsmethode ist die sogenannte *Speculative CCD*. Hierbei werden mögliche Kollisionspunkte eines Objektes anhand seiner Bewegungskomponenten, d.h. Winkel- und absolute Geschwindigkeit, sowie anhand ihrer Distanz zum Objekt zur Zeit $t0$ ermittelt. Sobald nun Kollisionspunkte erkannt wurden, werden die entsprechenden Kontaktflächen berechnet. Jedoch bilden die entstehenden Kontaktflächen die korrespondierenden Kontaktkörper nicht genau nach, sodass bspw. aus einer vertikalen Wand eine schräge Kontaktfläche ermittelt wird. Warum dies der Fall ist, wird von der Unity Dokumentation nicht weiter erklärt. Jedenfalls resultiert diese Ungenauigkeit in Kollisionsereignissen, welche unter normalen Umständen nicht stattfinden könnten. Bewegt sich beispielsweise ein Objekt zur Zeit $t0$ orthogonal über die gemeinsame Kante zwischen zwei weiteren Kollisionskörpern $b0$ und $b1$, ist es möglich, dass sich das ursprüngliche Objekt nicht wie erwartet über die Naht hinweg bewegt, sondern von dieser abgelenkt wird. Abbildung 5.7 zeigt diese Ablenkung anhand der korrekten Kontaktfläche $c1$ und der inkorrekten Kontaktfläche $c0$. Hierbei ist klar zu erkennen, dass die neue Position des Objektes zur Zeit $t1'$ nicht dem erwarteten Weg folgt. [4]

Bewegen sich jedoch alle Kontaktpunkte, während sie zusätzlich rotieren, entstehen weitere Tunneling Probleme, insofern sich eine Naht zwischen den bewegenden Kontaktpunkten befindet.

[4]Unity-Dokumentation Continuous Collision Detection: `https://docs.unity3d.com/Manual/ContinuousCollisionDetection.html`, letzter Aufruf 24.08.2023

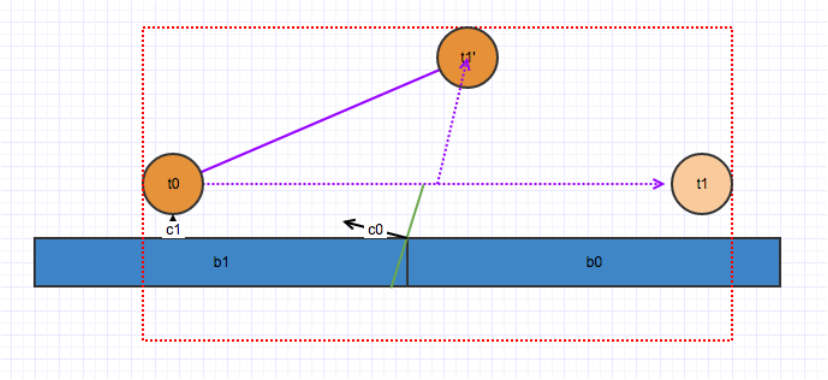


Abbildung 5.7: Ablenkung durch Speculative CCD [Quelle: [4]]

Nach dieser tiefgehenden Einleitung ist es möglich, das Tunneling-Verhalten des Balls zu erklären. Da der Schläger aus zwei aneinanderliegenden Hitboxen besteht, welche die Vorder- und Rückseite des Schlägers für die die Kollisionsbehandlung differenziert, existiert also eine sich bewegende Naht. Zusätzlich wird der Schläger aufgrund der bogenartigen Bewegung eines Schlags nicht nur durch den Raum bewegt, sondern zusätzlich rotiert. Abschließend wirken sich die verhältnismäßig kleinen Hitboxen der Objekte erschwerend auf die Kollisionsbehandlung aus. Diese Umstände haben zur Entscheidung geführt, Speculative CCD zu verwenden, da hier durch Funktionstests realitätsnahe Ergebnisse erzielt werden konnten. Aufgrund der vorher beschriebenen Umstände und den Schwächen der gewählten Kollisionsbehandlung ist es für den Ball weiterhin möglich, durch den Schläger zu tunneln. Dieses Problem lässt sich jedoch nicht ohne einen grundlegenden Eingriff in den Aufbau der Schläger-Hitboxen beheben. Daher entschließt sich die PGMMI2 diesen Bug nicht zu beheben und ihn stattdessen hier zu dokumentieren.

5.1.4 Interaktion mit virtuellen Objekten

Einer der bemerkenswertesten Aspekte von VR-Anwendungen ist die Vielzahl an Interaktionsmöglichkeiten, die es den Nutzern ermöglichen, nahezu nahtlos mit den virtuellen Objekten in ihrer Umgebung zu interagieren. Die PGMMI2 hat sich nach Bewertung der verschiedenen für Unity verfügbaren VR-Frameworks für das SteamVR Framework entschieden. Da dieses aufgrund seiner hohen Kompatibilität mit verschiedenen Ein- und Ausgabegeräten sowie bereits etablierten Interaktionsmethoden eine einfache Entwicklung von VR-Anwendungen verspricht. So bietet das SteamVR Framework bereits vorgefertigte Lösungen für diverse sich bewegende Knöpfe, generelle Fortbewegungsmöglichkeiten sowie vorgefertigte Lösungen, um mit virtuellen Gegenständen zu interagieren [5]. Hierzu zählt beispielsweise die von SteamVR bereitgestellte `Interactable`-Komponente, welche es einem Spieler ermöglicht, dessen virtuelle Hand mittels eines *Greifen*-Knopfes am Controller mit dem ausgewählten Objekt auszutauschen [5]. Dies sorgt dafür, dass der Spieler den Eindruck erhält, ein Objekt gegriffen zu haben. Hierbei ist es wichtig zu unterscheiden, dass kein Objekt tatsächlich gegriffen wird und sich dementsprechend in einer Hand befindet, sondern die

[5]Steam VR API Dokumentation Interaction System: `https://valvesoftware.github.io/steamvr_unity_plugin/articles/Interaction-System.html`, letzter Abruf: 29.09.2023

Hand des Spielers durch das entsprechende Objekt ausgetauscht wird. Dies sorgt dafür, dass die Funktionalitäten der Hand erhalten bleiben, anstelle der Hand jedoch ein neues Objekt mit neuen Kollisionsboxen erscheint. Da dieser Ansatz einfach umzusetzen ist, hat sich die PGMMI2 initial dazu entschieden, Gegenstände wie den Tischtennisschläger oder Ball als tatsächliche im Raum befindliche Gegenstände darzustellen und diese mit der entsprechenden Interactable-Komponente zu versehen. Dieser Ansatz hat sich jedoch bereits in der Entwicklung aus verschiedenen Gründen als besonders anstrengend für den Spieler erwiesen, da auf dem Boden liegende virtuelle Gegenstände oftmals dazu führen, dass der Spieler die VR-Controller unbewusst auf den physischen unter sich befindlichen Boden schlägt. Weiterhin verhindert das Greifen eines Gegenstandes durch die Interactable-Komponente, einem Objekt komplexe Eigenschaften zuzuweisen, da diese an die Hand des Spielers gebunden wären. Aus diesen Gründen hat sich die PG MMI2 dazu entschieden, eine weitere Interaktionsmöglichkeit des SteamVR-Frameworks auszunutzen: *Item-Packages* bilden eine starke Möglichkeit, um einem greifbaren Objekt komplexe Funktionen zuzuweisen, die teils sogar den Einsatz beider Hände erfordern können. Bspw. liefert das Framework ein vorgefertigtes Beispiel eines Langbogens, dessen Pfeile durch diesen geschossen und sogar an einem Feuer angezündet werden können [6]. Ermöglicht wird dies durch das tatsächliche Austauschen der Hand des Spielers mit dem entsprechenden gegriffenen Objekt. Zusätzlich dazu bieten *Item-Packages* die Möglichkeit, einen festen Punkt im Raum zu definieren, von dem aus ein Gegenstand aufgenommen und zurückgelegt werden kann. Diese Gegebenheit hat zur Implementierung des `Item-Holsters` geführt, welches wie das Holster einer Pistole an der Hüfte des Spielers verweilt und dem Schläger sowie einem eventuell spielbaren Ball eine feste und immer greifbare Position zuweist. Zusätzlich ermöglichen *Item-Packages* den Punkt zur Aufnahme eines Gegenstandes anders aussehen zu lassen als der gegriffene Gegenstand selbst. So wäre es möglich, bspw. einen Schrank im Raum zu platzieren, welcher bei einer Interaktion jedoch nicht selber gegriffen wird, sondern stattdessen einen Schläger in die Hand des Spielers legt. Aufgrund der damit verbundenen Verwirrungsmöglichkeiten für neue Spieler hat sich die PG MMI2 für eine identische Repräsentation des Aufnahmepunktes eines Gegenstandes sowie für den Gegenstand selbst entschieden.

Das Holster ist in fester Relation zum Kopf des Spielers befestigt, sodass dieses jederzeit vor dem Spieler ist. Genauer liegt das Holster, wie in Abbildung 5.8 zu sehen ist, vor der Hüfte des Spielers. Das Holster behält hierbei eine feste Höhe auf der globalen y-Achse, um zu verhindern, dass es sich an eine unerreichbare Stelle bewegen kann. Eine solche Situation würde beispielsweise auftreten, sobald sich ein Spieler auf den Boden kniet, und so das Holster unter den Boden bewegen. So wird sichergestellt, dass das Holster immer sichtbar bleibt, was dazu folgt, dass die Affordanz, die darin abgelegten virtuellen Gegenstände zu greifen, stets gegeben ist.

5.1.5 Umsetzung des Bots mit ML-Agents

Die Implementierung des Bots mithilfe von ML-Agents findet überwiegend in zwei C#-Klassen statt. Zum einen der `PingPongAgent`, eine Klasse, in der die Beobachtungen und die daraus folgenden Aktionen des Bots definiert sind. Zum anderen der `PingPongEnvironmentController`, welcher den Rahmen für die Trainingsumgebung gibt und insbesondere die Reward-Verteilung bereitstellt, sowie das Training

[6] Steam VR API Dokumentation Class Longbow: `https://valvesoftware.github.io/steamvr_unity_plugin/api/Valve.VR.InteractionSystem.Longbow.html`, letzter Aufruf: 29.09.2023

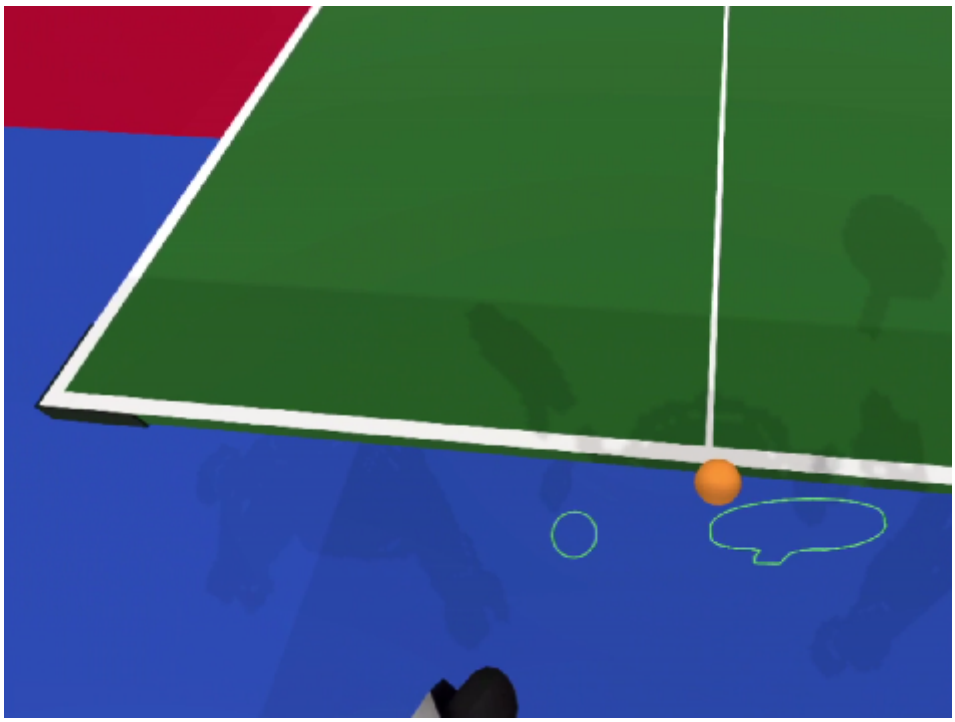


Abbildung 5.8: Itemholster an der Hüfte des Spielers mit in der Hand gehaltenem Ball

steuert und Hilfsmethoden enthält. In diesem Kapitel sollen diese elementaren Klassen und ihre Implementierung näher beleuchtet werden.

Der `PingPongEnvController` ist ein Monobehaviour [7], welches sich als Komponente in der `PlayArea` des Spiels befindet. Er enthält eine Liste der Agenten in der Trainingsumgebung sowie Timer und Zähler, welche den Zeitpunkt diktieren, an dem die Trainingsumgebung zurückgesetzt werden soll. Beim Start des Monobehaviours wird überprüft, ob im `GameModifierManager` das Training aktiviert ist. Ist dies der Fall, so werden die Agenten instanziiert und einige Variablen gesetzt, die vom `PingPongEnvController` benötigt werden. Zuletzt wird das Training von der Methode auf den Ausgangszustand gesetzt.

Bei jedem `FixedUpdate` wird zum einen der `resetTimer` hochgesetzt, welcher das obere zeitliche Ende einer Episode begrenzt, sowie eine Methode für das Beenden einer Episode aufgerufen, sollte diese zu Ende sein. Außerdem bekommen die Agenten Rewards für ihre Position vor der Platte sowie für das korrekte Halten des Schlägers. Diese werden verteilt, indem eine Methode namens `ResolveEvent` aufgerufen wird. Der Methode wird ein Event sowie der Agent, den es betrifft, übergeben. Die Events sind ebenfalls im `PingPongEnvController` definiert und decken Ereignisse ab wie die Position und Schlägerhaltung des Bots, das Erzielen eines Punktes, erfolgreiche Ballwechsel, das Treffen des Balls und das knappe Treffen des Balls. In der `ResolveEvent`-Methode gibt es einen Switch-Case und jedes dieser Events ist ein Fall, in dem Rewards verteilt werden. In einem dieser Fälle wird eine Hilfsmethode verwendet. Fliegt der Ball am Schläger vorbei, so trifft er einen *Collider*. Dabei wird der Abstand vom Ball zum Schläger ermittelt und der Bot bekommt einen höheren Reward, je näher er seinen Schläger zum Ball bewegt hat. Dies geschieht über eine Skript-Komponente des *Colliders*, welche die Methode `ResolveBallDistance` im `PingPongEnvController` aufruft. Ebenfalls im `FixedUpdate` aufgerufen wird die Methode `HandReachChecker`. Der Schläger ist für den Bot frei beweglich und kann auch zu Stellen bewegt werden, welche außerhalb der Armreichweite des Bots sind. Um den Bot zu motivieren, den Schläger in Reichweite zu halten, wird regelmäßig der Abstand vom Schultergelenk zum Schläger gemessen und der Bot bestraft, wenn der Schläger außer Reichweite ist.

Die Methode `ResetScene`, welche vom `FixedUpdate` aufgerufen wird, sollte die Episode zu Ende sein, setzt die Bots auf ihre ursprüngliche Position zurück, indem sie eine zusätzliche Methode namens `ResetAgentPosition` dafür aufruft. Außerdem ruft sie die Aufschlag-Methode auf, wenn einer der Bots einen Aufschlag machen muss. Der Aufschlag wird stets mit zwei Sekunden Verzögerung ausgeführt, damit ein Spieler Zeit hat, sich auf den erforderlichen Rückschlag vorzubereiten. Beim Aufschlag selber wird ein neuer Ball instanziiert, welcher 30 cm oberhalb des Schlägers des Bots spawnt und dann in Richtung eines Punktes über der Platte beschleunigt wird.

Der `PingPongAgent` erbt von der Klasse `Agent`, welche durch das ML-Agents-Paket bereitgestellt wird. Das Skript des `PingPongAgents` befindet sich im modellierten Kopf des Bots und steuert seine Bewegungen, ebenso definiert es die Beobachtungen des Bots. Der `PingPongAgent` hat eine Instanz des `PingPongEnvControllers` und des `GameTrackers`. Dadurch kann der Agent Events an den `PingPongEnvController` geben und durch den GameTracker den Status des Spiels beobachten. Ein Beispiel dafür, wie der Agent Events auslöst, ist, dass er dem `PingPongEnvironment` mitteilt, dass er

[7] Unity Documentation: MonoBehaviour `https://docs.unity3d.com/ScriptReference/MonoBehaviour.html`, letzter Aufruf: 1.10.2023

gegen einen der *Collider* gelaufen ist, welche seine Spielfläche begrenzen sollen.

Die `MoveAgent`-Methode übersetzt eine Aktion, welche ihr in Form einer Ganzzahl übergeben wird, in eine Bewegung. Bei jeder Bewegung handelt es sich um einen kleinen Schritt in der Umgebung. In seiner finalen Version kann sich der Bot auf der x-Achse vor der Platte bewegen und die Hand, welche den Schläger hält, auf und ab sowie lateral bewegen.

In der `CollectObservations`-Methode werden dem Bot Werte gegeben, welche er als Beobachtungen nutzen kann. Diese Beobachtungen umfassen die Position und Geschwindigkeit des Bots, des Schlägers und des Balls. Außerdem kann der Bot den Zustand des Spiels beobachten. Dafür wird der Zustand des Spiels, wie er in 3.1.6 beschrieben ist, als Ganzzahl kodiert. Besonderheit ist hierbei, dass die Kodierung für die beiden Bots unterschiedlich sind, da sie das Spiel jeweils von der anderen Seite betrachten. So können zum Beispiel beide Aufschlagszustände, mit eins kodiert sein, abhängig davon, welcher Bot den Zustand betrachtet.

Zuletzt enthält der `PingPongAgent` eine Methode, welche es einem Menschen ermöglicht, den Bot mithilfe der Tastatur zu steuern. Dies ist besonders für Debugging relevant, da sich so spezifische Zustände gezielt forcieren und somit überprüfen lassen.

Elementar für das Spiel des Bots ist zuletzt ein `MonoBehaviour` namens `BotRacket-CounterHit`. Dabei handelt es sich um ein Skript, welches dafür sorgt, dass der Ball beim Auftreffen auf den Schläger des Bots automatisch zurückgespielt wird. Diese Funktion ist unabhängig vom Training implementiert, um die Fähigkeiten, die der Bot eigenständig erlernen muss, zu minimieren. Zu Beginn eines Spiels wird eine Liste mit zufälligen Integern von null bis drei erstellt. Trifft der Ball auf den Schläger des Bots, wird der aktuelle Wert aus dieser Liste genommen und basierend auf dem enthaltenen Integer einer von drei Punkten gewählt. Zwischen dem Schläger und dem Punkt wird dann ein Richtungsvektor berechnet und der Ball in diese Richtung zurück beschleunigt. Anschließend wird der Index der Liste erhöht. Der Bot muss also selber nicht lernen, den Ball mit Schwung auf die andere Plattenseite zu schlagen. Es genügt für den Bot in der Lage zu sein, den Schläger zum Ball zu bewegen, sodass dieser vom Ball getroffen wird. Das eigentliche Schlagen auf die andere Plattenseite für den Bot von dieser Methode übernommen.

5.1.6 Umsetzung des Bots ohne ML-Agents

Für den Fall, dass kein erfolgreiches Training mit ML-Agents zustande kommt, sollte ein Bot implementiert werden, dem eine fest vorgegebene Strategie zugrunde liegt, die nicht Ergebnis von Machine Learning ist, sondern von der PG selbst implementiert wird. Dieser Bot übernimmt teilweise Funktionsweisen des ML-basierten Bots. Zum Beispiel wird die Bewegung des Bots über Rigidbodies im Kopf und einer der Hände des Bots verwendet, wie es in der ursprünglichen Implementierung des Bots der Fall ist. Ebenso sind die Funktion für den Aufschlag und das Zurückschlagen des Balls beim Auftreffen auf den Schläger in ihrer Funktionsweise identisch zu den Versionen des ML-Agents-basierten Bots. Viele Collider, welche den ML-Agenten bei Beobachtungen unterstützen und für die Rewardverteilung zuständig sind, sind für diesen Bot nicht mehr nötig, da diese Version des KI-Gegenspielers kein Reinforcement Learning nutzt. Dafür kommen für den Bot zwei neue *Collider* zum Einsatz. Einer dieser *Collider* liegt über der Platte des Spielers und der andere erstreckt sich auf Höhe des Bots und bildet eine Wand auf der Bewegungslinie des Bots. Der *Collider* über der Platten-

seite des Spieler startet das Verhalten des Bots und ist ein wichtiger Bestandteil der Fähigkeit des Bots, die Position des Balls zu antizipieren.

Spielt der Spieler einen legalen Aufschlag, so passiert der Ball unweigerlich den *Collider* über der Platte des Spielers. Der *Collider* überprüft zunächst, ob das Objekt, mit dem er kollidiert, tatsächlich ein Ball ist. Ist dies der Fall, so wird anschließend ein bool'scher Wert, welcher den Bot in seiner Ausgangsposition hält, so gesetzt, dass sich der Bot nun bewegen kann. Vom Ball wird ein sogenannter Raycast erzeugt. Dieser Raycast erzeugt einen Strahl entlang eines Vektors, mit dem Kollisionen erkannt werden können [8]. In diesem Fall wird vom Ball aus ein Raycast entlang des Bewegungsvektors des Balls gesendet, wobei der Y-Wert des Raycast-Vektors auf null gesetzt wird. Da der Ball springt, soll die Richtung unabhängig von der Höhe betrachtet werden. Kollidiert dieser Raycast nun mit dem *Collider* auf Höhe des Bots, so kann die Position dieser Kollision verwendet werden, um den Bot entlang der x-Achse zu positionieren. Damit der Bot für den Spieler zu besiegen ist, teleportiert sich der Bot allerdings nicht zu seiner Zielposition, sondern bewegt sich jeden Frame um 1,5 cm in seine Zielrichtung, bis der Schläger an der richtigen Stelle ist. Die Geschwindigkeit von 1,5 cm pro Frame ist dabei so gewählt, dass der Bot es in den meisten Fällen schafft, den Ball zu erreichen, jedoch ist es dem Spieler immernoch möglich durch geschickte Wechsel der Schlagrichtung, den Bot auszuspielen. Die Höhe, auf der der Schläger gehalten wird, wird im Spiel immer der Höhe des Balls angeglichen. Trifft der Ball nun auf den Schläger, so wird einer von drei möglichen Punkten über der Platte gewählt und ein Vektor zwischen Ball und Punkt ermittelt. Der Ball wird dann in diese Richtung beschleunigt, sodass er in den meisten Fällen auf die Platte des Spielers gespielt wird. Das Zurückschlagen wurde nicht weiter perfektioniert, da es beabsichtigt war, dass der Bot nicht perfekt spielt und somit Fehler macht.

Hat der Bot Aufschlag, so wird der Bot in seine Standardposition gebracht. Der Bot wartet zwei Sekunden bis zum Aufschlag und dann wird ein Ball 30 cm über dem Schläger instanziiert und in Richtung eines Punktes über der Platte beschleunigt. Aus Spielersicht erscheint der Ball dabei plötzlich in der Luft. Eine Sekunde danach wird die Bewegung des Bots freigegeben und nachdem der Spieler den Ball zurückgeschlagen hat, fängt der Bot wieder an, die Flugbahn des Balls zu antizipieren.

5.2 Vergleich und Diskussion beider Implementierungsansätze des KI-Gegenspielers

Zu Beginn des Projektes war es vorgesehen, dass der Gegenspieler, welcher einen elementaren Teil des Spielerlebnisses darstellt, durch ML-Agents' Self-Play trainiert werden soll. Motivation war es unter anderem herauszufinden, ob dies eine effiziente Alternative dazu ist, einen Bot zu erstellen, welcher einem manuell erstellten Algorithmus folgt. Außerdem hätten verschiedene Modelle mit unterschiedlichem Trainingsfortschritt verwendet werden können, um Schwierigkeitslevels zu realisieren.

Die Konzeptionierung und die initiale Implementierung des Bots verlief durch gegebene Beispiele und vorherige Erfahrung mit dem Paket ohne Probleme. Ein erstes Training, in dem der Bot nur darauf trainiert werden sollte, sich innerhalb manuell gesetzter Grenzen vor der Platte aufzuhalten, konnte bereits früh in der Entwicklung des

[8] Unity-Dokumentation: Physics.raycast: `https://docs.unity3d.com/ScriptReference/Physics.Raycast.html`, letzter Aufruf 25.09.2023

Bots ausgeführt werden. Dieses Training zeigte auch innerhalb der geplanten Dauer eines Trainings einen Effekt, sodass weitere Aktionen und Rewards hinzugefügt wurden. Als Nächstes sollte der Bot darauf trainiert werden, den Schläger, welchen er beliebig weit von seinem Körper entfernen konnte, in einem Abstand zu halten, welcher eine Armlänge von seiner rechten Schulter nicht überschreitet. Dafür wurde der Bot jedes Mal bestraft, wenn der Schläger außer Reichweite lag. Auch nach dieser Erweiterung des Trainingsprozesses schien das Training noch seinen gewünschten Effekt zu haben. Im nächsten Schritt sollte der Bot darauf trainiert werden, seinen Schläger in Richtung des Balles zu bewegen und diesen gegebenenfalls zu blocken. Dabei konnte der Bot den Schläger in alle Richtungen bewegen. Ein richtiger Schlag sollte erst im nächsten Schritt erfolgen. Bei diesem Schritt waren mehrere Iterationen und einige Anpassungen nötig, bis es den Anschein machte, dass das Training Erfolg hat. Trotzdem blieb das Treffen des Balls unzuverlässig, sodass sich dazu entschieden wurde, die Komplexität zukünftiger Trainings zu reduzieren. Eine Reduktion von Komplexität sollte dadurch herbeigeführt werden, dass der Bot keine vollständigen Schlagbewegungen mehr lernen sollte. Stattdessen sollte eine gerichtete Beschleunigung des Balles automatisch bei Berührung mit dem Schläger erfolgen. So wäre es dem Bot schon über korrekte Bewegungen zum Ball möglich gewesen, einen Ballwechsel zu erwirken und dadurch einen großen Reward zu bekommen. Dennoch kam der Bot nicht über den Trainingserfolg der vorherigen Iteration hinaus. In den nachfolgenden Schritten wurde die Schlägerrotation fixiert, und in der nächsten Iteration wurde dem Bot die Fähigkeit genommen, sich in der Tiefe zu bewegen, um zusätzliche Handlungskomplexität zu entfernen. Die gewollte Konsistenz des Bots blieb dennoch aus. Weil die von Unity gebotenen Optionen damit ausgeschöpft schienen und die Zeit für die Implementierung aufgebraucht war, hat die Projektgruppe beschlossen, auf einen KI-Gegner mit festem Algorithmus zurückzufallen (beschrieben in 5.1.6).

Beim Vergleich der Implementierungen ist auffällig, dass der Bot, welcher nicht auf ML-Agents basiert, wesentlich weniger Code benötigt, während die Komplexität des Codes und die Dauer der Implementierung im Falle beider Implementierungen vergleichbar ist. Wesentlich bei dem ML-Agents-Ansatz ist jedoch die benötigte Trainingsdauer. Die Trainingsdauer wurde zum einen dadurch erhöht, dass der `GameTracker` als Singleton implementiert wurde und somit eine Parallelisierung des Trainings nicht möglich war. Eine Parallelisierung hätte vorgesehen, dass es eine Hauptszene gibt und die Arena samt aller ihrer Unterobjekte und damit verbundenen Skripte dupliziert wird. Seitens der Entwickler gibt es, dem Wissensstand er Projekgruppe zufolge, keine Aussage darüber, wie in so parallelisierten Trainingsprozessen die Reward-Verteilung oder die Verhalten angepasst werden. Da es allerdings nicht mögliche ist mehrere Instanzen, des `GameTracker` zu haben war dies keine Option. Jedoch insbesondere auch durch das iterative Vorgehen ist das Training sehr zeitintensiv, da die Auswirkungen einzelner Anpassungen nur während fortgeschrittenen Trainingsprozessen betrachtet werden können. Ebenfalls zu beleuchten ist das Ergebnis: Trotz mehrerer Wochen Entwicklungsphase, in der viele mehrtägige Trainings durchgeführt wurden, war es nicht möglich, einen Bot mit ML-Agents zu erstellen, der das Spiel eigenständig spielt. Außerdem ist zu erwarten, dass selbst bei einem erfolgreichen Training die Bewegungen des letztendlich ohne ML-Agents erstellten Bot natürlicher auf einen Benutzer wirken als ein potenzielles Ergebnis eines Self-Play-Trainings. Der Grund dafür ist, dass ein Bot, welcher mit ML-Agents trainiert wurde, bei jedem `FixedUpdate` eine zunächst willkürliche Aktion ausführt. Diese Aktionen werden im Verlauf des Trainings weniger willkürlich, jedoch macht der Bot, wenn er

sich beispielsweise nach rechts bewegen will, immer wieder kleine Bewegungen nach links. Durch dieses erratische Verhalten ist der Bot für den Spieler bedeutend weniger vorhersehbar.

5.3 Implementierung der Schnittstelle zwischen Frontend und Backend

Die Schnittstelle (*REST*ful API) zwischen Frontend und Backend wird mithilfe von Flask[31] implementiert. Was *REST*ful Schnittstellen sind, ist in Kapitel 3.2.3 beschrieben.

Zunächst wurde eine Flask[31]-App implementiert, die unter Angabe von Host und Port als Kommandozeilenparameter in `main.py` gestartet werden kann. Innerhalb dieser App sind die unterschiedlichen Routen implementiert – jede Route wird mit dem entsprechenden Dekorateur „`@flask_app.route()`“ gekennzeichnet. Der Übergabeparameter der Route definiert den Namen der Route als String. Wie schon in Kapitel 3.2.3 beschrieben, werden bei *REST*ful APIs die üblichen HTTP-Methoden verwendet, um Daten abzufragen oder zu posten. Welche HTTP-Methode(n) die jeweilige Route ausführen soll, kann ebenfalls als Übergabeparameter der Route definiert werden – dies geschieht als Liste aus Strings der benötigten Methodentypen (in unserem Fall `POST`, `PUT` und `GET`), die mit dem Schlüsselwort *methods* definiert werden.

Je Route können eine oder mehrere Funktionen definiert werden, wenn die entsprechende Route über die zugehörige URL aufgerufen wird.

Die aufgelisteten Routen sind definiert worden und werden genutzt. Der Übersichtlichkeit halber wird hier nicht auf die ungenutzten Routen eingegangen, die in der von Herrn Bengt Lüers bereitgestellten Version der API bereits existierten und nicht weiter verwendet worden sind.

- `/object_data/send`
- `/object_data/process`
- `/object_data`
- `/chatbot`
- `/tutorial`
- `/simple_mistakes`
- `/stop_chatbot`
- `/forehand_backhand`

Die Routen `/object_data/send` und `/object_data/process` sind zu Beginn implementiert worden, um einen grundlegenden Datenaustausch zwischen Frontend und Backend zu ermöglichen und zu testen. Die API empfängt über die Route `/object_data/process` eine Nachricht, die in einem festgelegten Intervall vom Frontend geschickt wird. Um den korrekten Datenaustausch zu testen, wird im Backend eine zufällige Antwort generiert, die dann über die Route `/object_data/send` gesendet wird. In der finalen Version der Implementierung wird diese Route nicht mehr genutzt, sondern galt nur der Einarbeitung in die Implementierung einer Flask[31] *REST* API.

Im Folgenden werden die übrigen Routen hinsichtlich ihrer Funktionen beschrieben. Dabei wird zwischen der Anbindung ans Frontend und ans Backend unterschieden.

5.3.1 Anbindung an das Backend

Die Route `/forehand_backhand` enthält die erste Version der Implementierung zur Klassifikation von Schlägen, die nachfolgend in Abschnitt 5.4.2 beschrieben wird. Da, wie in Abschnitt 5.4.2 erklärt, das Format der Daten anders aussieht, als initial bei dieser Implementierung erwartet, wurde diese Route nie produktiv genutzt.

Die Route `/object_data` ist in der finalen Version der Implementierung für die Klassifikation von Schlägen verwendet worden. Diese Route empfängt die Schlagdaten im Format JSON aus dem Frontend und startet die Klassifikation im Backend, was in Abschnitt 5.4.2 beschrieben wird.

Die Funktionalität, eine Konversation mit dem Chatbot zu starten, ist in der Route `/chatbot` definiert. Wenn auf dieser Route ein PostRequest mit einer Nachricht eingeht, wird die eingegangene Nachricht im Backend verarbeitet und, wie in Kapitel 5.4.1 beschrieben, an den Rasa[25]-Server gesendet und eine Konversation mit dem Chatbot ermöglicht.

Um die Konversation und damit auch das Abspielen der Audiodatei zu stoppen, wird die Route `/stop_chatbot` verwendet. Sie ruft bei eingegangenem PostRequest auf dieser Route die entsprechende PlaySound-Methode, beschrieben in Kapitel 5.4.1, auf.

Mithilfe eines PostRequest an die Route `/tutorial` konnten anfangs der Implementierung die entsprechenden Audiodateien der Tutorials für Vorhand, Rückhand und Aufschlag erstellt und abgespielt werden. Aufgrund der nachfolgend in Kapitel 5.3.2 beschriebenen Voicekonflikte wird diese Route letztlich nicht mehr zum Abspielen, sondern nur noch zum Erstellen der Audiodateien der Tutorials genutzt. Das Abspielen der Tutorials wurde, wie nachfolgend in Kapitel 5.3.2 beschrieben, ins Frontend verlagert.

Die Route `/simple_mistakes` ist ein Relikt aus einer Implementierungsphase, in der der Ball und Schläger oft zu Boden fallen können. So wurde die Überlegung umgesetzt, für diese Fälle einfache Tipps auszugeben, wenn Ball oder Schläger oft zu Boden gefallen sind. Werden diese Fehler im Frontend erkannt, könnte ein PostRequest an diese Route gesendet werden und der entsprechende Tipp ausgegeben werden, wie in Kapitel 5.4.1 beschrieben.

5.3.2 Anbindung an das Frontend

Zunächst sind die Chatbot- und TTS-Funktionalitäten, die genauer in Kapitel 5.4.1 beschrieben sind, wie folgt ans Frontend angebunden:

Um ein Tutorial auszugeben, muss eine Nachricht vom Typ String an die Route `/tutorial` der API gesendet werden. Die Nachricht zeigt an, welches Tutorial gewünscht ist: „forehand“, „backhand“ oder „service“.

Um den Chatbot um Hilfe zu bitten und eine Konversation mit ihm zu starten, wird eine Nachricht vom Typ String wie „hallo“ an die Route `/chatbot` der API gesendet. Bei der Nachricht muss es sich um eine Begrüßungsnachricht handeln, ansonsten wird die Konversation mit dem Chatbot nicht korrekt gestartet.

Hier gab es während der Implementierung aber das Problem, dass die Ausgabe der Audiodateien nicht gestoppt werden konnte. So konnten mehrere Audiodateien gleichzeitig abgespielt werden. Daher wurde diese Anforderung des direkten Stoppens von Audiodateien im Backend dann agil hinzugefügt, wie in Kapitel 5.4.1 beschrieben. Im Frontend wurde die bisherige Implementierung wie folgt angepasst:

Für die Tutorials wurde entschieden, keinen API-Call durchzuführen, um die Audiodateien abzuspielen, sondern sie direkt in Unity zu importieren. Dies ermöglicht ein einfaches Abspielen und Stoppen der Audiodateien direkt in Unity.

Für den Chatbot wurde die Route `/stop_chatbot` der API genutzt. An diese kann eine Nachricht vom String gesendet werden, die „stop“ enthält, und daraufhin wird die momentan abgespielte Audiodatei gestoppt und die gesamte, laufende Chatbotkonversation gestoppt.

So kann sichergestellt werden, dass die Tutorials oder der Chatbot gestoppt werden, wenn beispielsweise der Trainingsraum verlassen wird oder der Nutzer ein anderes Tutorial hören möchte.

Die beschriebenen API-Calls werden mit PostRequests durchgeführt. Dafür muss die URL, die die Route enthält, und die Nachricht angegeben werden. Die PostRequests werden mithilfe der Klasse UnityWebRequest aus *UnityEngine.Networking*[9] durchgeführt. Mit einem Objekt dieser Klasse wird eine HTTP-Kommunikation mit Webservern ermöglicht, also auch das Posten und Abfragen von Daten. Die API-Calls werden dann ausgeführt, wenn ein entsprechender Button betätigt oder der Trainingsraum verlassen wird.

Letztendlich wird in der finalen Version also nur der Chatbot über die API gestartet oder gestoppt. Das Abspielen und Stoppen der Tutorials läuft fast komplett übers Frontend. Das Backend wird hier nur dafür genutzt, die Audiodateien der Tutorials einmal zu erstellen. Dies ist aber nur einmal während der Implementierungsphase geschehen und muss nur dann wiederholt werden, wenn die Tutorials nachträglich angepasst werden.

Vom Frontend werden außerdem Daten bezüglich von Schläger und Ball in Form von JSON gesendet. Diese Daten werden in einem Ringspeicher gespeichert und bei Kollision des Balls mit dem Schläger des Spielers werden 100 Datensätze vor und 50 Datensätze nach der Kollision an den Server gesendet. Dies geschieht im `GameDataSender` und die Daten werden an die Route `/object_data` der API zur weiteren Verarbeitung gesendet.

5.3.3 Unittests der Frontendmethoden

Der für das Framework erstellte Testfall, der Daten vom Frontend sendet, sieht wie folgt aus: Er verwendet einen Testclient, um eine simulierte JSON-Payload an den Endpoint `/object-data` zu senden. Die Payload enthält zufällig generierte Datenfelder, die typische Eingaben imitieren. Der Test überprüft dann, ob der Server mit einem HTTP-Statuscode von 200 antwortet (was auf Erfolg der Anfrageverarbeitung hinweist) und ob die empfangenen JSON-Daten mit den in der Anfrage gesendeten Daten übereinstimmen. Dieser Test verifiziert im Wesentlichen, ob der `/object-data`-Endpoint eingehende Daten korrekt verarbeitet und wie erwartet reagiert.

[9] Unity-Dokumentation UnityEngine.Networking:`https://docs.unity3d.com/Packages/com.unity.multiplayer-hlapi@1.0/api/UnityEngine.Networking.html`, letzter Aufruf 22.09.2023

Act (Ausführung)

Im „Act"-Abschnitt simuliert der Code die tatsächliche Ausführung des Testfalls. Hierbei wird eine HTTP-POST-Anfrage an den Endpoint `/object-data` mithilfe des Flask[31]-Testclients (Client) erstellt. Diese POST-Anfrage enthält das Datenverzeichnis als JSON-Payload. Das Ziel besteht darin, ein Szenario zu emulieren, in dem eine Client-Anwendung Daten über diesen speziellen Endpoint an den Server sendet. Durch diese Maßnahme bewertet der Test, wie der Server eingehende Daten behandelt, wenn eine echte Anfrage gestellt wird.

Assert (Validierung)

Im „Bestätigung"-Abschnitt werden durch den Code Validierungen durchgeführt, um sicherzustellen, dass das Verhalten des Servers den erwarteten Ergebnissen entspricht. Dabei werden zwei Aussagen getroffen:

- `assert response.status-code == HTTPStatus.OK`: Diese Aussage prüft, ob der HTTP-Antwortstatuscode gleich HTTPStatus.OK ist, was einem Statuscode von 200 entspricht. Eine erfolgreiche Antwort sollte den Statuscode 200 zurückgeben, was darauf hinweist, dass der Server die Anfrage ohne Fehler verarbeitet hat.
- `assert response.json == data`: Diese Aussage überprüft, ob die in der Antwort des Servers erhaltenen JSON-Daten mit dem Datenverzeichnis identisch sind, das in der Anfrage gesendet wurde. Dies bestätigt, dass der Server die Daten korrekt empfangen, verarbeitet und als Teil seiner Antwort zurückgegeben hat.

Zusammengefasst führt der „Act"-Abschnitt das Testszenario aus, während der „Bestätigung"-Abschnitt sicherstellt, dass das Verhalten des Servers den Erwartungen entspricht. Dieser strukturierte Ansatz gewährleistet, dass der Endpoint `/object-data` ordnungsgemäß funktioniert, wenn er eingehende Daten verarbeitet.

Empfang und Verarbeitung von Objektdaten

Um Daten vom Frontend zu empfangen und zu verarbeiten, wird eine Flask[31]-Route erstellt, die JSON-Daten über eine POST-Anfrage vom Frontend erhält. Die empfangenen Daten werden anschließend verarbeitet, und eine Antwort wird generiert. Diese Funktionalität wird in Webanwendungen häufig verwendet, um mit Frontend- und Backend-Systemen zu interagieren.

Routendetails

- HTTP-Methode: POST
- Endpoint: `/object-data`

Verwendung

1. Die Frontend-Anwendung sendet eine POST-Anfrage an den Endpoint `/object-data` und übermittelt JSON-Daten als Teil des Anfragekörpers.
2. Der Flask[31]-Backend-Server empfängt die JSON-Daten und extrahiert sie mithilfe von `request.get-json()`.

3. Die extrahierten Daten werden dann verarbeitet. Im vorliegenden Beispiel werden die Daten als Antwort zurückgegeben. Es ist jedoch auch möglich, diese Daten mithilfe einer spezifischen Logik zu verarbeiten und eine Antwort auf Grundlage der empfangenen Daten zu generieren.

Antwort

Die Serverantwort wird auf Grundlage der empfangenen Daten generiert. Im aktuellen Prozess werden die empfangenen Daten als Antwort zurückgegeben.

5.4 Implementierung der Machine Learning Komponenten

Im Folgenden wird die Implementierung des Chatbots und der TTS-Funktionalitäten in Kapitel 5.4.1, die Klassifizierung der Schläge in Kapitel 5.4.2, die Fehlerevaluierung in dem Kapitel 5.4.3 und die Möglichkeit der Parallelisierung der Tests in Kapitel 5.4.5 beschrieben.

5.4.1 Chatbot und TTS-Funktionalitäten

In Kapitel 3.1.5 ist beschrieben, was Chatbots und Dialogsysteme auszeichnen und in welchen Ausführungen sie existieren. Für das in Kapitel 4 beschriebene, umzusetzende Tischtennisspiel soll ein Chatbot entwickelt werden, der spezielle Fragen beantworten kann. Bei diesen Fragen soll es sich um Fragen zur Technik der Tischtennisschläge handeln. Er verfolgt dementsprechend die Aufgabe, die im Spiel aufkommenden Fragen des Nutzers zu klären. Da die zu beantwortenden Fragen klar definiert sind, handelt es sich um einen geschlossenen Wissensbereich und hinsichtlich der Domäne um einen interpersonellen Chatbot. Denn der Chatbot soll nicht darauf spezialisiert sein, ein persönliches Gespräch je nach Nutzer zu führen. Der Chatbot soll insgesamt eine Konversation mit dem Spieler ermöglichen, die der Konversation zweier Menschen besonders ähnlich ist.

Zunächst sind einige Anforderungen an den Chatbot und die Text-To-Speech (TTS) – Funktionalitäten definiert worden, die nach agilem Vorgehen angepasst worden sind und letztendlich wie folgt lauten:

Anforderungen an den Chatbot und die TTS-Funktionalitäten

Folgende Anforderungen sind für den Chatbot definiert worden:

- Der Chatbot soll mithilfe von Rasa[25] implementiert werden
- Der Chatbot soll Deutsch sprechen und verstehen
- Der Chatbot soll via API-Call aktivierbar sein
- Implementierung des Chatbots als Voicebot: Die sprachliche Eingabe des Nutzers muss erfassbar sein und via Spracherkennung in einen Text umgewandelt werden können, damit der Chatbot darauf antworten kann. Die Chatbotantwort muss via TTS-Funktionalität in eine Audiodatei im Format mp3 umgewandelt und abgespielt werden können

- Chatbotinteraktionen müssen per API-Call zu stoppen sein
- Der Rasa[25]-Server für den Chatbot soll mittels einer Funktion automatisch gestartet werden
- Der Chatbot soll die aufkommenden Fragen des Nutzers beantworten und Tutorials zu Vorhand, Rückhand, Aufschlag, der Schlägereinstellung und Hinweise zur Halterung für Ball und Schläger bereitstellen
- Der Chatbot soll den Nutzer informieren können, was für Fehler er gemacht hat, und ihm vorschlagen, welches Tutorial er sich im Trainingsraum erneut anhören könnte
- Wenn der Chatbot nicht antworten kann, muss er dies dem Nutzer mitteilen
- Die URL des Rasa[25]-Chatbotservers muss als *Action Endpoint* definiert werden
- Der Chatbot soll inhaltlich getestet werden, nicht nur via Commandline
- Das NLU-Modell könnte getestet und die verwendeten NLU-Daten und Stories könnten validiert werden[10]
- Das Rasa[25]-Sprachmodell, auf dem der Chatbot basiert, soll statt via Commandline per Funktion trainierbar sein
- Falls der Chatbot nicht antworten kann, soll die Nutzereingabe in einer Textdatei gespeichert werden. Damit können die Trainingsdaten des Chatbots in Zukunft verbessert werden

Einfache Tutorials, für die keine Interaktion zwischen Nutzer und System notwendig ist, können ohne den Chatbot, nur über TTS-Funktionen realisiert werden. Die Anforderungen für die TTS-Funktionalitäten lauten wie folgt:

- Als String definierte Tutorials müssen in Sprache umgewandelt und als Audiodatei im Format mp3 gespeichert und abgespielt werden können
- Tutorials für Aufschlag, Vorhand, Rückhand, Schlägereinstellung sollen implementiert werden
- Die Tutorials sollen via API-Call aktivierbar sein
- Die Tutorial-Sprachausgabe muss per API-Call zu stoppen sein

Implementierung des Chatbots und der TTS-Funktionalitäten

Wie in Kapitel 3.2.5 beschrieben, müssen Trainingsdaten definiert werden, um ein Rasa[25]-Sprachmodell zu trainieren:

Die Datei `config.yml` enthält die NLU-Konfiguration, um das Sprachmodell zu trainieren. Diese Konfiguration wird bei der erstmaligen Erstellung des Rasa[25]-Projekts schon beispielhaft vorgeschlagen und kann nach Belieben angepasst werden. Weiterhin werden Policies beispielhaft vorgeschlagen, die auch zum Trainieren des Sprach-

[10] Testing Your Assistant: `https://rasa.com/docs/rasa/testing-your-assistant/`, letzter Aufruf 01.10.2023

modells genutzt werden.[11] Im Fall dieses Projekts sind die standardmäßig vorgeschlagenen Pipelines und Policies in der Konfiguration beibehalten worden. Dies sind u.a.: `WhitespaceTokenizer`, `RegexFeaturizer`, `LexicalSyntacticFeaturizer`, `CountVectorsFeaturizer`; `MemoizationPolicy`, `RulePolicy` und `UnexpecTEDIntentPolicy`.

Nur die Sprache wurde auf Deutsch geändert, da der zu implementierende Chatbot Deutsch sprechen und verstehen soll. Außerdem kann der Chatbot mit einer Assistant-ID versehen werden, der den Assistenten mit einem eindeutigen Bezeichner identifizieren lässt.

In der Datei `endpoints.yml` wird der Endpunkt definiert, auf dem der Rasa[25]-Server läuft. Im Fall des hier zu implementierenden Chatbots wird als Endpunkt ein Action-Server auf einem bestimmten Port (5055) mit Webhook-Funktion definiert, um die Kommunikation zur Flask[31] *REST* API zu ermöglichen.

In der Datei `credentials.yml` muss der Key `rest` definiert werden, um die Kommunikation mit der Flask[31] *REST* API zu ermöglichen.[12]

Konkret wurden folgende Intents in der Datei `nlu.yml` und in der Datei `domain.yml` samt einiger Beispiele je Intent definiert:

- `greet`
- `goodbye`
- `affirm`
- `deny`
- `bot_challenge`
- `service_tutorial`
- `forehand_tutorial`
- `backhand_tutorial`
- `racket_tutorial`
- `need_help`
- `thanks`
- `look_for_ball_racket`

Es werden sowohl allgemeingültige Gespächs-Intents wie Begrüßung, Verabschiedung, um Hilfe bitten, bedanken, Bestätigen und Verneinen als auch tischtennisspezifische Intents wie Tutorials zu Aufschlag, Vorhand, Rückhand und der Schlägereinstellung sowie Suche nach dem Ball und Schläger definiert.

Die folgenden Äußerungen werden in der finalen Umsetzung in der Datei `domain.yml` definiert:

- `utter_greet`
- `utter_did_that_help`

[11] Rasa-Dokumentation Tuning Your NLU Model: `https://rasa.com/docs/rasa/tuning-your-model/#choosing-a-pipeline`, letzter Aufruf 16.07.2023

[12] Rasa - *REST* API: `https://dev.to/petr7555/rasa-rest-api-4kp6`, letzter Aufruf 16.07.2023

- utter_happy
- utter_goodbye
- utter_happy_to_help
- utter_iamabot
- utter_service_tutorial
- utter_forehand_tutorial
- utter_backhand_tutorial
- utter_racket_tutorial
- utter_offer_help
- utter_where_is_ball_racket
- utter_welcome

Bei den folgenden Äußerungen handelt es sich um solche, die in einer frühen Phase der Implementierung in der domain.yml definiert worden sind. Zu diesem Zeitpunkt existierte die Überlegung, auch Fehler wie ein zu langer oder kurzer Schlag und falsches Timing des Balltreffpunkts zu erkennen und entsprechende Tipps mittels des Chatbots auszugeben. Dies wurde aber im weiteren Verlauf der Implementierung aus Komplexitäts- und Zeitgründen nicht umgesetzt. Trotzdem sind die Äußerungen weiterhin in den Trainingsdaten des Chatbots definiert, da sie in Zukunft noch eingebunden werden können, siehe Kapitel 9.

- utter_mistake_timing_early
- utter_mistake_timing_late
- utter_mistake_stroke_short
- utter_mistake_stroke_long
- utter_mistake_hold_racket
- utter_overload
- utter_unfocused

Die folgenden Regeln werden in der Datei rules.yml definiert:

- Say goodbye anytime the user says goodbye
- Say 'I am a bot' anytime the user challenges
- Explain how the service works anytime the user asks for a service tutorial
- Explain how the forehand works anytime the user asks for a forehand tutorial
- Explain how the backhand works anytime the user asks for a backhand tutorial

- `Explain how the racket adjustment works anytime the user asks for a racket tutorial`
- `Say 'You're welcome' anytime the user says thank you`
- `Ask what the user needs help with`
- `Tell the user where the ball and racket are placed`

Pro Regel wird ein Intent angegeben und eine oder mehrere Aktionen definiert, die der Chatbot dann ausführen soll, wenn die Regel angewandt wird.

Die folgenden Stories werden in der Datei `stories.yml` definiert:

- `problem with service path 1`
- `problem with service path 2`
- `problem with forehand path 1`
- `problem with forehand path 2`
- `problem with backhand path 1`
- `problem with backhand path 2`
- `problem with racket adjustment path 1`
- `problem with racket adjustment path 2`

Die Stories, gekennzeichnet mit „1“ bzw. „2“, unterscheiden sich nur dadurch, ob das ausgegebene Tutorial dem Nutzer geholfen hat oder nicht. Hier gibt es also eine positiv und eine negativ verlaufende Story. Pro Story sind verschiedene Steps aus Intents und Actions festgelegt, die den Gesprächsverlauf definieren.

Sind die Trainingsdaten definiert, kann das Sprachmodell trainiert werden. Das Trainieren eines solchen Modells funktioniert standardmäßig per Commandline. Um das Trainieren unabhängig von der Commandline zu starten, wurde eine Python-Funktion geschrieben, die diesen Commandlinebefehl `rasa train` in einem Subprozess ausführt. Dies ermöglicht, das Trainieren direkt in der erstellten virtuellen Umgebung auszuführen. Für diesen Terminalbefehl musste das aktuelle Betriebssystem bestimmt werden, da das Aktivieren der virtuellen Umgebung je nach Betriebssystem anders ausgeführt werden muss. Hier wurde nur zwischen Windows und Linux unterschieden, da dies die beiden Betriebssysteme sind, die genutzt werden. Eine Erweiterung, beispielsweise auf macOS, ist in Zukunft möglich, siehe Kapitel 9.

Außerdem muss der Rasa[25]-Server gestartet werden, um Antworten des Chatbots erhalten zu können. Auch dies erfolgt per Commandlinebefehl `rasa run`, wobei dazu noch das verwendete Sprachmodell, die Endpoints-Datei, der Port und die Credentials-Datei angegeben werden müssen. Um die Interaktion mit der Flask[31] *Rest* API zu ermöglichen, muss außerdem der Flag `–enable -api` in diesem Befehl gesetzt werden. Auch für das Starten des Rasa[25]-Servers wird eine separate Python-Funktion implementiert, die den Befehl innerhalb eines Subprozesses ausführt. Hier wurde sich aus den gleichen Gründen auf die Betriebssysteme Windows und Linux beschränkt. Ebenfalls ist hier eine Erweiterung auf macOS möglich, siehe Kapitel 9.

Um den Chatbot bzw. das erstellte Sprachmodell nutzen zu können, müssen der Rasa[25]-Server über die eben beschriebene Funktion und die Flask[31]-App gestartet wer-

den. Außerdem muss der Chatbot an die Flask[31]-API angebunden werden. Die Implementierung der API ist genauer in Kapitel 5.3 beschrieben.

Konkret wird die Route `/chatbot` genutzt, um dem Chatbot via PostRequest eine Nachricht zu senden und ihn so zu triggern.

Um die Interaktion des Nutzers mit dem Chatbot zu realisieren, sind mehrere Methoden implementiert worden:

- `get_audio_data`
- `convert_answer_to_speech`
- `save_user_message_to_file`
- `create_mp3_and_play`
- `handle_no_chatbot_answer`

Die Methode `get_audio_data` ist die Methode, die essenziell für die Chatbotkonversation ist. Denn sie enthält Funktionalitäten zum Aufnehmen und Verarbeiten der Nutzernachricht und zum Empfangen und Ausgeben der jeweiligen Chatbotantwort und ruft alle nachfolgenden Methoden, wenn nötig, auf.

`convert_answer_to_speech` sorgt dafür, die Chatbotantwort aus dem JSON - Antwortobjekt zu extrahieren und dann sprachlich auszugeben.

Mit der Methode `save_user_message_to_file` können diejenigen Nutzernachrichten in einer Textdatei gespeichert werden, auf die der Chatbot nicht antworten konnte.

Die Methode `create_mp3_and_play` ermöglicht das Erstellen und Abspielen einer mp3-Datei. Zur Umwandlung von Text in Sprache und fürs Speichern dieser als mp3-Datei wird die Bibliothek gTTS[32] (Google Text-to-Speech) verwendet.

Die Methode `handle_no_chatbot_answer` ruft alle nötigen Funktionen auf, falls der Chatbot nicht antworten konnte, um die ursächliche Nutzernachricht zu speichern und den Nutzer zu informieren, dass seine Äußerung nicht verstanden wurde. Dadurch bekommt der Nutzer eine akustische Rückmeldung, falls seine Eingabe nicht vom Chatbot verstanden werden konnte. Außerdem können die gespeicherten Nutzernachrichten in Zukunft dafür verwendet werden, die Trainingsdaten und damit das Sprachmodell des Chatbots zu verbessern, wie in Kapitel 9 beschrieben.

Die folgende Abbildung 5.9 zeigt den Programmablauf einer Konversation des Nutzers mit dem Chatbot.

Um den Chatbot zu aktivieren, muss vom Frontend aus eine Nachricht als String an die entsprechende Route `/chatbot` der API gesendet werden. Zum Beginn der Konversation muss diese Nachricht eine Begrüßungsnachricht wie „Hallo“ oder „Guten Tag“ sein. Daraufhin startet die Konversation mit dem Chatbot, dessen Ablauf in Abbildung 5.9 zu sehen ist.

Um mit dem Rasa[25]-Chatbot in Echtzeit zu interagieren und Antworten zu erhalten, muss die Nutzernachricht an den Endpunkt `/webhooks/<channel>/webhook` mittels PostRequest gesendet werden. Beim Channel handelt es sich um *rest*, da es sich hier um eine Interaktion mit der *REST* API handelt. Für das Senden solcher Anfragen wird hier die Bibliothek requests[45] verwendet. Zurückgegeben wird ein Response-Objekt,

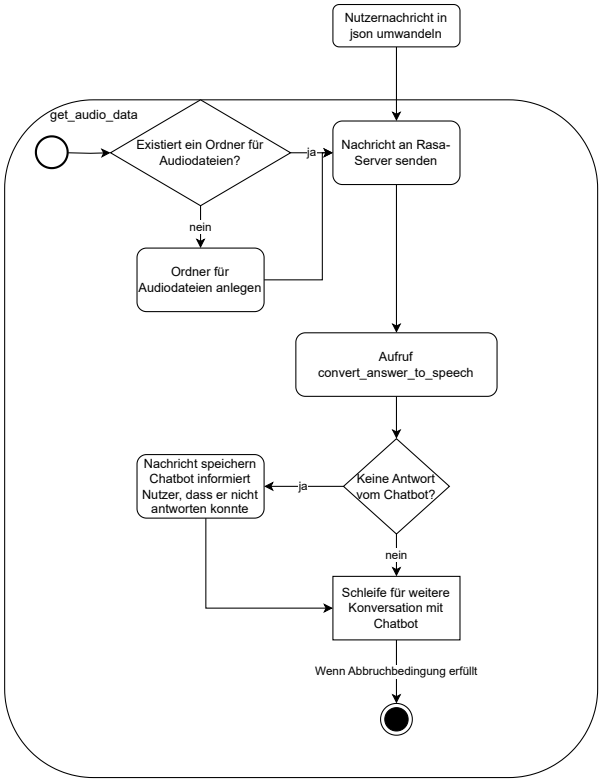


Abbildung 5.9: Ablaufdiagramm der Methode `get_audio_data` für die Konversation zwischen Nutzer und Chatbot in der finalen Version des Chatbots. Eigene Darstellung.

das die Antwort des Chatbots enthält. Im nächsten Schritt wird die Antwort aus diesem JSON-Objekt extrahiert und mittels der TTS-Bibliothek[32] in Sprache umgewandelt und als mp3-Datei gespeichert und abgespielt.

Immer wenn eine Audiodatei abgespielt wird, findet eine Abfrage statt, ob eine Anfrage zum Stoppen des Chatbots an die Route `/stop_chatbot` der API gesendet worden ist. Dies wurde so implementiert, da sonst der Chatbot nicht zu stoppen wäre. Es sei denn, der Nutzer verabschiedet sich oder der gesamte Prozess, und damit der gesamte Backendserver, würde beendet werden. Aber da manche Tutorials über längere Zeit ausgegeben werden, ist eine solche Stopp-Funktionalität eine sinnvolle Option, die Konversation zu verlassen. Um sicherzustellen, dass die nachfolgend beschriebene Schleife auch beim Stoppen verlassen wird, wird die Chatbot-Antwort manuell auf „Auf Wiedersehen“ gesetzt. Wenn aber keine Antwort vom Chatbot geliefert werden kann, wird der Nutzer darüber informiert und die ursächliche Nutzernachricht in einer Textdatei gespeichert.

Nach Verarbeitung der ersten Nutzernachricht und Empfangen der Chatbotantwort beginnt der eigentliche Konversationsverlauf, der innerhalb einer `while`-Schleife implementiert wurde und in Abbildung 5.10 dargestellt ist:

Zu Beginn wird mithilfe der Bibliothek SpeechRecognition[48] die Eingabe des Nutzers via Mikrofon erfasst. Die sprachliche Eingabe kann mithilfe der Google Spracherkennung aus dieser Bibliothek[48] in Text umgewandelt werden, der dann mittels HTTP-POST-Request an den Rasa[25]-Server gesendet wird, wenn der Eingabestring nicht leer ist. Wie bereits erwähnt, wird die Antwort des Chatbots dann in Sprache umgewandelt und als mp3-Datei ausgegeben, und ggf. die ursprüngliche Nutzernachricht gespeichert und der Nutzer darüber informiert, dass der Chatbot seine Äußerung nicht beantworten konnte. Die Schleife wird so lange ausgeführt, bis der Chatbot sich verabschiedet.

Zunächst müssen auch die TTS-Funktionalitäten an den Flask[31]-Server angebunden werden. Über die Route `/tutorial` können die verschiedenen Tutorials gestartet werden: Dafür muss eine Stringnachricht per HTTP-POST-Request an diese Route gesendet werden, die einem der folgenden Strings entspricht:

- „forehand“
- „backhand“
- „service“
- „racket“

Je nach übergebenen String wird das entsprechende Tutorial gestartet. So wird für „forehand“ das Vorhandtutorial, für „backhand“ das Rückhandtutorial und für „service“ das Aufschlagtutorial gestartet. Mit „racket“ werden Hinweise dazu gegeben, wie der Schläger in der Hand rotiert werden kann.

Für die TTS-Funktionalität zur sprachlichen Ausgabe von Vorschlägen, wenn der Fehlerschwellwert im Spiel überschritten wurde, ist eine zum Tutorial ähnliche Funktion implementiert worden. Übergeben wird hier einer der folgenden Strings, der dann den erkannten Fehler enthält, für den der Schwellwert überschritten wurde:

- „forehand“
- „backhand“

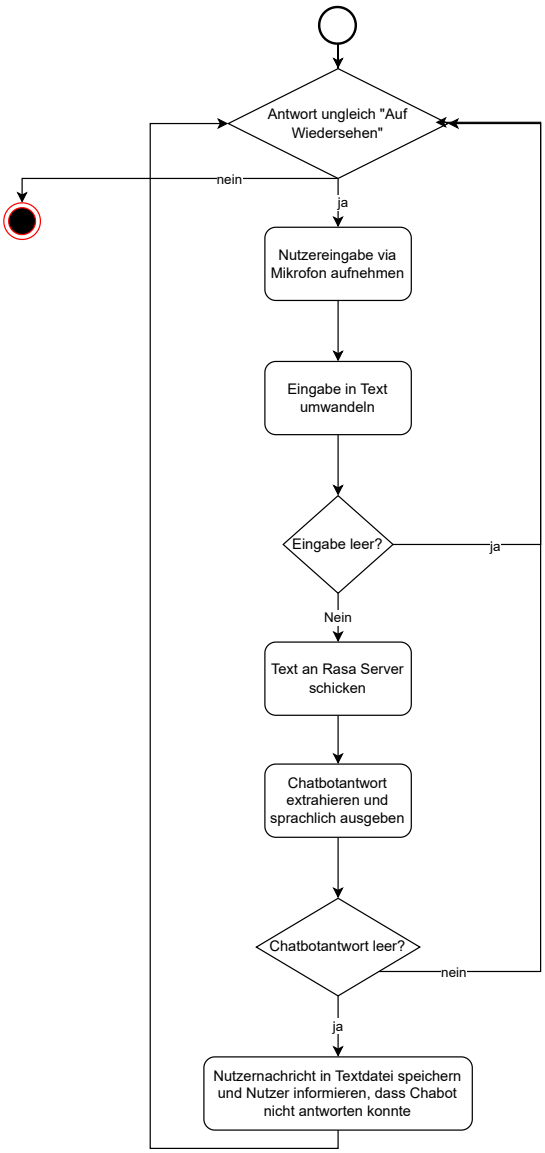


Abbildung 5.10: Ablaufdiagramm des Konversationsverlaufs zwischen Nutzer und Chatbot in der Methode `get_audio_data` in der finalen Version des Chatbots. Eigene Darstellung.

- „service“

Die sprachliche Ausgabe gibt für den entsprechenden Schlag (bei „forehand“ für die Vorhand, bei „backhand“ für die Rückhand und bei „service“ für den Aufschlag) aus, dass einige Fehler in dieser Schlagart erkannt wurden. Dem Spieler wird der Hinweis gegeben, dass er zum Üben des entsprechenden Schlages den Trainingsraum betreten kann. Der Prozess der Fehlerevaluierung wird näher in Abschnitt 5.4.3 beschrieben.

In beiden Fällen sind die Tutorials bzw. Vorschläge als Strings vordefiniert und werden bei dem entsprechenden Funktionsaufruf in eine Audiodatei (mp3) konvertiert und anschließend ausgegeben.

Die Methode `convert_to_audio_play` ist ähnlich zur bereits erwähnten Methode `convert_answer_to_speech`. Hier wird ebenfalls zunächst ein eigener Ordner für die mp3-Dateien der Tutorials und Vorschläge erstellt, wenn dieser noch nicht existiert. Existiert der Ordner bereits und befindet sich schon eine mp3-Datei fürs Tutorial darin, wird diese gelöscht. Dadurch soll verhindert werden, dass sich zu viele Audiodateien in diesem Ordner befinden und dies zu unübersichtlich werden würde. Im Gegensatz zur erwähnten Methode `convert_answer_to_speech` wird der hier beschriebenen Methode der String übergeben, der direkt ausgegeben werden kann. In der entsprechenden Methode für den Chatbot muss der String erst aus dem Antwortobjekt extrahiert werden. Außerdem muss dort der Fall berücksichtigt werden, dass die Chatbotantwort aus mehreren Nachrichten bzw. Äußerungen bestehen kann. Ansonsten sind die gleichen Methoden der implementierten PlaySound-Klasse aus Abbildung 5.11 verwendet worden, um die mp3-Datei des Tutorials abzuspielen. Auch hier findet eine Abfrage statt, ob ein API-Call an die Route `/stop_chatbot` vorliegt, um die Audioausgabe zu stoppen.

Außerdem wurde in einer zwischenzeitlichen Version des chatbots realisiert, kurze Tipps mittels TTS-Funktionen auszugeben, wenn der Spieler leichte Fehler gemacht hat. Diese Tipps können über API-Calls an die Route `/simple_mistakes` ausgegeben werden. Bei den leichten Fehlern handelt es sich um das Herunterfallen von Ball und Schläger aus der Hand auf den Boden. Um dies umzusetzen, wurde die Tipps als Strings definiert und eine Methode analog zur Methode `convert_to_audio_play` aus dem Tutorial implementiert.

Im weiteren Verlauf der Implementierung fand die Anpassung statt, dass der Schläger nicht mehr zu Boden fallen und der Ball aus dem Holster genommen werden kann, wenn dieser heruntergefallen ist. Außerdem werden die Index-Controller nicht mehr verwendet, bei denen der Nutzer seine Hände stets geschlossen halten muss, um Schläger und Ball in der Hand zu behalten. In der finalen Version ist es also nicht mehr notwendig, Tipps zu geben, wie der Nutzer Ball und Schläger aufhebt und festhält, wenn Ball oder Schläger oft zu Boden gefallen sind. Dementsprechend sind die TTS-Funktionalitäten fürs Geben dieser einfachen Tipps zwar in der finalen Version noch vorhanden, werden aber nicht mehr genutzt.

Wahl der Bibliothek für Audiofunktionalitäten

Die Wahl der Bibliothek zum Abspielen von mp3-Dateien ist zunächst auf playsound[13] gefallen. Dies ist ein Package, dessen einzige Funktionalität ist, Audiodateien abzuspielen. Dies schien sehr simpel und für unsere Implementierung ausreichend zu sein.

[13] Playsound-Dokumentation: `https://pypi.org/project/playsound/`, letzter Aufruf 22.09.2023

Bei den ersten Tests traten aber einige Fehler beim Abspielen der Audiodaten mit playsound[13] auf. Diese Fehler traten unregelmäßig auf und waren schlecht zu reproduzieren. Es ist bis heute unverständlich, warum diese Fehler aufgetreten sind. Nach entsprechender Recherche des Fehlers und Möglichkeiten zur Lösung war der einzige, hilfreiche Ansatz playsound[13] auf die Version **1.2.2** downzugraden.

Im weiteren Verlauf der Implementierung sind verschiedene Packages zur Datengenerierung ins Repository eingebunden worden, siehe Kapitel **??**. Dabei sind Fehler bei der Installation der Packages aufgetreten. Eine Ursache der Probleme ist, dass playsound[13] in der Version 1.2.2 eine veraltete Numpy[34]-Version nutzt, die die Installationsfehler verursacht haben. Daraufhin wurde die playsound[13]-Bibliothek durch die pygame[40]-Bibliothek ersetzt.

Letztendlich hat sich pygame[40] als sinnvollere Alternative herausgestellt, da es viele Audiofunktionalitäten wie das Abspielen und auch das Stoppen von Audiodateien bereits unterstützt. Dies ist eine Funktionalität, die seitens des Frontends benötigt wird. In playsound[13] wäre die einzige, unschöne Option zum Stoppen gewesen, das Abspielen der Audiodatei in einem eigenen Thread durchzuführen und diesen direkt zu beenden, wenn das Abspielen gestoppt werden soll. Außerdem ist pygame[40] eine große Bibliothek für die Spieleentwicklung, die viel genutzt und immer noch weiterentwickelt wird.

Die Audiofunktionalitäten wurden also in der finalen Version unter Verwendung der Bibliothek pygame[40] implementiert. Abbildung 5.11 zeigt das UML-Klassendiagramm der erstellten Klasse `PlaySound`. Es sind Methoden zum Abspielen und Stoppen einer Audiodatei und zum Beenden von pygame[40] definiert worden. Für das Stoppen sind außerdem die Klassenvariable `stop_chatbot` und Methoden zum Setzen und Abfragen dieser Variable definiert worden.

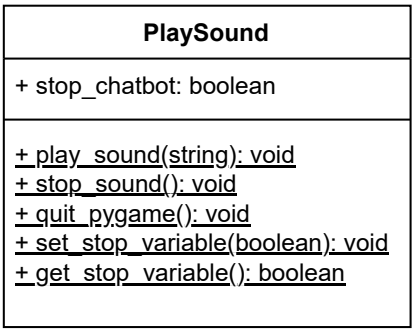


Abbildung 5.11: UML-Diagramm des Klasse `Playsound` zur Implementierung der Audiofunktionalitäten unter Verwendung der Bibliothek pygame[40]. Eigene Darstellung.

Testen des Chatbots und der Hilfsmethoden

Während der Implementierung des Chatbots und der TTS-Funktionalitäten sollten, wenn möglich, alle Funktionen und Hilfsmethoden getestet werden. Dies wurde auch soweit umgesetzt, wobei aber manche Methoden nur implizit über andere Tests getestet worden sind, statt über dedizierte Unit-Tests.

Dabei sind serverseitige Tests, die die Flask[31]-Routen testen, in der `test_server.py`

geschrieben worden. Hier werden die Chatbot- bzw. TTS-Routen aber nur hinsichtlich ihrer Existenz getestet, während in der `test_rasa_functionalities.py` diese Routen bezüglich des Datenaustauschs bzw. ihrer Funktionalität getestet werden. Dies musste so umgesetzt werden, da die Servertests auf dem Flask[31]-Testclient aufsetzen, welcher nur Daten vom Typ JSON senden kann. In der tatsächlichen Implementierung der Chatbot-Routen werden aber Strings gesendet. Daher werden diese Routen in der bereits erwähnten Testdatei konkret getestet, basierend auf der Bibliothek urllib aus der Standardbibliothek von Python[14].

Die Methoden, die sich auf den Rasa[25]-Server beziehen, und der tatsächliche Datenaustausch über die Routen der API werden, wie bereits erwähnt, in `test_rasa_functionalities.py`, die Hilfsmethoden fürs Tutorial und die Vorschläge in `test_tutorial.py`, die Hilfsmethoden für die einfachen Fehler in `test_simple_mistakes` und die Audiofunktionalitäten in `test_audio_functionalities.py` getestet.

Beim Testen der mit pygame[40] implementierten Audiofunktionalitäten trat in der Test-Pipeline der Fehler auf, dass das Audiogerät nicht gefunden werden konnte. Ursache für diesen Fehler war, dass der Runner in der Continous Integration von GitLab des DFKI nicht über eine Soundkarte verfügt. Daher kann kein Ausgabegerät gefunden werden, um die Audiodatei in den Tests abzuspielen oder zu stoppen. Die Lösung dafür war, unter dem Linux-Betriebssystem der Test-Jobs einen virtuellen Audioadapter mit pulseaudio[15] zu erstellen.

Des Weiteren lässt sich der Chatbot-Assistent mittels Funktionalitäten von Rasa[25] zu testen: Einerseits lässt sich das trainierte Sprachmodell anhand von Teststories mittels `rasa test` testen. In den Teststories können Nutzereingaben im Rahmen einer bestimmten Story angegeben werden. So kann überprüft werden, ob der Chatbot sich zur gegebenen Nutzereingabe entsprechend der Story verhält und antwortet. Automatisiert werden nach Aufruf des genannten Befehls ein Bericht, eine Konfusionsmatrix und ein Konfidenzhistogramm fürs Intent-Klassifizierungsmodell erstellt. Im Report werden beispielsweise Metriken wie *Precision*, *Recall* und der *F1-Score* für jeden Intent aufgelistet.[16]

Dafür wird in der Testdatei `test_chatbot.py` ein Test definiert, der den Commandlinebefehl `rasa test` innerhalb eines Subprozesses ausführt. Auch das erfolgreiche Trainieren des Sprachmodells wird in dieser Datei getestet. Wie auch bei den anderen Rasa[25]-Funktionen zum Starten des Rasa[25]-Servers oder zum Trainieren des Sprachmodells werden hier nur die Betriebssysteme Windows und Linux unterstützt. Die Erweiterungsmöglichkeit auf macOS ist in Kapitel 9 beschrieben. Andererseits lassen sich die Trainingsdaten und Stories mittels `rasa data validate` validieren, um mögliche Inkonsistenzen in den NLU- und Trainingsdaten und der Domäne zu finden. Zuletzt bietet Rasa[25] mit `rasa data split nlu` die Möglichkeit, um das Sprachmodell entweder mittels Crossvalidation oder mit Held-Out Testset, einem Datensatz, welcher während des Trainings zurückgehalten wurde, zu testen. Bei der Crossvalidation werden einige train/test-Spilts generiert, während des Held-Out Testset durch

[14] Urllib-Dokumentation: `https://docs.python.org/3/library/urllib.html`, letzter Aufruf 22.09.2023

[15] PulseAudio-Dokumentation: `https://www.freedesktop.org/wiki/Software/PulseAudio/`, letzter Aufruf 22.09.2023

[16] Rasa-Dokumentation Testing Your Assistant: `https://rasa.com/docs/rasa/testing-your-assistant/`, letzter Aufruf 06.07.2023

Mischen und Trennen der vorhandenen NLU Daten erzeugt wird.[16]

Verwendung der mittels TTS implementierten Tutorials

Im Laufe der Implementierung sowohl im Front- als auch im Backend erschien es einfacher, die Tutorials nicht im Backend anzufordern und dort abzuspielen, sondern die Audiodateien der Tutorials direkt in Unity zu hinterlegen. Ein Vorteil davon ist, dass die Audioausgabe in Unity leichter gestoppt werden kann.

Trotzdem sind die TTS-Funktionalitäten weiter an den Flask[31]-Server angebunden. Denn so können die Audiodateien der Tutorials über einen entsprechenden API-Call erstellt und manuell ans Frontend gegeben werden.

5.4.2 Klassifizierung der Schläge

Um zwischen den Schlägen unterscheiden zu können, wurden Machine-Learning-Komponenten verwendet. Konkret wurden zunächst verschiedene neuronale Netze aufgebaut und im Nachgang im Training anhand der Accuracy und dem *Loss* evaluiert und abschließend in der Klassifizierung verschiedener Schläge gegenübergestellt, um das beste Netz auszuwählen. In diesem Abschnitt werden zunächst die ersten Versuche erklärt, danach werden die Daten beschrieben. Anschließend folgt eine Beschreibung der Netze. Abgeschlossen wird der Abschnitt mit der Beschreibung und Erläuterung des Trainings sowie der Auswahl des Netzes.

Erste Versuche

Da zu Beginn noch nicht klar war, wie die Daten für die Schläge aussehen und wie diese aufgenommen werden, wurde zunächst ein Datengenerator geschrieben, der verschiedene x, y und z-Koordinaten sowie -Rotationen zufällig in Schwellwerten pro Schlagart generiert. Es wurde dabei ein einzelner Punkt betrachtet und in Vorhand und Rückhand unterteilt. Darauf aufbauend wurde ein neuronales Netz trainiert und eine Route implementiert. Es stellte sich heraus, dass die Daten aus dem Frontend anders aufgebaut waren. Statt der Aufnahme eines Datenpunktes wurde ein Schlag als eine Sequenz von Datenpunkten aufgenommen. Aus diesem Grund wird der erste Versuch nicht weiter betrachtet und wurde als ersten Einstieg in *PyTorch*[17] und neuronale Netze genutzt. Im Folgenden wird näher auf die tatsächlichen Daten und auf die implementierten sowie das verwendete neuronale Netz eingegangen.

Datenaufnahme und -vorbereitung

Im Folgenden soll es um die tatsächlichen Daten gehen, die im Spiel generiert werden und mit denen die neuronalen Netze trainiert wurden. Neben der Position und Geschwindigkeit des Schlägers in x, y und z sind ebenfalls die Winkelgeschwindigkeit in x, y und z sowie die Rotation in x, y, z und w und der Zeitpunkt aufgenommen. Für die Aufnahme von Schlagdaten wurde vom Frontend ein Branch bereitgestellt, in dem eine frühe Version des Bots dem Spieler auf Knopfdruck einen immer gleichen Aufschlag zuspielen konnte. Der Branch ist isoliert vom Development-Branch und wurde nur zur Datenaufnahme genutzt. Es ist eine Funktion bereitgestellt, die bei jedem Ballkontakt mit dem Schläger die 100 Frames vor und die 50 Frames nach Kontakt aufnimmt und als JSON-Datei abspeichert. Diese 150 Frames werden als ein

[17] PyTorch: `https://pytorch.org/`, letzter Aufruf 21.09.2023

Schlag zusammengefasst. Jedes Frame enthält die entsprechenden Daten für Position, Geschwindigkeit, Winkelgeschwindigkeit und Rotation für den jeweiligen Zeitpunkt.

Für das Training wurden 1687 Aufschläge, 814 Rückhandschläge und 858 Vorhandschläge mit dem zuvor beschriebenen System aufgezeichnet, die also im Folgenden als Ground-Truth-Daten betrachtet werden können. Es wurden mehr Aufschläge aufgenommen, da diese ohne einen tatsächlichen Ballwechsel möglich sind. Die Möglichkeit sich einen Ball zuspielen zu lassen, stand erst zu einem späteren Zeitpunkt zur Verfügung, um Vorhand- und Rückhandschläge aufzunehmen. Um u.a. die Datenschieflage auszugleichen, wurde ebenfalls ein Datengenerator eingesetzt, was im nächsten Abschnitt näher erläutert wird.

In den Daten ist ein Label zu finden, mit welcher Seite des Schlägers der Ball getroffen wurde. Dieses Kennzeichen war ausschließlich dafür eingebaut, die Daten vor dem Training der Netze zu filtern und in Vorhand- und Rückhandschlag einzuteilen. In der Echtzeitklassifizierung wurde es nicht verwendet, da der Spieler den Schläger rotieren und damit auch andersherum in der Hand halten kann. Aufschläge wurden separat aufgenommen und nicht gefiltert. Für die Klassifizierung wurden aus den Daten die jeweils 13 Werte bestehend aus x, y und z-Koordinaten, Winkelgeschwindigkeit und Geschwindigkeit in x, y und z, die Rotation in x, y, z und w jeder Sequenz genutzt. Rotation w ist dabei ein Skalar, der die Drehung um die Rotation x, y und z repräsentiert. [18] Die Rotation, Position, Winkelgeschwindigkeit und Geschwindigkeit werden im Folgenden als Eigenschaften bezeichnet. Der Zeitpunkt wurde für das Training nicht verwendet, da der Abstand der verschiedenen Frames untereinander gleich war und daher implizit in der Sequenz der Datenpunkte eines Schlages enthalten ist. Weitere Datenfilterungen wurden nicht durchgeführt. Nach dem ersten Test der Netze mussten die Daten noch standardisiert werden, was in Abschnitt „Test der Netze und deren Anpassung“ näher erläutert wird.

Datengenerierung

Zusätzlich zu den aufgenommenen Daten wurde ein Datengenerator genutzt, um die Vielfalt der Daten für Vorhand, Rückhand und Aufschlag zu erweitern. Ebenfalls konnte mithilfe der Datengenerierung die Anzahl an Vorhand-, Rückhand- und Aufschlagdaten für das Training gleich gehalten werden, um eine ausbalancierte Datenlage zu sichern. Für die Datengeneratoren wurden zwei Pakete genutzt. Ein Datengenerator basiert auf dem Python-Paket *synloc*[19] und ein Datengenerator basiert auf dem Python-Paket *ctgan*[6]. Die Funktion der Datengeneratoren wurden bereits in Abschnitt 3.1.4 beschrieben.

Als erste Daten generiert wurden, stellte sich heraus, dass die Generierung mit *synloc* bessere Ergebnisse erzielte als mit *ctgan*. Im Listing 5.1 ist ein Auszug von den ersten drei Sequenzen eines generierten Aufschlages dargestellt. Dagegen ist im Listing 5.2 ein Auszug der generierten Daten von *synloc* dargestellt. Offensichtlich passen die Daten der Sequenzen bei *synloc* besser zueinander, weil bspw. die Positionsdaten der einzelnen Sequenzen aussehen, als würden sie aufeinander folgen. Das ist bei der Datengenerierung mit *ctgan* nicht der Fall. Aus diesem Grund wurde sich für den Datengenerator *synloc* entschieden. In Abschnitt 5.4.2 und in der Studienauswertung

[18] Unity-Dokumentation Quaternion.w: `https://docs.unity3d.com/ScriptReference/Quaternion-w.html`, letzter Aufruf 06.09.2023

[19] Synloc-Repository: `https://github.com/alfurka/synloc`, letzter Aufruf 26.06.2023

7 wird genauer analysiert, inwiefern die vom Datengenerator generierten Daten das Training beeinflussen.

Listing 5.1: Auszug der generierten Daten mit ctgan


```
    {
    "_time":79.2803281992,
    "_positionX":-0.6111925489,
    "_positionY":1.0171927671,
    "_positionZ":3.979495722,
    "_rotationX":-0.7558568366,
    "_rotationY":0.7144066358,
    "_rotationZ":-0.6558080664,
    "_velocityX":0.0016284721,
    "_velocityY":-0.0030865435,
    "_velocityZ":0.0005432655,
    "_angularVelocityX":-0.0021116439,
    "_angularVelocityY":0.002521118,
    "_angularVelocityZ":0.0015794295,
    "_rotationW":0.0200656687
  },
  {
    "_time":60.473956769,
    "_positionX":-0.3660180548,
    "_positionY":1.1791267578,
    "_positionZ":3.5594732739,
    "_rotationX":0.1737839073,
    "_rotationY":0.7751320084,
    "_rotationZ":0.6031103517,
    "_velocityX":0.0014433705,
    "_velocityY":-0.0022507175,
    "_velocityZ":0.0011745565,
    "_angularVelocityX":-0.0014635322,
    "_angularVelocityY":-0.0020879833,
    "_angularVelocityZ":-0.0023989463,
    "_rotationW":0.2968301337
  },
  {
    "_time":83.8341147216,
    "_positionX":-0.6052261782,
    "_positionY":0.9416717007,
    "_positionZ":3.0104751242,
    "_rotationX":0.3488704727,
    "_rotationY":0.3257843652,
    "_rotationZ":0.6210612504,
    "_velocityX":-0.0003825247,
    "_velocityY":0.0029188489,
    "_velocityZ":0.0027842929,
    "_angularVelocityX":0.0008132599,
    "_angularVelocityY":-0.0001245546,
    "_angularVelocityZ":0.0014936262,
```

```
    "_rotationW":0.1860711963
  },
```

Listing 5.2: Auszug der generierten Daten mit synloc

```
  {
    "_time":24.1819935705,
    "_positionX":0.0186406364,
    "_positionY":1.227533825,
    "_positionZ":2.8855540065,
    "_rotationX":0.540148692,
    "_rotationY":0.6226658113,
    "_rotationZ":0.433002936,
    "_velocityX":0.0,
    "_velocityY":0.0,
    "_velocityZ":0.0,
    "_angularVelocityX":0.0,
    "_angularVelocityY":0.0,
    "_angularVelocityZ":0.0,
    "_rotationW":0.2670783964
  },
  {
    "_time":24.1911005335,
    "_positionX":0.0200027841,
    "_positionY":1.2264816794,
    "_positionZ":2.8898984597,
    "_rotationX":0.5390083737,
    "_rotationY":0.6238264509,
    "_rotationZ":0.4351187716,
    "_velocityX":0.0,
    "_velocityY":0.0,
    "_velocityZ":0.0,
    "_angularVelocityX":0.0,
    "_angularVelocityY":0.0,
    "_angularVelocityZ":0.0,
    "_rotationW":0.2680902333
  },
  {
    "_time":24.2018841305,
    "_positionX":0.0213103229,
    "_positionY":1.2258396328,
    "_positionZ":2.8928420595,
    "_rotationX":0.5384768704,
    "_rotationY":0.6235111818,
    "_rotationZ":0.4372406775,
    "_velocityX":0.0,
    "_velocityY":0.0,
    "_velocityZ":0.0,
    "_angularVelocityX":0.0,
    "_angularVelocityY":0.0,
```

```
46      "_angularVelocityZ":0.0,
47      "_rotationW":0.2690909741
48    },
```

Für die Daten und die Datengenerierung wurde ein neues Repository erstellt, um die Datengenerierung und die große Anzahl an Daten gekapselt vom Backend zu halten. Außerdem wurden damit Probleme zwischen kollidierenden Packages, was in Abschnitt 5.4.1 beschrieben wurde, umgangen. In diesem Repository befinden sich die Funktionen zum Einlesen der Daten und zur Generierung neuer Daten mithilfe von *synloc*. Ebenfalls befindet sich dort eine Funktion, mit der die Daten in *Tensoren* gespeichert werden. Diese *Tensoren* wurden in das Backend-Repository kopiert, um diese *Tensoren* als Trainingsdaten einlesen zu können. Ein *Tensor* ist ein *PyTorch*-Datenyp und stellt eine multidimensionale Matrix von Werten eines Datentyps dar [20]. Durch die Umbenennungen der einzelnen Schlüssel-Wert-Paaren (key-value-pairs) in den JSONs mussten zwei verschiedene Einlesemethoden geschrieben werden. Für die Echtzeitklassifizierung wurde die Funktion zum Einlesen einer JSON in das Backend überführt.

Implementierung der neuronalen Netze

Wie bereits im vorangegangen Kapitel beschrieben, wurde im ersten Versuch ein *Feed-Forward Netzwerk* verwendet, um zwischen Vorhand- und Rückhandschlägen zu unterscheiden. Da es sich bei den tatsächlichen Daten um eine feste Länge an Sequenzen handelt, wurde sich entschieden, kein reines *Feed-Forward Netzwerk* zu verwenden, da diese Sequenzdaten nur schwer erlernen können [Sutskever et al., 2014]. Stattdessen wurden verschiedene andere Arten von künstlichen neuralen Netzen trainiert, um empirisch das auf den relevanten Daten am besten performende Netz auszuwählen.

Bei der Auswahl der Netzwerkarchitektur wurde sich zunächst für ein *CNN* entschieden. Dabei wurde sich an [Shintani et al., 2021] orientiert, da diese Arbeit ebenfalls Sequenzdaten verarbeitet hat und dafür ein *CNN* verwendet hat: Beim Schreiben mit einem digitalen Stift wurden Daten aufgenommen, die in die unterschiedlichen Buchstaben klassifiziert wurden. Die Wahl des *CNNs* wurde damit begründet, dass ein *CNN* die Abhängigkeiten der Daten zueinander erkennen kann. [Shintani et al., 2021]

Neben dem *CNN* wurde ein *RNN* verwendet, da es vor allem für die Verarbeitung von sequentiellen Daten geeignet ist [Bengio et al., 2017]. Neben dem *RNN* wurde ein *LSTM* verwendet, welches ebenfalls für sequentielle Daten geeignet ist und dabei ältere Datenpunkte mehr berücksichtigen kann als das *RNN* [Bengio et al., 2017]. Die Funktionsweise von *CNNs*, *RNNs* und *LSTMs* wurde bereits in Abschnitt 3.1.3 erläutert.

Im Folgenden wird der Aufbau der implementierten Netze beschrieben. In allen implementierten neuronalen Netzwerken wurde eine Ausgabeschicht bestehend aus drei Neuronen für die möglichen drei Klassen verwendet. Für die Ausgabeschicht wurde jeweils die Aktivierungsfunktion *Softmax* verwendet, da mithilfe der *Softmax*-Funktion Wahrscheinlichkeitsverteilungen für die verschiedene Klassen erzeugt werden [Sharma et al., 2017]. In allen anderen Schichten wurde in den *CNNs* und dem

[20] PyTorch-Dokumentation torch.tensor: `https://pytorch.org/docs/stable/tensors.html`, letzter Aufruf: 24.09.2023

LSTM die *ReLU-Aktivierungsfunktion* verwendet, da sie eine der weit verbreiteten Aktivierungsfunktionen ist und zu schnellem Training beitragen kann
[Qiumei et al., 2019]. Die *Leaky-ReLU* wurde nicht mehr getestet, da das Training mit *ReLU* bereits erfolgreich war (s. Abschnitt „Training der Netze„). Auf die im *RNN* verwendeten Aktivierungsfunktionen wird bei der Beschreibung der Architektur näher eingegangen.

Zunächst werden die drei verwendeten *CNNs* beschrieben. Beim ersten *CNN* sind die Daten zweidimensional strukturiert. Die Dimensionen bestehen aus jedem Zeitpunkt und den verschiedenen Daten eines Schlages pro Zeitpunkt. Dieses ist auch in Abbildung 5.12 dargestellt. Es sind zur Vereinfachung, wie auch in den Abbildungen 5.13 und 5.14, nur die ersten sechs Zeitpunkte abgebildet. Der *Kernel*, der im ersten *CNN* verwendet wird, hat die Größe $2 \cdot 13$. Dadurch kann dieser zwei Zeitpunkte während eines Schlages gleichzeitig betrachten. Wie in der Abbildung 5.12 dargestellt ist, untersucht der *Kernel* zunächst die Zeitschritte 1 und 2 (rot umrandet). Folgend betrachtet der *Kernel* die nächsten beiden Zeitschritte (blau umrandet), bis er das Ende der Sequenz erreicht hat. Danach beträgt das Format der Daten $1 * 149$. Die Daten werden an ein *Feed-Forward Netzwerk* weitergegeben. Die erste Schicht des *Feed-Forward-Teils* besteht aus 149 Neuronen, die mit einer zweiten Schicht, bestehend aus 298 Neuronen verknüpft sind. Es folgt die Ausgabeschicht.

Im zweiten *CNN* sind die Daten ebenfalls zweidimensional strukturiert, ähnlich wie im ersten *CNN*. Zusätzlich zu den verschiedenen Eigenschaften werden die Daten mithilfe von *Zero-Padding* erweitert. Im zweiten *CNN* werden zwei aufeinanderfolgende *CNN*-Schichten verwendet. In der ersten Schicht kommt zunächst ein *Kernel* der Größe $2 \cdot 3$ zum Einsatz. Die Abbildung 5.13 veranschaulicht, dass dieser *Kernel* jeweils die drei x-, y- und z-Komponenten jeder Eigenschaft sowie die Rotation w und die beiden Werte des *Zero-Paddings* für zwei aufeinanderfolgende Zeitpunkte vergleicht. Die Abbildung 5.13 zeigt, dass der *Kernel* somit erst durch die Positionsdaten (beispielhaft mit rot und blau umrandet), danach durch die Rotationsdaten (mit grün und orange umrandet) und weiter durch die anderen Eigenschaften läuft. Die hier beschriebene Sequenz des Durchlaufs wird mittels eines *Strides* von $1 \cdot 3$ ermöglicht. Die Daten werden in ein Format $150 * 5$ komprimiert.

In der zweiten *Faltungsschicht* wird ein *Kernel* der Größe $2 \cdot 5$ angewendet. Dieser *Kernel* durchläuft die neu berechneten Daten und betrachtet dabei jeweils zwei aufeinanderfolgende Zeitpunkte miteinander. Nach dem Durchlauf dieser Schicht beträgt das Ausgabeformat der Daten $1 * 148$. An diese Schicht schließt sich ein *Feed-Forward Netzwerk* an, das aus drei Schichten besteht. Die erste Schicht dieses Netzwerks besteht aus 148 Neuronen. Diese Schicht ist mit einer zweiten Schicht von 128 Neuronen verbunden, die wiederum mit der Ausgabeschicht des Netzwerks verknüpft ist.

Beim dritten *CNN* werden die Daten dreidimensional strukturiert. Beispielhaft ist in Abbildung 5.14 dargestellt, wie die Daten in das Netz gegeben werden. Die Breite des dort dargestellten Würfels gibt die Anzahl der *Channels* an, in die jeweils die einzelnen Eigenschaften sowie die Rotation w, erweitert um das *Zero-Padding*, gegeben werden. In die Tiefe des Würfels werden die Zeitschritte und in die Höhe die jeweiligen x-, y- und z-Werte (bei Rotation w der Wert erweitert um das *Zero-Padding*) abgebildet. Der *Kernel* betrachtet jede Eigenschaft, drei Zeitpunkte und die drei Werte jeder Eigenschaft, welches in rot dargestellt ist. Danach iteriert er durch die weiteren Daten, was beispielhaft für die erste Iteration in blau dargestellt ist. Die Ausgabe der Daten besteht aus einem Format $1 * 148$. Auch in diesem Netz schließt sich ein *Feed-*

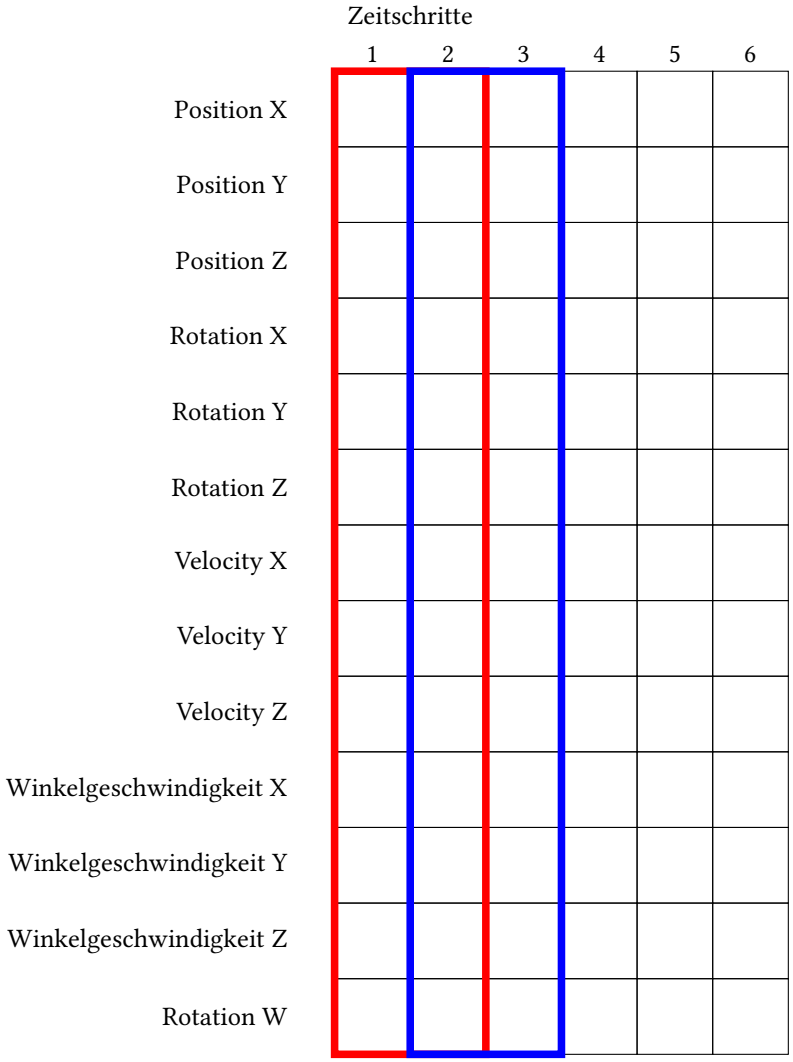


Abbildung 5.12: Konzept CNN 1

Forward Netzwerk an. Die erste Schicht besteht aus 148 Neuronen, die wiederum mit einer Schicht bestehend aus 128 Neuronen verbunden ist. Es folgt eine vollvermaschte Ausgabeschicht für die Klassifikation.

Neben den drei *CNNs* wurde ein *RNN* verwendet. Das *RNN* wurde nicht mit den von *PyTorch* bereitstehenden RNN-Schichten [21], sondern anhand eines Tutorials[22] implementiert. In Abbildung 5.15 ist dargestellt, wie das *RNN* aufgebaut ist. Dabei sind alle Komponenten des Netzes in violett gefärbt. Die Eingaben in das Netz sind blau

[21] PyTorch-Dokumentation RNN: `https://pytorch.org/docs/stable/generated/torch.nn.RNN.html`, letzter Aufruf: 22.09.2023

[22] Coding a Recurrent Neural Network (RNN) from scratch using PyTorch: `https://medium.com/@VersuS_/coding-a-recurrent-neural-network-rnn-from-scratch-using-pytorch-a6c9fc8ed4a7`, letzter Aufruf: 22.09.2023

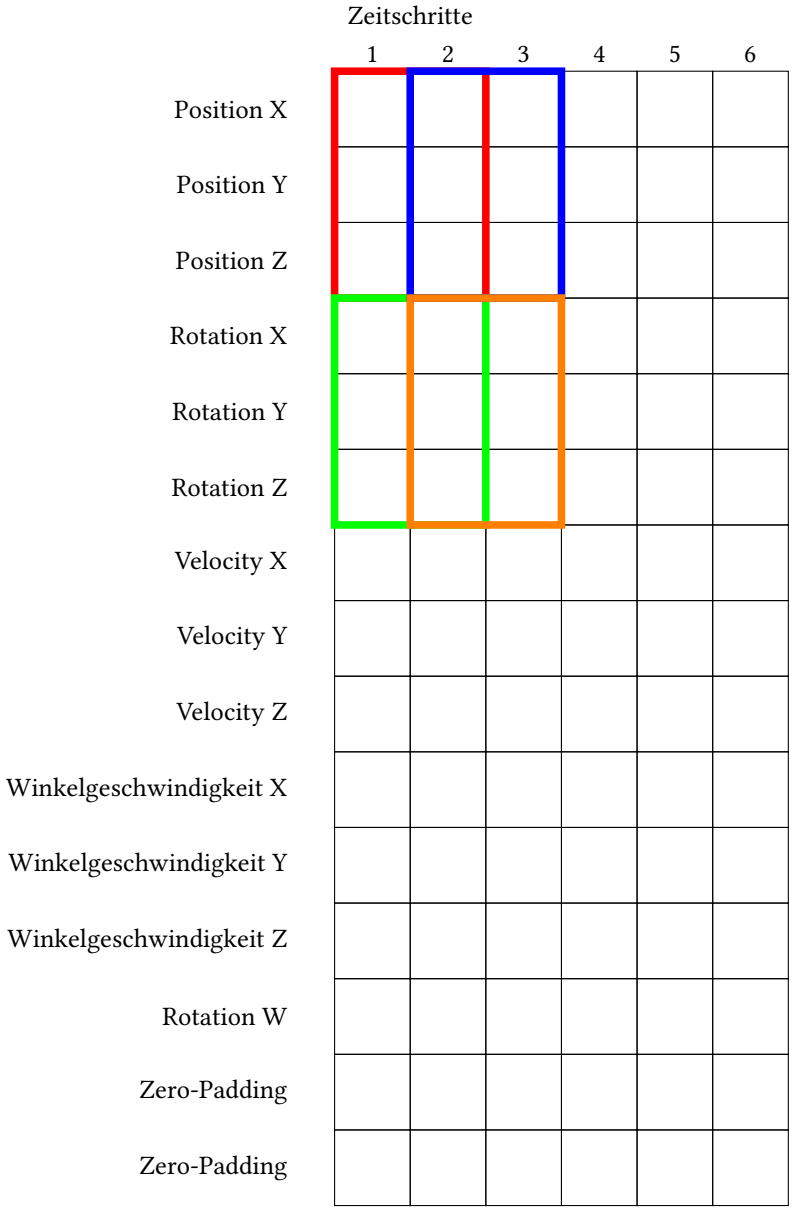


Abbildung 5.13: Konzept CNN 2

gefärbt und die Ausgaben sind grün dargestellt. Bei der Klassifizierung eines Schlages wird zunächst der *Hiddenstate* mit null initialisiert. Der *Hiddenstate* soll die Historie der vorherigen Sequenzen abbilden. Für eine Klassifikation wird jede einzelne Sequenz dem Netz zugeführt: Die Daten der ersten Sequenz werden in eine Eingabeschicht, bestehend aus 13 Neuronen, übergeben. Diese ist mit einer Schicht, bestehend aus 32 Neuronen, verbunden. Der *Hiddenstate* wird gleichzeitig in eine Schicht bestehend aus 32 Neuronen gegeben. Diese Schicht ist mit einer Schicht bestehend aus 32 Neuronen verbunden. Die Ausgaben der Hiddenschicht und der Eingabeschicht

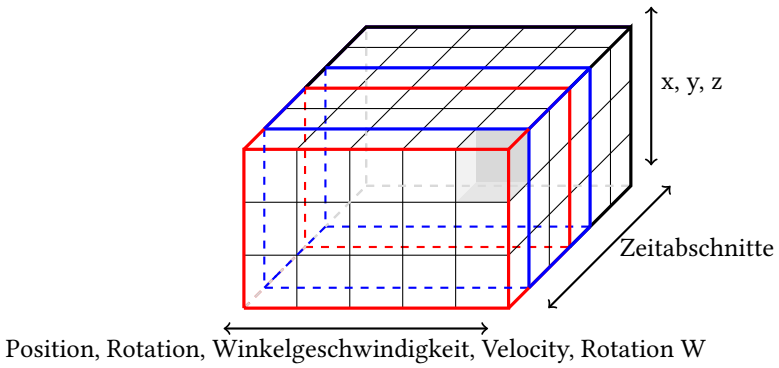


Abbildung 5.14: Konzept CNN 3

werden addiert und mit der Aktivierungsfunktion *tanh* wird der neue *Hiddenstate* berechnet. Die *tanh* wurde aufgrund ihrer Nichtlinearität[23] gegenüber *ReLU* eingesetzt. Ebenfalls wurden mit *tanh* bessere Ergebnisse erzielt, worauf bei der Beschreibung des Trainings näher eingegangen wird. Der *Hiddenstate* wird zwischengespeichert. Mithilfe der Aktivierungsfunktion *Softmax* wird in der Ausgabeschicht die Ausgabe und der *Loss* berechnet. Folgend wird die nächste Sequenz als Eingabe genommen. Der vorherige, zwischengespeicherte *Hiddenstate* wird ebenfalls als Eingabe genutzt. Nachdem die letzte Sequenz durch das Netz verarbeitet wurde, wird in der Ausgabe die entsprechende Klasse ausgegeben. Im Training wird nach der letzten Ausgabe die *Backpropagation* anhand der addierten *Loss*-Werte jeder Sequenz durchgeführt.

Das *LSTM* besteht aus zwei *LSTM*-Schichten [25]. Die Eingabegröße der ersten *LSTM*-Schicht beträgt 13 Features, da diese jeweils einen Zeitpunkt darstellen. Die zweite *LSTM*-Schicht ist mit einem *Feed-Forward Netzwerk* verbunden. Die erste Schicht des *Feed-Forward Netzwerkes* ist 16 Neuronen groß. Diese Schicht ist mit der Ausgabeschicht verbunden.

Training der Netze

Für das Training wurden Daten aus 5000 Vorhand-, 5000 Rückhandschlägen und 5000 Aufschlägen in 80% Trainingsdaten und 20 % Testdaten aufgeteilt. Diese Daten bestehen aus den aufgenommenen und generierten Daten. Um zu überprüfen, ob das Modell auf den reinen realen Daten performt, wurden vor Aufteilung in Test- und Trainingsdaten jeweils 50 Schläge aus den jeweiligen Schlagarten als Validierungsdatensatz zusammengestellt. Als *Hyperparameter* für das *LSTM* und das *RNN* wurde zunächst der Optimierer *Adam* und für die *CNNs* der *SGD mit Momentum* genutzt, da beide Optimierer eine der verbreiteten Optimierer im Feld neuronaler Netze sind [Haji and Abdulazeez, 2021]. Da das Training mit diesen erfolgreich war (s. Abschnitt 5.4.2), wurde kein anderer Optimierer getestet. Die *Lernrate* wurde für das *RNN* und das *LSTM* auf 0,001 und für die *CNNs* auf 0,005 festgesetzt, um Minima beim *Gradientenabstieg* des Optimierens nicht zu überspringen, aber auch kein zu langes Training

[23] PyTorch-Dokumentation RNN: `https://pytorch.org/docs/stable/generated/torch.nn.RNN.html`, letzter Aufruf: 22.09.2023

[25] PyTorch-Dokumentation LSTM: `https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html`, letzter Aufruf: 22.09.2023

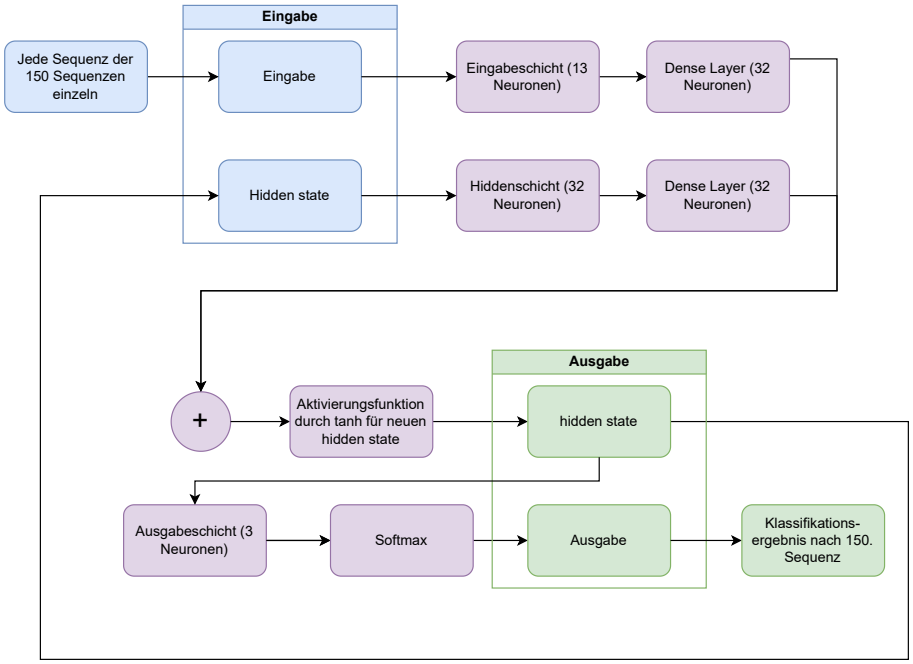


Abbildung 5.15: Architektur des genutzten RNNs[24]

zu haben [Haji and Abdulazeez, 2021]. Für die *Lossfunktion* wurde die für die Klassifikation bestimmte *Cross-Entropy* verwendet.[26] Als Epoche wurde 250 für die *CNNs*, 50 für das *LSTM* und 20 für das *RNN* festgelegt. Um das *Overfitten* des *LSTM* und des *RNN* zu verhindern und die Modelle zu generalisieren, wurde als *Regulierungsmethode* der *L2-Regulierer* eingesetzt. Als *Batchsize* wurden 8, 16, 32 und 64 getestet [Lin, 2022] und sich auf 32 festgelegt, da hier der Trainingsverlauf am besten war. Das Training vom *RNN* erzielte mit einer *Batchsize* von 16 bessere Ergebnisse hinsichtlich der Modelperformanz. Für das *RNN* wurde eine Größe der Hiddenschicht von 16, 32 und 64 Neuronen getestet. Am besten war das Training beim *RNN* mit einer Größe der Hiddenschicht von 32 Neuronen. Für das *RNN* wurden mit der Wahl der *ReLU*-Aktivierungsfunktion keine guten Ergebnisse erzielt, weshalb, wie im Vorangegangenen beschrieben, die *tanh*-Aktivierungsfunktion verwendet wurde, welche keine Linearität wie die *ReLU*-Funktion aufweist (s. Abschnitt 3.1.3). Mit der *tanh*-Aktivierungsfunktion wurden bessere Ergebnisse erzielt.

Bei der Ausführung der Implementierung des Trainings in *PyTorch*[17] wurden die folgenden Schritte pro Epoche durchgeführt:[26]

- *Labels* werden durch das Netz zur Vorhersage der entsprechenden *Features* gegeben
- *Loss* und Genauigkeit der Vorhersage wird berechnet, indem die tatsächlichen *Labels* zu den vom Netz berechneten *Labels* verglichen werden

[26] PyTorch Neural Network Classification: `https://www.learnpytorch.io/02_pytorch_classification/`, letzter Aufruf 09.09.2023

- Auf dem Modell wird `model.zero_grad()` aufgerufen
- Der *Loss* wird zurück durch das Netz gegeben: `loss.backward()`
- Der Optimierer macht einen *Gradientenabstieg*: `optimizer.step()`

Um beim Training des *RNNs explodierende Gradienten* präemptiv zu vermeiden, wurde hier zusätzlich `nn.utils.clip_grad_norm_()` vor dem Optimiererschritt aufgerufen[27], [28].

In den folgenden Abbildungen 5.16 bis 5.25 ist der *Loss*- und Accuracy-Verlauf der einzelnen Netze zu erkennen. Dabei ist in rot der Trainingsdatensatz, in grün der Testdatensatz und in blau der Validierungsdatensatz dargestellt. Alle Abbildungen zeigen eine Accuracy, die während des Trainings auf 95% - 100% ansteigt. Die *CNNs overfitten* leicht und das *LSTM* ist augenscheinlich am erfolgreichsten. Wie die Abbildungen zeigen, wurden sowohl für das *LSTM* als auch vor allem für das *RNN* eine geringe Anzahl an Epochen verwendet: Bei früheren Trainingsdurchläufen kam es häufiger zum *Overfitting* der beiden Modelle, weshalb angenommen wurde, dass die Modelle zu lange trainiert haben. Als Konsequenz wurde die Trainingszeit verringert. Es stellte sich jedoch heraus, dass die Modelle je nach Randomsplit der Test- und Trainingsdaten unterschiedlich schnell overfitteten. Zum Zeitpunkt, als das rausgefunden wurde, war jedoch keine Zeit für ein vollständiges neues Training des *RNN* vor der Studie. Dennoch erreichten die Accuracy-Werte bei beiden Netzen 95%-100%. Auch war das *RNN* das Netz, was in der Echtzeitklassifizierung am besten performte, worauf im Folgenden noch näher eingegangen wird.

Ebenfalls zeigen die Abbildungen, dass sowohl *Loss* als auch Accuracy des Validierungsdatensatzes unter Accuracy und *Loss* des Test- und Trainingsdatensatz liegt, was daran zu erklären ist, dass der Validierungsdatensatz zu klein gewählt ist und somit fehlerhaft klassifizierte Daten deutlich mehr ins Gewicht fallen oder der Validierungsdatensatz möglicherweise ungünstige Daten beinhalten könnte, die zu fehlerhaften Klassifizierungen führen. Ebenfalls kann es sein, dass die Datengenerierung nicht genau genug performt. Der Einsatz der Datengenerierung wird in der Evaluierung der Studienergebnisse 7 näher evaluiert.

[27] Understanding Gradient Clipping (and How It Can Fix Exploding Gradients Problem): `https://neptune.ai/blog/understanding-gradient-clipping-and-how-it-can-fix-exploding-gradients-problem`, letzter Aufruf: 22.09.2023

[28] PyTorch-Dokumentation nn.utils.clip_grad_norm_: `https://pytorch.org/docs/stable/generated/torch.nn.utils.clip_grad_norm_.html`, letzter Aufruf: 22.09.2023

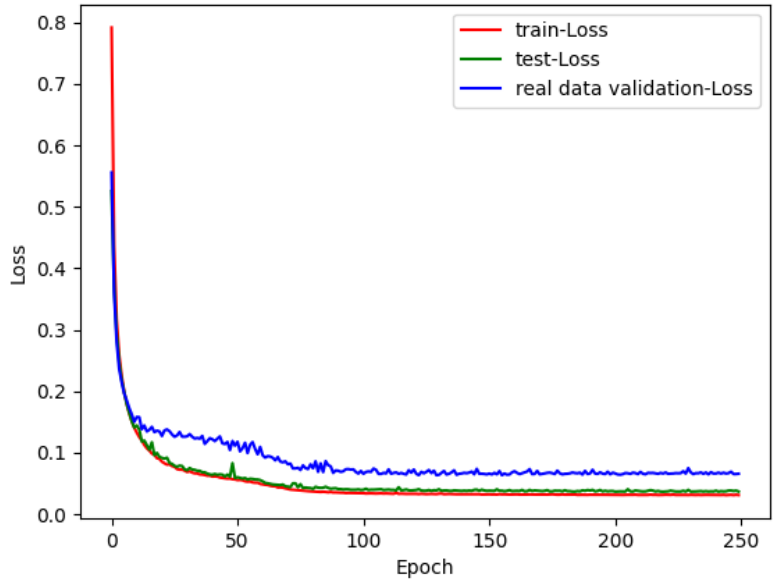


Abbildung 5.16: CNN 1 Lossverlauf. Eigene Darstellung.

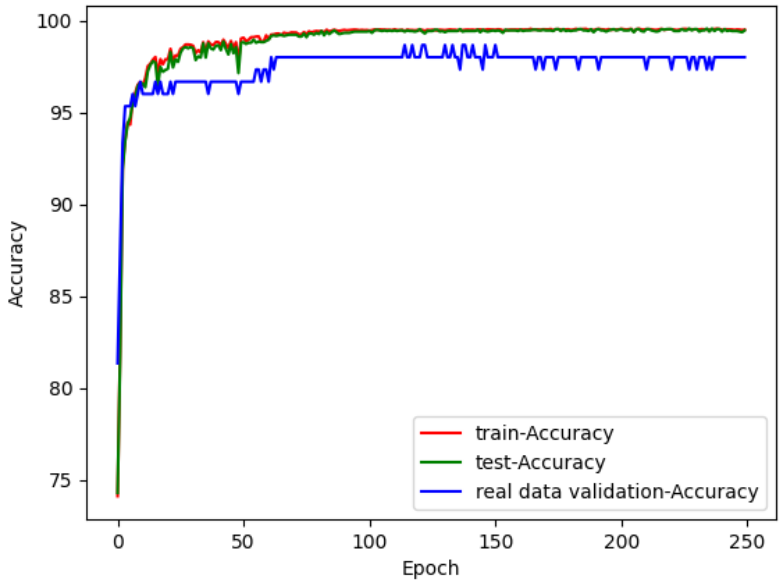


Abbildung 5.17: CNN 1 Accuracyverlauf. Eigene Darstellung.

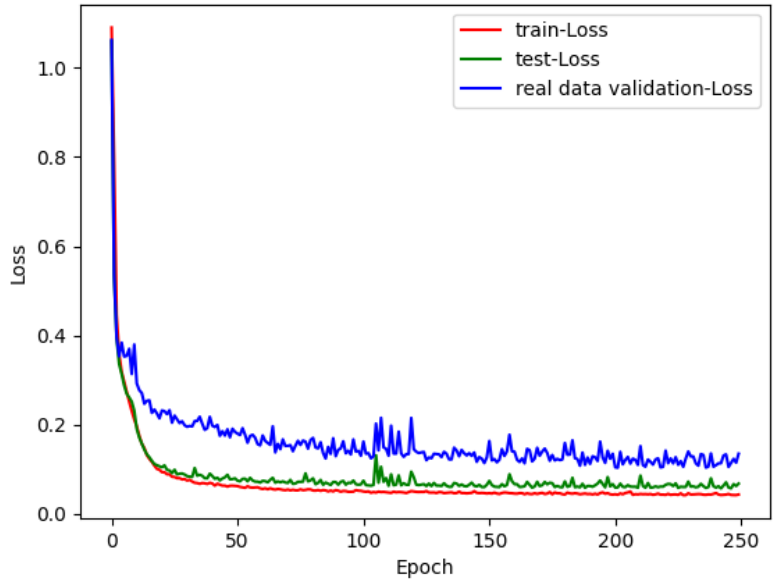


Abbildung 5.18: CNN 2 Lossverlauf. Eigene Darstellung.

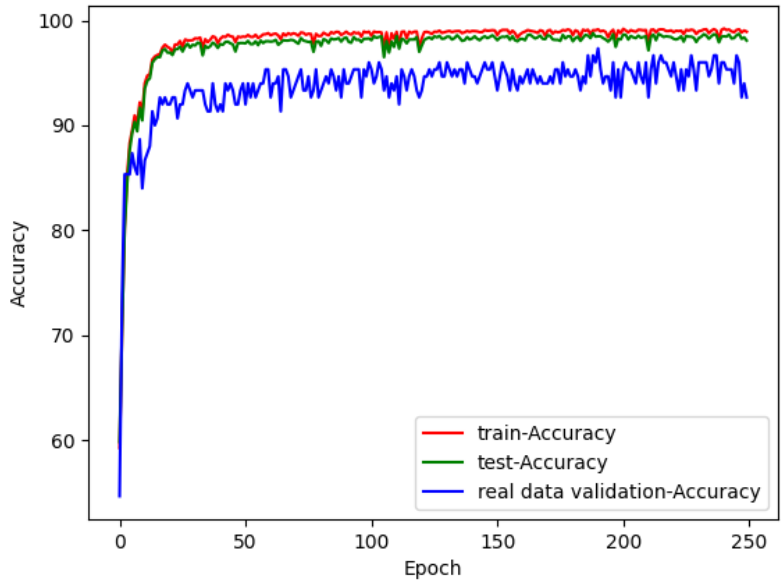


Abbildung 5.19: CNN 2 Accuracyverlauf. Eigene Darstellung.

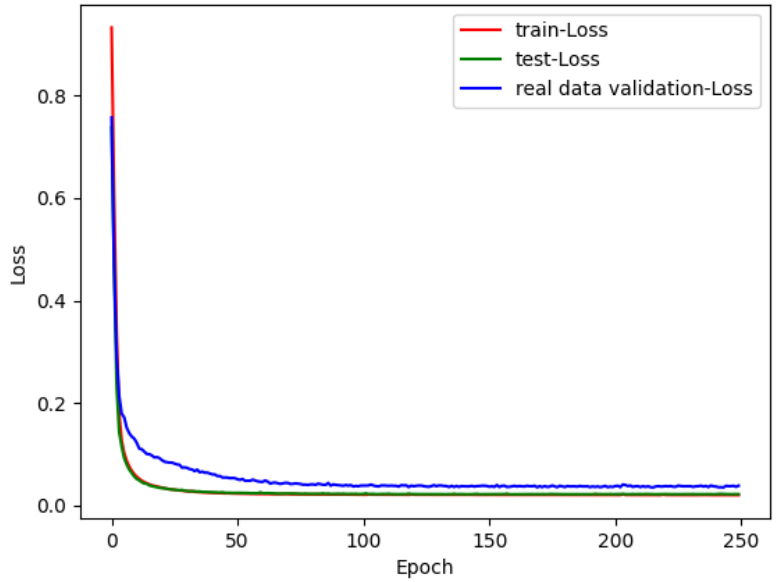


Abbildung 5.20: CNN 3 Lossverlauf. Eigene Darstellung.

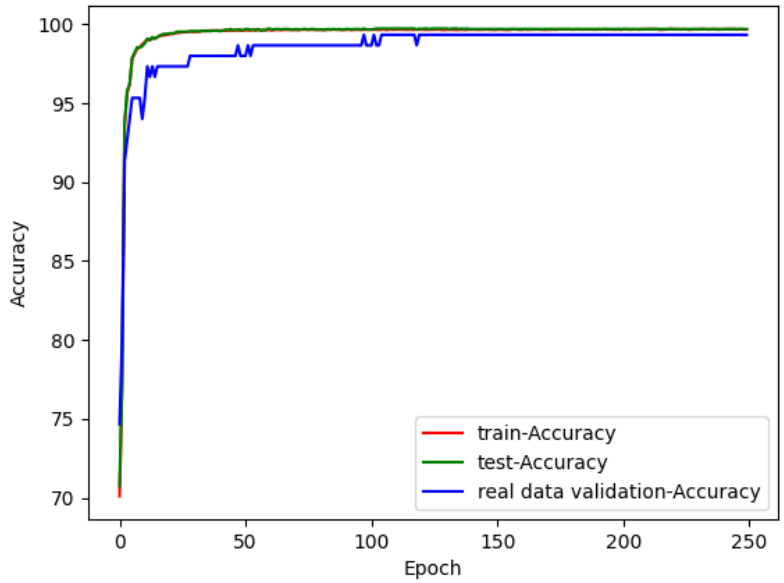


Abbildung 5.21: CNN 3 Accuracyverlauf. Eigene Darstellung.

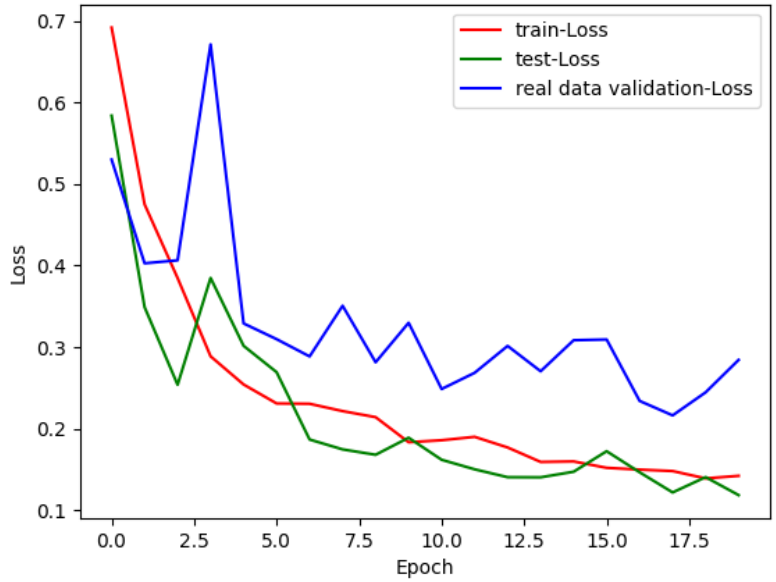


Abbildung 5.22: RNN Lossverlauf. Eigene Darstellung.

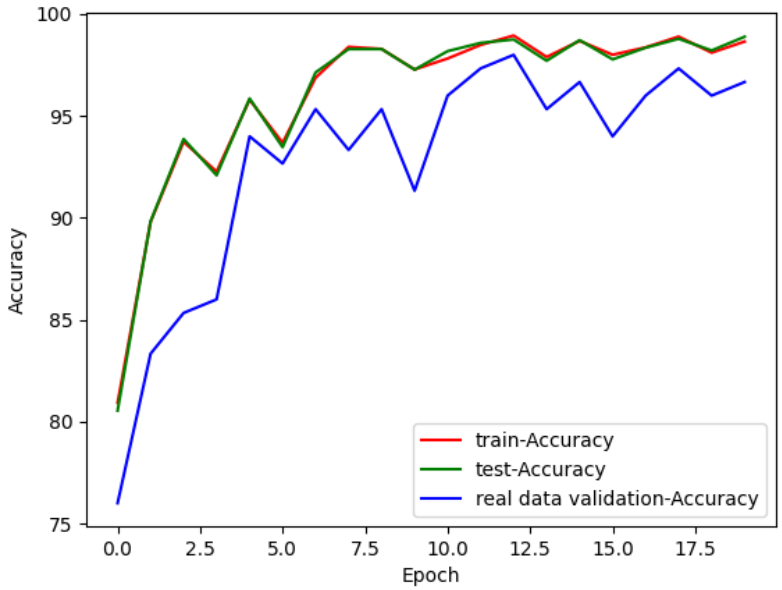


Abbildung 5.23: RNN Accuracyverlauf. Eigene Darstellung.

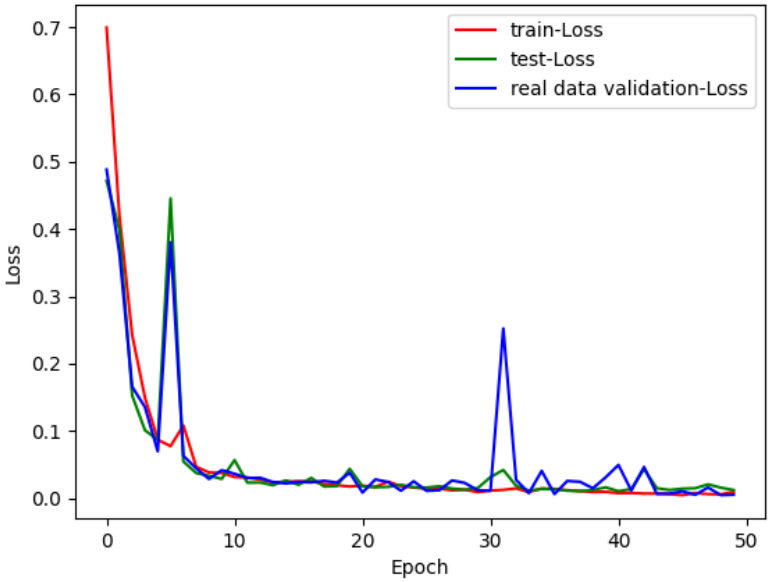


Abbildung 5.24: LSTM Lossverlauf. Eigene Darstellung.

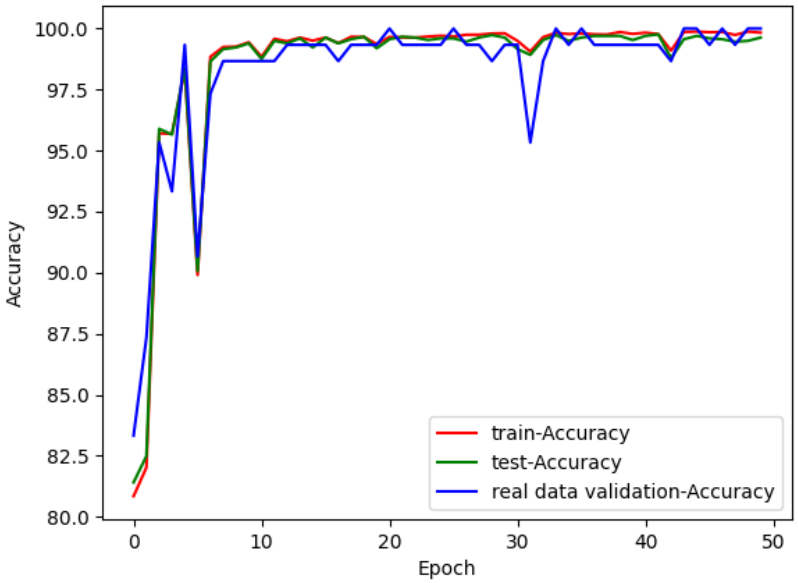


Abbildung 5.25: LSTM Accuracyverlauf. Eigene Darstellung.

Test der Netze und deren Anpassung

Nachdem die Netze trainiert wurden, wurde zunächst ein Test aller Netze durchgeführt, um das für die Studie am besten passende Netz auszuwählen. Dabei wurden im Datenaufnahme-Branch mithilfe der Aufschlagfunktion des Bots Vorhände, Rückhände und Aufschläge in Echtzeit klassifiziert. Die Ergebnisse aus dem Test sind in den Tabellen 5.4.2 für den Aufschlag, 5.4.2 für die Vorhand und in 5.4.2 für die Rückhand aufgelistet. Für den Test sind hier die Ergebnisse einer älteren Version des *RNNs* und des *LSTM* abgebildet. Die Tabellen zeigen, dass das *RNN* für die Vorhand und das *LSTM* für die Rückhand am besten performen. Vor allem bei der Vorhand erkannten viele Netze einen Aufschlag, worauf im nächsten Abschnitt zur „Wahl des Netzes für die Studie„ nochmal eingegangen wird. Da das *LSTM* und das *RNN* einige fehlerhafte Klassifizierungen aufwiesen, wurden weitere manuelle *Hyperparameteroptimierungen* für diese durchgeführt. Sie wurden erneut mit einem *L2-Regularisierer* trainiert, um die Modelle verstärkt zu generalisieren. Die abschließenden Ergebnisse des Trainings sowie der finale Aufbau der Netze sind auch jene, die im Abschnitt für das Training verwendet und bei der Beschreibung der Netze erläutert werden.

Netz	klassifizierte Vorhände	klassifizierte Rückhände	klassifizierte Aufschläge
CNN 1	0	0	100
CNN 2	0	0	100
CNN 3	0	0	100
RNN	0	0	100
LSTM	0	0	100

Tabelle 5.1: Klassifikationsergebnisse für Aufschlag

Netz	klassifizierte Vorhände	klassifizierte Rückhände	klassifizierte Aufschläge	nicht ermittelbar
CNN 1	82	0	46	1
CNN 2	84	1	44	0
CNN 3	116	0	13	0
RNN	120	0	9	0
LSTM	118	5	6	0

Tabelle 5.2: Klassifikationsergebnisse für Vorhand

Netz	klassifizierte Vorhände	klassifizierte Rückhände	klassifizierte Aufschläge
CNN 1	1	236	9
CNN 2	0	237	9
CNN 3	0	240	6
RNN	0	219	27
LSTM	0	240	6

Tabelle 5.3: Klassifikationsergebnisse für Rückhand

Da der Accuracy- und *Loss*-Verlauf des *LSTM* während des abschließenden Trainings bessere Werte erreichte als das *RNN*, wurde das *LSTM* zunächst als Netz ausgewählt.

Als der Gegenspieler im Spielraum so weit implementiert war, dass gegen ihn gespielt werden konnte, wurde ein weiterer Test im Spielraum durchgeführt. In dieser Untersuchung wurden keine präzisen Ergebnisse erfasst, da aufgrund der komplexen Spielumgebung eine genaue Erfassung einzelner Schläge erschwert wurde, was zur Ungenauigkeit der Ergebnisse führte. Es ist aufgefallen, dass ausnahmslos alle Schläge von allen Netzen als Aufschläge erkannt wurden. Begründet werden kann dieses dadurch, dass die Version des Spiels, indem die Trainingsdaten aufgezeichnet wurden und der finalen Version unterschiedliche Koordinaten aufweisen, weil der Spielraum im Laufe der Zeit versehentlich verschoben wurde und für die Datenaufnahme ein gesonderter, älterer Branch genutzt wurde. Im neusten Spielraum wurden nur Aufschläge aufgenommen, weil der Bot dort bis zum Ende nicht zur Verfügung stand. Somit unterscheiden sich die Koordinaten der Aufschläge zu den Koordinaten der anderen Schläge. Nur die Aufschlagdaten waren somit den Koordinaten im echten Spiel ähnlich, was dazu führte, dass alle Schläge als Aufschläge erkannt wurden. Aus diesem Grund wurden die Daten auf den ersten Datenpunkt normalisiert. Dazu wurde in jedem Schlag der erste Datenpunkt für x, y und z abgespeichert und auf 0 gesetzt. Die abgespeicherten Werte wurden jeweils von allen weiteren x-, y- und z-Koordinaten der jeweiligen Zeitpunkte eines Schlages abgezogen. Im Effekt sind also alle folgenden Datenpunkte relativ zu einer ersten Koordinate, die sich am Ursprung des Koordinatensystems befindet. Eine andere Möglichkeit wäre gewesen, den Schlag-Ball-Kontakt als 0-Punkt zu betrachten. Allerdings wurden, wie bereits beschrieben, die 100 Frames vor dem Kontakt und die 50 Frames nach dem Kontakt aufgenommen. Der genaue Zeitpunkt, an dem sich Ball und Schläger getroffen haben, war also nicht Bestandteil des Datensatzes. Nach Veränderung wurden die Netze neu trainiert. Die Ergebnisse des Trainings wurden bereits in Abschnitt 5.4.2 aufgeführt.

Wahl des Netzes für die Studie

Wie bereits beschrieben, wurde zunächst als Netz für die Echtzeitklassifizierung das *LSTM* eingestellt. Durch die größere Variation des Rückspielens vom Bot im Spielraum war es schwieriger, mehrere gleiche Schlagarten hintereinander durchzuführen. Aus diesem Grund wurde versucht, jeden Aufschlag zu zählen und dann jeden Rückschlag mit der gleichen Schlagart zu beantworten. Dabei konnte festgestellt werden, dass die durchgeführten Vorhand- und Rückhandschläge vermehrt als Aufschläge erkannt wurden. Als Begründung kann die veränderte Spielumgebung zur Aufnahmeumgebung genannt werden: Dadurch dass bei der Aufnahme der Schläge die stets identischen Aufschläge des Bots verwendet wurden, war bekannt, wo der Ball aufkommen wird und der Schlag wurde intuitiv darauf vorbereitet. Im echten Spiel werden Schläge spontaner durchgeführt, sodass ältere Sequenzen für den Schlag irrelevanter werden.

Da ein *LSTM* sich dadurch auszeichnet, dass Langzeitabhängigkeiten in den Daten gelernt werden [Hochreiter and Schmidhuber, 1997], kann hier die Begründung angestellt werden, wieso vermehrt Aufschläge erkannt wurden. Um die Vermutung zu bestätigen, wurden die anderen Netze getestet. Die *CNNs* zeigten ebenfalls Probleme bei der Klassifikation von im Spiel gespielten Vorhänden als solche. Auch im Test im Trainingsraum wurden diese Probleme trotz eines erfolgreichen Trainings der *CNNs* identifiziert. Dies legt nahe, dass *CNNs* im Vergleich zu dem *RNN* und *LSTM* eine geringere Generalisierungsfähigkeit auf unbekannte Daten aufweisen und wurden daher nicht weiter in Betracht gezogen. Es fiel auf, dass das *RNN* die Rückschläge korrekt erkennen konnte. Dies lässt sich durch die Tatsache erklären, dass bei der Klassifizie-

rung durch das *RNN* insbesondere die jüngsten Sequenzen von größerer Bedeutung sind [Bengio et al., 1994]. Aus diesem Grund wurde das *RNN* für die Echtzeitklassifizierung eingesetzt.

Einbau der Klassifizierung in das Tischtennisspiel

Um die Klassifizierung in das Spiel einzubauen, wurde eine Route `object_data` erstellt, an welche das entsprechende JSON des Schlages gesendet wird. Dieses wird weitergeleitet an die Funktion `handle_json` der Klasse `Classifier`, welche in Abbildung 5.26 dargestellt ist. Der Ablauf der Klassifizierung ist weiterhin in Abbildung 5.28 im Kontext der Fehlerevaluierung dargestellt. Ist die JSON eine Liste, wird die Methode `classify_stroke` aufgerufen, in welcher zunächst die GUID (ein kollisionsfreier, zufälliger Identifikator)[29] aus dem Schlag geladen wird und die relevanten Daten in einen Tensor überführt werden. Nachfolgend wird mit dem *RNN* der Schlag klassifiziert und mit der entsprechenden GUID in dem Dictionary `stroke_classification` abgespeichert.

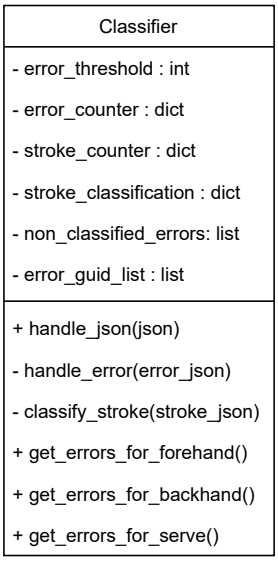


Abbildung 5.26: Klasse `Classifier`. Eigene Darstellung.

5.4.3 Klassifizierung der Schläge und Fehlerevaluierung

In diesem Abschnitt wird die Fehlerevaluierung erläutert. Dabei wird zunächst das generelle Konstrukt beschrieben. Danach wird beschrieben, wie der Chatbot in die Fehlerevaluierung eingebunden ist.

Konzept

Wird im Spiel ein Fehler festgestellt, wird das Frontend durch den `Gamestatetracker` registriert. Danach wird ein entsprechendes Fehlerobjekt an das Backend über die

[29] GUID Struktur: `https://learn.microsoft.com/de-de/dotnet/api/system.guid?view=net-7.0`, letzter Aufruf: 01.10.2023

Route `object_data` gesendet. Dieses Objekt ist in Abbildung 5.27 in Beziehung zu dem Schlag dargestellt. Die Abbildung zeigt, dass sowohl Schlagobjekt als auch das Fehlerobjekt eine `racketHitGUID` abbilden, mit der der Fehler zum entsprechenden Schlag im Backend zugeordnet werden kann.

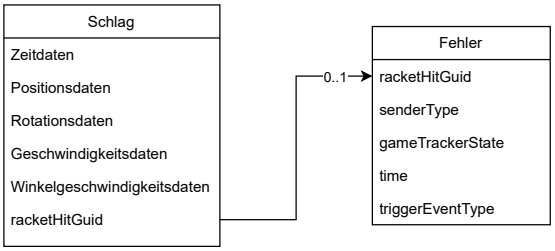


Abbildung 5.27: Beziehung zwischen Schlag und Fehler. Eigene Darstellung.

Das Fehlerobjekt wird an dieselbe Route geschickt, zu welcher auch die Schlagdaten geschickt werden. Der Prozess der Klassifizierung wurde bereits in Abschnitt 5.4.2 beschrieben. Der Prozess der Fehlerevaluierung und Klassifizierung der Schläge wird in Abbildung 5.28 dargestellt. Bei Senden eines Fehlerobjektes wird die GUID aus dem Fehlerobjekt extrahiert und in eine Liste aller Fehler abgespeichert. Ist zu der GUID bereits ein Schlag klassifiziert worden, kann das Fehlerobjekt dem entsprechenden Schlag zugeordnet werden. Für jede Art von Schlag wird die Anzahl von Fehlern gezählt. Darauf aufbauend war die Idee, nach einer gewissen Anzahl an Fehlern den Chatbot anzustoßen, was im nächsten Abschnitt näher beschrieben wird.

Chatbot-Integration

Von Seiten des Chatbots sollte der Nutzer darüber informiert werden, was für einen Fehler er gemacht hat, und ihm vorgeschlagen werden, das entsprechende Tutorial im Trainingsraum nochmal zu hören und zu üben. Dafür wurden zwei Pläne der Umsetzung festgestellt:

1. Wenn der Fehlerschwellwert für eine Schlagart überschritten wird, soll der Chatbot getriggert werden.
   Der Chatbot fragt, ob der Nutzer das Tutorial zum entsprechenden fehlerhaften Schlag hören und den Trainingsraum betreten möchte. Wenn ja, wird mittels einer *Custom Action* von Rasa[25] eine Information an die API gesendet, dass der Trainingsraum betreten und geübt werden möchte. Diese Information kann dann vom Frontend mittels HTTP-GET-Request abgefragt werden.

2. Der Vorschlag wird dem Nutzer akustisch unterbreitet, den Trainingsraum zu betreten und mithilfe des entsprechenden Tutorials die Schläge zu verbessern.
   Mit einem TTS-Skript wird eine Audiodatei im Backend erstellt und abspielt, um dem Nutzer den Vorschlag für das Tutorial bezüglich des Schlags, bei dem der Fehlerschwellwert überschritten wurde, mitzuteilen.
   Nun kann der Nutzer im Spiel selbst entscheiden, ob der Trainingsraum betreten und geübt werden soll.

Beim ersten Plan hätte der Chatbot bei jeder Erreichung des Fehlerschwellenwertes den Nutzer gefragt, ob er den Trainingsraum betreten möchte. Dies hätte jedes mal

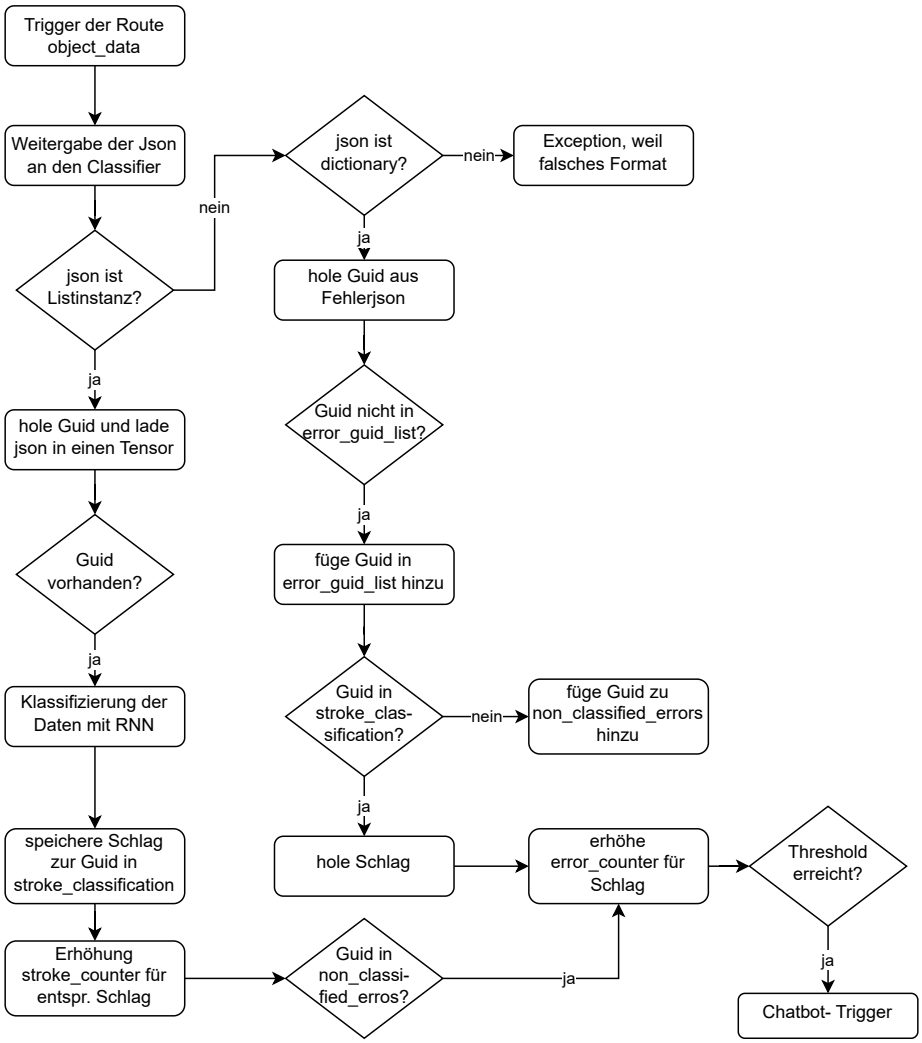


Abbildung 5.28: Ablaufdiagramm zur Klassifizierung und Fehlerevaluierung. Eigene Darstellung.

eine Reaktion des Nutzer erfordert, was möglicherweise als störend empfunden werden könnte. Beim zweiten Plan werden nur akustische Vorschläge unterbreitet, ohne dass der Nutzer zwingend auf diese reagieren muss. Deswegen wurde sich für den zweiten Plan entschieden, der umgesetzt worden ist. Aber auch der erste Plan ist bereits grundlegend implementiert, sodass hier noch eine Erweiterungsmöglichkeit in der Zukunft besteht, siehe Kapitel 9.

5.4.4 Ursachen für die lange Ausführungszeit der Tests und der Installation der Packages

Im Laufe der Implementierung stellte sich heraus, dass die Testlaufzeiten besonders hoch sind. Dies lag zum Großteil an den Tests der TTS-Funktionalitäten, beschrieben in Kapitel 5.4.1. Zeitkritisch sind hauptsächlich diejenigen Funktionen, die die Audiodateien für die Tutorials, Vorschläge und einfache Fehler erstellen und ausgeben. Es wird jede dieser Funktionen getestet, daher wird jede Audiodatei einzeln ausgegeben. Da vor allem die Tutorials recht lang sind und die Sprachausgabe eher langsam erfolgt, dauern diese Tests in Summe lange.

Außerdem wird das Starten des Rasa[25]-Servers selbst getestet, was zwischen einer und anderthalb Minuten dauert. Für die Chatbot-Funktionen, die abhängig vom laufenden Rasa[25]-Server sind, muss der Rasa[25]-Server zunächst gestartet werden, bevor die entsprechenden Tests gestartet werden. Dies erhöht also ebenfalls die Laufzeit der Tests.

Weiterhin nahmen auch die CI/CD-Pipeline-Jobs viel Zeit in Anspruch: Grund dafür war, dass die Installation der nötigen Python-Pakete teilweise je Job erneut ausgeführt wurde. Dabei ist vor allem die Bibliothek Rasa[25] eine sehr umfangreiche, die viele Abhängigkeiten hat, die mit installiert werden müssen. Eine Lösung für dieses Problem der langen Installationszeiten ist in Kapitel 5.4.6 dargestellt.

5.4.5 Ansatz zur Optimierung der Testlaufzeiten: Parallelisierung

Um die Ausführungszeit der Tests zu reduzieren und somit die Effizienz der CI/CD-Pipeline zu erhöhen, wurde eine Parallelisierung der Tests geplant. Hierzu wurden zunächst folgende Schritte vorgenommen:

- Identifikation des Bedarfs einer Parallelisierung: Bewertung der Ausführungszeit der Testsuite und Feststellung, ob eine parallele Ausführung erforderlich ist. Die Pipeline benötigte mit 22 Minuten eine beträchtliche Zeit, womit fortwährend eine Parallelisierung der Tests angestrebt wurde.
- Analysieren der Testabhängigkeiten: Es wurde festgestellt, dass keine Abhängigkeiten oder sonstige Wechselwirkungen zwischen verschiedenen Unit-Tests bestanden.
- Aufteilen der Test-Suite: Die Unit-Tests wurden entsprechend ihrer Funktionalitäten einzeln implementiert, sodass es nicht nötig war, die Tests weiter aufzuteilen.
- Implementierung der parallelen Durchführung: Letztlich wurden zur Implementierung der Parallelisierung mit `PyTest` die Plugins `pytest-xdist` und `pytest-parallel` nacheinander implementiert, um die Effizienz der parallelisierten Ausführung zu testen. Dies erfolgte, nachdem die parallele Ausführung konfiguriert wurde, so zum Beispiel die Anzahl der zu parallelisierenden Prozesse.

Im Anschluss galt es zu testen, ob die Parallelisierung erfolgreich aktiviert wurde. Hierzu wurden die CI/CD-Konfigurationen, sowie Parallelisierungseinstellungen der `python-testX.X` Pipeline-Jobs, wobei `X.X` die jeweilige Python-Version ist, überprüft.

Um festzulegen, welche Tests parallel laufen sollen, mussten diese mit dem `parallel`-Keyword versehen werden.

Mit der Nutzung mehrerer Threads ging auch eine höhere Ressourcennutzung einher, somit musste abschließend getestet werden, ob die Parallelisierung der gewünschten Laufzeitreduzierung geführt hat, ohne die Ressourcennutzung disproportional zu erhöhen. Bereits bei lokalen Testdurchläufen konnte keine spürbare Laufzeitverringerung festgestellt werden, während die lokal genutzte IDE *PyCharm* bei vereinzelten Durchläufen abstürzte. Aufgrund dessen wurde sich dazu entschieden, die Parallelisierung nicht einzuführen und weiterzunutzen.

5.4.6 Ansatz zur Optimierung der Testlaufzeiten: Kaniko Cache Nutzung

Für die Implementierung wurden einige Softwarepakete als Abhängigkeit verwendet. Die Python-Pakete waren nicht nur zahlreich (ca. 20 Pakete), sondern auch umfangreich. Im Endeffekt lag die Größe des Docker Image bei knapp 7GB. Durch das zeitintesive Herunterladen und Installieren brauchte die CI/CD-Pipeline sehr viel Zeit für alle Jobs, welche die Abhängigkeiten erforderten. Häufiger wurde über eine halbe Stunde benötigt, bis die Pipeline durchlaufen ist.

Um dieses Problem zu lösen, wurden Caches verwendet, sodass die Installation der benötigten Python-Packages nicht bei jedem Job ausgeführt werden muss. Ein Cache kann eine Menge an benötigten Python-Packages speichern. Nachfolgende Jobs, die denselben Cache verwenden, müssen diese gecachten Packages nicht erneut installieren, sondern nur den Cache entpacken, weswegen sie schneller ausgeführt werden.

Die Durchlaufzeit der CI/CD-Pipeline, konnte so von über einer halben Stunde auf ungefähr 20 Minuten reduziert werden. Das bedeutet scheint zunächst wenig, hat aber praktisch die Arbeit erleichtert, weil weniger Zeit auf das Ausführen von Pipelines gewartet werden musste. Außerdem hat das häufigere Durchlaufen der Pipelines für mehr Vertrauen in die Tests geführt, die subjektiv ein selbstsicheres und schnelleres Arbeiten befördert hat.

Kapitel 6

Studienaufbau und Ablauf

Um das Produkt zu evaluieren, wurde eine Studie durchgeführt. Bei der Studie testen Probanden das entwickelte Produkt und geben mithilfe von Fragebögen ihr Feedback zurück. In den folgenden Kapiteln wird die Planung und Durchführung der Studie geschildert. Die Ergebnisse werden in Kapitel 7 vorgestellt.

6.1 Planung der Studie

Bevor die Studie durchgeführt wurde, wurde diese zunächst ausführlich geplant. Zuerst musste ein Studienantrag bei der Kommission für Forschungsfolgenabschätzung und Ethik gestellt werden. Der Studienantrag ist in Anhang B zu finden. Der Antrag beinhaltet Informationen von Motivation und Hintergründen bis zur genauen Durchführung.

6.1.1 Probanden

Für die Studie gab es keine spezifische Zielgruppe. Es wurden Personen mit und ohne Erfahrungen in VR und Tischtennis gesucht. Allerdings gibt es einige Voraussetzungen, die erfüllt werden mussten, um an der Studie teilzunehmen.

- Die Probanden müssen das 18. Lebensjahr vollendet haben
- Die Probanden müssen Rechtshänder sein
- Deutschkenntnisse müssen vorhanden sein
- Die Probanden dürfen keine Augenerkrankungen haben, die nicht mit Sehhilfe korrigiert werden können.

Grund dafür sind rechtliche und gesundheitliche Voraussetzungen, die erfüllt werden müssen. Die Probanden müssen Rechtshänder sein, da die Unterscheidung zwischen Vorhand- und Rückhandschlag nur auf Rechtshänder ausgelegt ist.

Für die Rekrutierung wurde ein Werbetext (siehe E) auf dem schwarzen Brett von Stud.IP veröffentlicht. Zusätzlich wurde Probanden über die Freunde der Projektgruppenmitlieder akquiriert.

Insgesamt haben sich elf Interessierte gemeldet, von denen drei Personen ihre Termine absagen mussten. Die Teilnehmer hatten unterschiedlich viel Erfahrung in Tischtennis vorzuweisen. Es gab ein paar Anfänger, aber auch einige, die schon längere Zeit Tischtennis gespielt haben. Erfahrung in der virtuellen Realität hatten allerdings nur drei von den acht Personen vorzuweisen, wobei nur eine Person regelmäßig eine VR-Brille benutzt.

6.1.2 Auswahl der Fragebögen

Ein wichtiger Punkt für die Evaluation ist die Auswahl der Fragebögen. Da das Ziel der Studie ist, die Nutzbarkeit des Produkts zu testen, wurden Fragebögen ausgesucht, die sich darauf spezialisieren.

Der System Usability Scale (SUS) (siehe D.5) wird verwendet, um die Nutzerfreundlichkeit und Gebrauchstauglichkeit des Produkts zu evaluieren. Mit dem User Experience Questionaire (UEQ) (siehe D.4) wird der unmittelbare Eindruck der Probanden erfasst. Der NASA-TLX (siehe D.3) wird verwendet, um die Belastung der Probanden während des Aufenthalts in VR zu ermitteln. Mithilfe dieser Fragebögen können Probleme beim Spiel, Chatbot usw. entdeckt werden.

Zusätzlich wurden zwei weitere Fragebögen erstellt. Der erste Fragebogen (siehe D.1) wird den Probanden vor dem Spiel gegeben und konzentriert sich auf ihre bisherigen Tischtennisfähigkeiten sowie ihre Erfahrungen mit VR-Tischtennisspielen. Dieser Fragebogen soll Informationen darüber sammeln, wie vertraut die Teilnehmer mit dem Spiel und der virtuellen Umgebung sind.

Der zweite Fragebogen (siehe D.2) wird nach dem Spiel den Probanden gegeben und konzentriert sich auf die Bewertung der Erfahrung mit dem Chatbot. Hierbei steht im Fokus, wie hilfreich der Chatbot dabei war, die Schläge zu erlernen, wie gut die Anweisungen des Chatbots verstanden wurden und wie die Reaktionsfähigkeit des Chatbots empfunden wurde. Ebenfalls wird erfragt, ob die Probanden das Gefühl hatten, dass der Chatbot ihre Tischtennisfähigkeiten verbessert hat, und wie die Interaktion mit dem Chatbot wahrgenommen wurde.

6.2 Durchführung

Die Studie wurde im Zeitraum vom 14.08.2023 bis zum 25.08.2023 im Core IML[1] durchgeführt. Die Probanden bekamen vorab Informationen zum Versuchsablauf und eine Einwilligungserklärung zugeschickt. Falls sie diese zu Beginn der Studie noch nicht gelesen und unterschrieben haben, bekamen sie dafür vor Ort die nötige Zeit. Für die Versuchsleiter gab es ein Skript, welches für eine problemlose Durchführung sorgen soll. Das Skript befindet sich in Anhang C.

Vor Beginn der Studie wurde das Spiel getestet. Dabei wurden alle Funktionen wie Chatbot, Schlagklassifikation, die Kommunikation zwischen Front- und Backend usw. überprüft. Ebenfalls wurde vor jedem Probanden das VR-Headset und die Controller desinfiziert. Sobald ein Proband eingetroffen ist, wurde ihm der Ablauf und wichtige Hinweise beschrieben. Nachdem alles erklärt wurde und der Proband den ersten Fragebogen ausgefüllt hat, wurde das VR-Headset aufgesetzt. Bevor die Probanden angefangen haben Tischtennis zu spielen, lernen sie die Steuerung und haben Zeit

[1] LibraryName-Core IML: `https://www.core-oldenburg.de/`, letzter Aufruf: 29.09.2023

sich an die virtuelle Umgebung zu gewöhnen. Zusätzlich wurden den Probanden alle möglichen Funktionen im Spiel gezeigt. Danach haben die Probanden angefangen, gegen die Platte Tischtennis zu spielen, um sich mit Ball und Schläger vertraut zu machen. Sobald sie sich eingespielt hatten, bekamen die Teilnehmer die Chance, gegen den Bot zu spielen.

Für den kompletten Aufenthalt in der virtuellen Welt wurden 30 Minuten eingeplant. Die Teilnehmer hatten aber auch die Möglichkeit länger zu spielen oder vorzeitig abzubrechen. Ebenfalls hatten sie die Option, eine Pause zu machen. Während der Studie sollte jeder Teilnehmer mindestens einmal den Chatbot benutzen, sodass dieser evaluiert werden kann. Nachdem die Probanden das Experiment beendet haben, bekamen sie die anderen vier Fragebögen zur Beantwortung dieser. Die komplette Durchführung dauerte in der Regel eine knappe Stunde.

Für die Evaluation wurden folgende Daten gespeichert:

- Videoaufnahme des Spiels mithilfe von OBS
- Daten der Schläge
- Liste der fehlerhaften Schläge
- Anzahl der verschiedenen Schlagtypen
- Anzahl der Fehler für die verschiedenen Schlagtypen

Kapitel 7

Ergebnisse der Studie

Die Nutzerstudie wurde mit acht Probanden durchgeführt. Die Projektteilnehmer wichen bei der Durchführung in einer grundlegenden Art vom geplanten Ablauf ab, und zwar wurde nach 30 Minuten Nutzungszeit jedem Probanden freigestellt noch weiter zu spielen. Dies wurde nur in zwei Fällen wahrgenommen. Ebenso konnten die Probanden selbst bestimmen, die zehn-minütige Gewöhnungsphase sowie das Spielen im Trainingsraum frühzeitig zu beenden und gegen den Bot zu spielen. Darüber hinaus wurde es den Probanden überlassen, nach wiederholten Fehlern den Trainingsraum zum Üben zu nutzen. Da der Fokus der Studie auf der Usability des Systems lag, werden diese Abweichungen von den Projektteilnehmern nicht als kritisch angesehen.

Probanden wurden durch eine vierstellige Zahlenkombination codiert. Tabelle 7.1 zeigt die Vorkenntnisse der Probanden, die mit D.1 erhoben wurden, sowie die Ergebnisse der Fragebögen D.5 und D.3.

Während der Studie fanden zwei Änderungen am System statt:

1. Nach den Studiensitzungen der ersten beiden Probanden (5236 und 7341) wurde eine Geräuschunterdrückung für den Chatbot eingeführt, damit dieser Eingaben besser verstehen kann. Mehr dazu wird in Abschnitt 7.7.3 beschrieben.
2. Die letzten beiden Probanden (6392 und 2056) spielten mit einer konsistenteren Schlägerhitbox.

7.1 Ergebnisse des NASA-TLX

Um die Arbeitsbelastung der Probanden zu evaluieren, wurde der bereits erwähnte NASA-TLX Fragebogen verwendet. Dieser Fragebogen zielt auf das Erfassen unterschiedlicher Belastungsformen, wie geistige Anforderungen, körperliche Anforderungen, Frustration, Leistung und Anstrengung, ab. Mithilfe der erfassten Werte lässt sich die übergreifende Belastung berechnen. Die durchschnittliche Belastung der Probanden auf einer Skala von 0 bis 100 liegt bei dem Wert 49,25.

Die Gewichtung der unterschiedlichen Kategorien ist in Abbildung 7.1 zu sehen. Die höchste Belastung liegt mit Hinblick auf Leistung vor. Ein dafür vorliegender Erklärungsansatz ist, dass Probanden Probleme hatten, Punkte zu erzielen und somit mit

Proband	Tischtennis Erfahrung (0 = gar keine, 5 = viel Erfahrung	VR Erfahrung (0 = gar keine, 5 = viel Erfahrung	SUS Score (0 bis 100)	NASA-TLX Score (0 bis 100)
1267	2	1	90	34
1956	2	2	72,5	36
2056	5	1	52,5	58
5236	3	3	97,5	32
6932	5	1	50	56
7341	3	1	65	66
7743	2	1	15	48
9283	3	4	85	64
Durchschnitt	**3,125**	**1,75**	**65,938**	**49,25**

Tabelle 7.1: Ergebnisse der Fragebögen zur Vorerfahrung der Probanden und die gegebenen SUS und NASA-TLX Scores

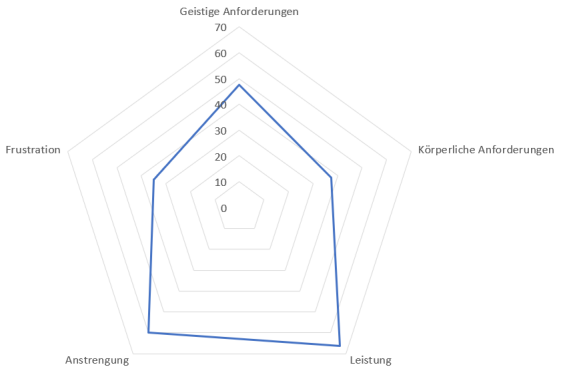


Abbildung 7.1: Gewichtung der Belastung je Kategorie

ihrer Leistung unzufrieden waren. Die Kategorie Frustration hat den niedrigsten Wert. Erklären ließe sich dies damit, dass den Probanden vorab erklärt wurde, dass es sich um einen Prototyp handelt und dieser stellenweise fehlerhaft sein kann.

7.2 Ergebnisse der System Usability Scale

Um die Usability des Tischtennisspiels zu evaluieren, wurde die System Usability Scale verwendet. Diese erfasst die Zufriedenheit des Anwenders während der Benutzung eines Systems. Folgende Ergebnisse wurden durch den Fragebogen ermittelt:

- Overall SUS Score: 66
- Max. SUS Score: 97,5
- Min. SUS Score: 15
- Grade Scale: C

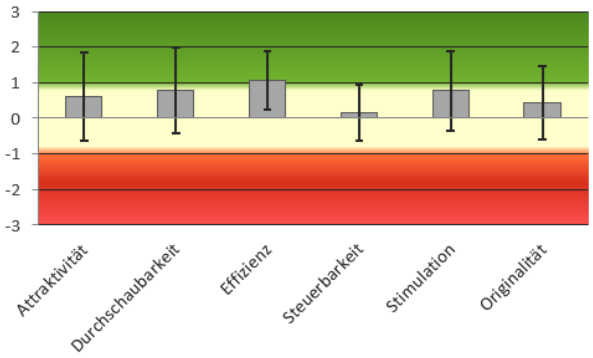


Abbildung 7.2: Ergebnisse von UEQ in den einzelnen Kategorien.

- Adjective Scale: Ok

Der Overall SUS Score des SmartTableTennis-Systems liegt mit einem Wert von 66 nah am Durchschnitt von 68 [Lewis and Sauro, 2018]. Die einzelnen SUS Scores der Probanden lagen zwischen 50 und 90. Allerdings gab es auch einen sehr hohen Wert von 97,5 und einen sehr niedrigen Wert von 15. Ebenfalls ist der SUS-Score bei den zwei Probanden mit den meisten VR-Kenntnissen überdurchschnittlich hoch ausgefallen (97,5 und 85). Erklärung hierfür könnte es sein, dass die Steuerung des Systems für diese Nutzer aufgrund ihrer Vorerfahrungen mit ähnlichen Systemen intuitiver war als für die übrigen Probanden.

7.3 Ergebnisse des User Experience Questionnaire

Der User Experience Questionnaire (UEQ) reflektiert die Erfahrungen der Probanden in unterschiedlichen Kategorien.

- Attraktivität: Gesamteindruck des Produkts
- Durchschaubarkeit: Ist es einfach, sich mit dem Produkt vertraut zu machen?
- Effizienz: Können die Benutzer ihre Aufgaben ohne unnötigen Aufwand lösen?
- Steuerbarkeit: Hat der Benutzer das Gefühl, die Kontrolle über die Interaktion zu haben
- Stimulation: Ist die Nutzung des Produkts spannend und motivierend?
- Originalität: Ist das Produkt innovativ und kreativ?

Die Abbildung 7.2 stellt die Ergebnisse des Fragebogens dar. Positive Erfahrungen hatten die Probanden insbesondere in den Kategorien Durchschaubarkeit und Effizienz. Da es sich bei dem Produkt um einen Prototyp handelt, wurden die Funktionen aufs Notwendigste reduziert. Zusätzlich hatten alle Teilnehmer bereits ein wenig Erfahrung in Tischtennis, sodass sie sich schnell mit dem Spiel in VR vertraut machen konnten. Negative Erfahrungen hatten die Probanden im Bereich Steuerbarkeit. Der Grund dafür liegt bei den verschiedenen Bugs, die während dem Aufenthalt aufgetreten sind. Dadurch könnten die Benutzer das Gefühl bekommen, die Kontrolle über die Interaktionen zu verlieren.

7.4 Ergebnisse des Chatbot-Fragebogens

In Abbildung 7.3 befinden sich die Ergebnisse des Fragebogens über den Chatbot. Die Ergebnisse sind in die folgenden Kategorien unterteilt:

- Hilfreich: Wie hilfreich war der Chatbot beim Erlernen des Spiels?
- Verständlichkeit: Wie gut konnten die Probanden den Chatbot verstehen?
- Reaktionsfähigkeit: Wie schnell hat der Chatbot auf die Probanden reagiert?
- Bedienbarkeit: Wie leicht war es, den Chatbot zu bedienen?

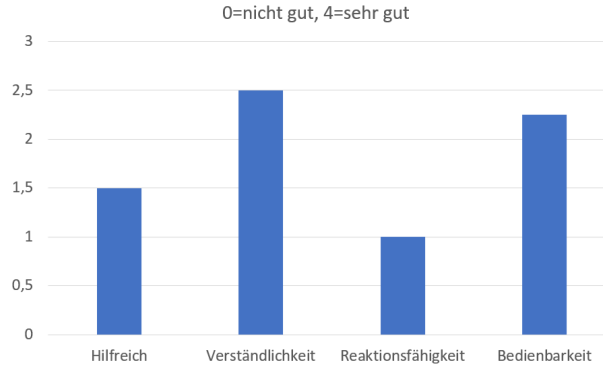


Abbildung 7.3: Ergebnisse des Fragebogens über den Chatbot

Positive Erfahrungen hatten die Teilnehmer bei der Verständlichkeit und Bedienbarkeit des Chatbots zu berichten. Mithilfe der TTS-Funktionen konnten die Probanden ohne Probleme die Tutorials des Chatbots verstehen. Da zum Starten der Tutorials nur ein Knopf gedrückt werden muss, war auch die Bedienbarkeit des Chatbots kein Problem. Negative Erfahrungen berichteten die Teilnehmer bei der Unterstützung und der Reaktionsfähigkeit des Chatbots. Da alle Teilnehmer bereits Tischtenniserfahrung hatten und der Chatbot nur die Grundlagen erklären kann, war der Chatbot für viele Teilnehmer nicht notwendig. Hinzu kommt, dass der Chatbot sehr lange gebraucht hat, um eine Frage zu beantworten. Für weitere Ergebnisse und Diskussion siehe Abschnitt 7.7.3 und 7.7.6.

7.5 Beobachtungen und Auswertung der Klassifikation

In diesem Abschnitt werden die Beobachtungen der Klassifikation aufgezeigt und evaluiert. Um die Genauigkeit der Klassifikation des Netzes außerhalb des Trainings zu betrachten, müssten folgende Schritte durchgeführt werden: In dem Videomaterial müsste jeder einzelne Schlag zu den entsprechenden Daten und dessen Klassifikation zugeordnet werden. Dafür hätten die Zeitstempel der Daten mit den Zeitstempeln des Videos abgeglichen werden müssen, um die Daten zu den Videosequenzen zuordnen zu können. Dabei hätte beachtet werden müssen, dass die Zeitstempel des Videos nicht mit den Zeitstempeln der Daten übereinstimmen und es hätte die Differenz dieser bestimmt werden müssen. Danach hätte jede einzelne Videosequenz der jeweils

beobachteten Schläge betrachtet werden müssen und manuell als Vorhand, Rückhand oder Aufschlag eingestuft werden müssen. Die jeweilige Klassifikation durch das Netz hätte mit der manuellen Klassifikation, die bei vor allem schnellen Ballwechseln nicht korrekt sein könnte, verglichen werden müssen. Dadurch hätte die Genauigkeit des Netzes ermittelt werden können. Insgesamt wurden 1046 Schläge aufgenommen und somit hätten 1046 Videosequenzen evaluiert, zu der jeweiligen Klassifikation des Netzes zugeordnet und verglichen werden müssen. Aufgrund von fehlenden zeitlichen Ressourcen und der Anfälligkeit dieses Prozesses auf menschliches Versagen bei der Zuordnung von aufgenommenem Schlag zu Videosequenz werden im Rahmen dieser Auswertung keine Aussagen über die Genauigkeit der Klassifikation durch das Netz im echten Spiel getroffen. Die Genauigkeit des Netzes in der Aufnahmeumgebung wurde bereits in Abschnitt 5.4.2 erläutert. Beachtet werden muss, dass diese Genauigkeit nicht auf die Klassifikation während des Spieles exakt übertragen werden kann, da die Daten aus der Aufnahmeumgebung mit einem Bot, der immer gleiche Aufschläge ausgeführt hat, und nicht einem Gegenspieler, der zwischen den Schlägen variiert hat, aufgenommen worden sind. Bei dem Bot mit denselben Aufschlägen ist dem Spieler im Vorhinein bewusst, wo der Ball auftreffen wird und er kann sich anders auf diesen vorbereiten. Das kann sich auf die Schlägerhaltung während des Schlages auswirken.

Stattdessen wird sich auf die Auswertung der Wahrscheinlichkeiten, mit welcher die jeweilige Schlagart klassifiziert wurde, fokussiert. Diese wird im Folgenden als Sicherheit der einzelnen Klassifikation betrachtet. In Abbildung 7.4 ist ein Histogramm über die Wahrscheinlichkeit, mit welcher die jeweilige Schlagklasse ausgegeben wurde, dargestellt. Die Abbildung zeigt, dass die meisten Schläge zwischen 98% - 100% Sicherheit klassifiziert wurden. Dabei wurden die Vorhandschläge zu durchschnittlich 92,2%, die Aufschläge mit 96,9% und die Rückhandschläge mit 96,7% Sicherheit erkannt.

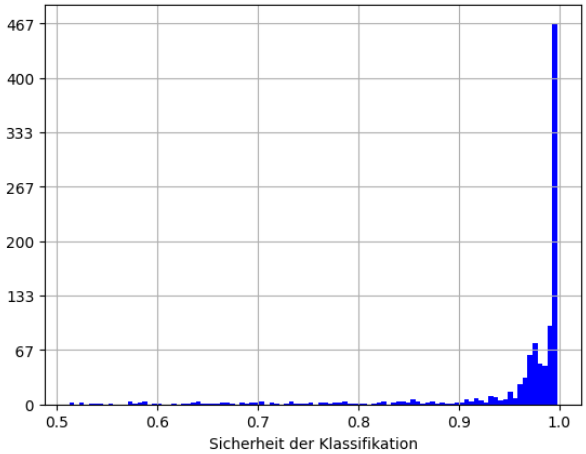


Abbildung 7.4: Verteilung über die Sicherheit der Klassifikation der Schläge

In Abbildung 7.5 ist die durchschnittliche Sicherheit der Klassifikation nach Teilnehmer und Schlag dargestellt. Die Abbildung zeigt, dass die durchschnittliche Sicherheit bei dem Großteil der Schläge und Teilnehmer bei über 90% liegt. Auffällig ist die geringe Sicherheit bei der Klassifikation der Vorhand von Teilnehmer 1956. Auch die Sicherheit bei der Vorhanderkennung vom Teilnehmer 6932 und der Rückhan-

derkennung von Teilnehmer 7743 ist im Durchschnitt geringer als die Sicherheit bei der Erkennung der weiteren Schlagarten pro Teilnehmer. Möglicherweise waren die Rückschläge jeweils nicht korrekt ausgeführt, was die geringe durchschnittliche Erkennung erklären könnte. Bestätigt werden kann dieses anhand der Ergebnisse der Fragebögen hinsichtlich Tischtennis und VR-Erfahrung. Die Probanden 7743 und 1956 haben sowohl wenig Tischtennis-Erfahrung als auch VR-Erfahrung angegeben. Im Abschnitt 7.6 werden die Schläge in Verbindung mit der Nutzerperformance näher analysiert.

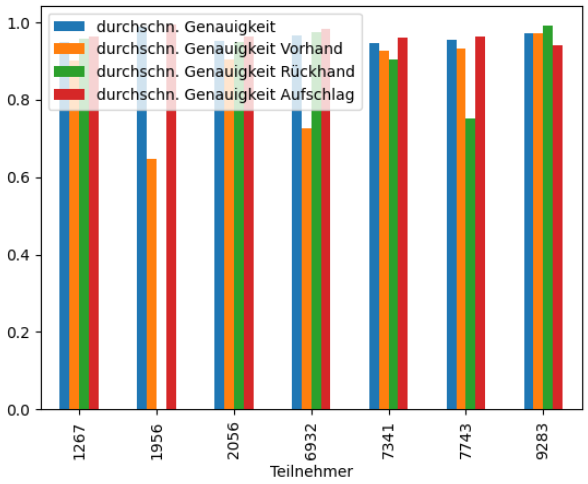


Abbildung 7.5: Durchschnittliche Genauigkeit der klassifizierten Schläge pro Teilnehmer

In 7.6 sind die Schläge, die mit einer Genauigkeit von unter 70% erkannt worden, pro Teilnehmer dargestellt. Auffällig ist der Teilnehmer 7341, bei dem die meisten Schläge nur mit einer Sicherheit von unter 70% erkannt werden konnten. Im Videomaterial konnte bei diesem Teilnehmer beobachtet werden, dass der Teilnehmer Probleme hatte, die Schläge korrekt auszuführen, was die vermehrten Unsicherheiten des Netzes in der Klassifikation erklären könnte. Ebenfalls zeigt die Abbildung, dass ein Großteil der Schläge, die mit einer Sicherheit von unter 70% klassifiziert wurden, als Vorhandschläge erkannt wurden.

Insgesamt lag die Sicherheit der Klassifikation in jeder Schlagart im Durchschnitt bei über 90%. Im Trainingsverlauf (Abbildung 5.23) konnten die realen Daten im Durchschnitt mit über 90% Genauigkeit klassifiziert werden. Da im Training auch synthetische Daten im Einsatz waren, kann insgesamt geschlossen werden, dass diese im Training der Netze im betrachteten Fall genutzt werden können.

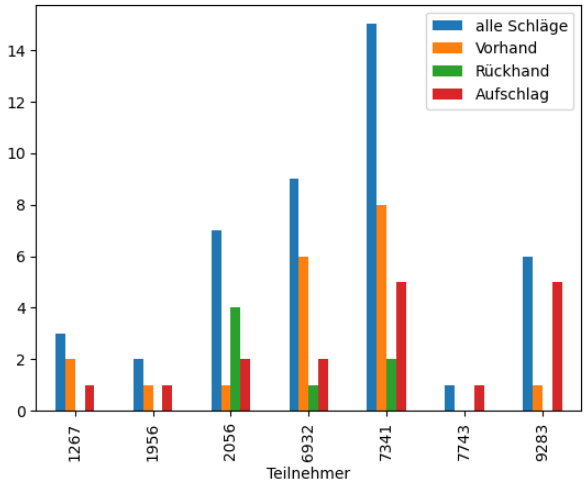


Abbildung 7.6: Klassifizierte Schläge mit einer Sicherheit von unter 70%

7.6 Beobachtungen und Auswertung der Nutzerperformance, sowie Kontextualisierung durch Fragebogenergebnisse

Während der Benutzerstudie wurden die klassifizierten Schläge jedes Probanden aufgezeichnet und gespeichert. Hierfür wurde jeder Schlag bestehend aus jeweils 150 Datenpunkten in einer csv-Datei gespeichert und anschließend innerhalb eines *Jupyter-Notebooks* eingelesen und in einen *pandas.DataFrame*[1] geladen. Innerhalb dieses Dataframes wurden die jeweiligen Schläge den Probanden zugeordnet sowie die Klassifikationsergebnisse und deren jeweilige vom Netz ausgegebene Wahrscheinlichkeit der Klasse diesen hinzugefügt. Mithilfe dieses vollständigen Dataframes wurden die Daten gesichtet und anhand verschiedener Metriken geplottet. Die daraus entstandenen Plots sowie Beobachtungen sind in diesem Abschnitt dargelegt.

Bei erster Sichtung der aufgezeichneten Daten ergab sich, dass die Daten eines der acht Probanden (5236) nicht verwertbar sind. Dies ist bedauerlich, da besagter Proband sowohl den höchsten SUS Score als auch den niedrigsten NASA-TLX Score abgab.

Die Auswertung dieser Daten mit Hinblick auf das Verhalten der Spieler erfolgt unter der Annahme, dass die Klassifikationen durch das neuronale Netz korrekt sind. Gegebenenfalls werden die mit OBS aufgezeichneten Videoaufnahmen konsultiert, um Auffälligkeiten zu überprüfen.

Abbildung 7.7 zeigt die Anzahl und Klasse der Schläge und Fehler nach Proband. Bei Betrachtung der Abbildung 7.7a ist zunächst auffällig, dass, mit Ausnahme von Proband 9283, jeder Proband am meisten Aufschläge gespielt hat. Dies kann als Indikator dafür dienen, dass Probanden Schwierigkeiten damit hatten, erfolgreiche Ballwechsel

[1] Pandas-Dokumentation pandas.Dataframe: `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html`, letzter Aufruf: 14.09.2023

mit dem Bot zu spielen. Dies lässt sich durch Betrachtung der Videos der Probanden ebenfalls bestätigen.

Eine weitere Anomalie stellt Proband 1956 dar. Dieser spielte keine Rückhandschläge und nur wenige Vorhandschläge, welche durch das neuronale Netz als solche zu erkennen waren. Darüber hinaus konnten die wenigen Vorhandschläge nur unsicher erkannt werden (siehe Abbildung 7.6). Ein Erklärungsansatz hierfür ist, dass der Proband Probleme bei der Bedienung des Systems und besondere Probleme beim Spielen von Ballwechseln mit dem Bot hatte. Diese Erklärung würde im Kontrast mit den Fragebogen-Ergebnissen des Probanden stehen, da dieser sowohl eine überdurchschnittlich hohe System-Usability (75,5), als auch eine unterdurchschnittliche Belastung (46) abgab. Nähere Betrachtung der Videoaufnahmen zeigt, dass der Proband versuchte, mit gezielten Aufschlägen Punkte gegen den Bot zu erzielen, indem auf Teile der Platte gezielt wurde, welche der Bot schwer erreichen kann. Dies kann die Ergebnisse der Fragebögen in diesem Fall erklären: da nicht ganzheitlich mit dem System interagiert wurde, nahm der Proband das System als einfach bzw. gut nutzbar war und da nur eine repetitive Bewegung ausgeführt wurde, welche zu Erfolgen seitens des Spielers führte, kam eine verhältnismäßig geringe Belastung zustande.

Die Beobachtungen von Proband 1956 liefern zwei wertvolle Einsichten, die es näher zu betrachten gilt:

1. Das Spielverhalten des Bots lässt sich ausnutzen und motiviert somit weder variables Spiel noch Bewegung
2. Vorerfahrungen mit Tischtennis führen nicht zu einem besseren und erfolgreicheren Spielerlebnis

Diese Einsichten haben Implikationen für die Bezeichnung des Systems als SG, denn, wenn SG Sachverhalte des Tischtennis vermitteln soll, sollten Erfahrungen mit Tischtennis auch zu besserer Performance im SG führen. Diese Einsichten gilt es zu validieren oder zu widerlegen, um Aussagen über die Eignung des Begriffes *Serious Game* zur Beschreibung dieses Systems zu treffen.

Die einzigen Probanden, welche angaben, viel Erfahrung mit Tischtennis zu haben, waren 6932 und 2056. Ebenso gab nur Proband 9283 an, bereits vermehrt Erfahrung mit VR-Systemen gemacht zu haben. Erwähnenswert ist hierbei, dass insbesondere diese drei Probanden verschiedene Schlagarten nutzten, jedoch keiner mehr erfolgreiche Schläge als Fehler hatte (siehe 7.7a und 7.7b). Im Gegenteil haben Probanden (1267, 1956 und 7743), welche weniger variiert gespielt und wenig Erfahrung mit sowohl Tischtennis als auch VR-Technologie haben, mehr erfolgreiche Schläge als Fehler erzielt. Hierbei gilt es zu erwähnen, dass die Anzahl erfolgreicher Schläge eine unpräzise Metrik ist, um die Performance eines Spielers zu erheben. Hier wäre eine Betrachtung der Anzahl erfolgreicher Ballwechsel für die System-Usability aussagekräftiger gewesen, jedoch konnte im Rahmen der Projektausarbeitung keine Lösung gefunden werden, um diese konsistent und präzise zu erfassen.

Eine Betrachtung der Abbildung 7.8a zeigt, dass Probanden sich dichter an der Platte befanden als der Bot, um den Ball zu spielen, und dass die Platte nicht in ihrer gesamten Breite ausgenutzt wurde. Hieraus lässt sich schließen, dass das Verhalten des Bots keine Bewegung entlang der Platte seitens der Spieler motiviert, was die Simulationseigenschaften des Systems weiter vermindert. Zudem geht aus den Heatmaps hervor, dass der Großteil der Aufschläge als Vorhand gespielt wurden, da die Formen der

Heatmaps einander stark ähneln. Dies ist erwähnenswert, da bei Betrachtung der Videoaufnahmen derjenigen Probanden, die erhöhte Tischtennis-Kenntnisse angaben, Aufschläge auch als Rückhand spielten.

Die Betrachtung der Videos ergab, dass das erfolgreiche Spielen von Ballwechseln nur mit der verbesserten Hitbox des Schlägers erfolgte. Dies ist erwähnenswert, da die Probanden 6932 und 2056, welche mit dieser spielten, sowohl eine stark unterdurchschnittliche System-Usability (50 und 52,5) als auch eine überdurchschnittliche Belastung (56 und 58) berichteten. Für diesen Zusammenhang gibt es verschiedene Interpretationsansätze, welche einander nicht ausschließen, sich jedoch mit den erfassten Daten und Videos nicht näher validieren lassen, darunter:

1. Durch vereinzelte erfolgreiche Ballwechsel mit dem Bot fallen nicht erfolgreiche Ballwechsel zunehmend auf und lassen auf eine mangelnde System-Usability schließen, daher der unterdurchschnittliche SUS-Score.
2. Vereinzelte erfolgreiche Ballwechsel erhöhen die Erwartungen an weitere, versuchte Ballwechsel. Schlagen diese fehl, führt dies zu Frustration seitens des Spielers und trägt somit zu einer höheren Belastung bei.
3. Häufiger auftretende Inkonsistenzen bei Systemfunktionalitäten erzeugen bei Probanden vermehrt den Eindruck, dass es sich beim Projekt um ein Produkt in einem frühen Entwicklungsstadium handelt. Dies führt zu mehr Kulanz und somit weniger negativem Feedback bei der Bewertung des Systems.

7.7 Weitere Ergebnisse

Im Folgenden werden die Bugs beschrieben, die im Verlauf der Studie auftraten. Außerdem werden hier die Bugs beschrieben, die schon während der Studie behoben worden sind. Darüber hinaus werden Vermutungen aufgestellt, welchen Einfluss die Behebungsmaßnahmen potenziell auf die Ergebnisse der Studie haben.

Bei Beginn der Studie lag ein Systemfehler vor, welcher dazu führte, dass keine Daten vom Frontend an das Backend gesendet wurden. Vermutet wird, dass ein fehlerhaft gelöster Merge-Konflikt und unvollständiges Testen des Gesamtsystems zwischen dem letzten Playtest und Studienbeginn diesen erzeugt haben. Der genaue Ursprung des Fehlers ist den Projektteilnehmern nicht ersichtlich, weshalb eine ältere Version des Frontends für die ersten beiden Probanden verwendet wurde. Diese Version enthielt verschiedene Fehler, welche für die Studie jedoch weniger Priorität als eine funktionale Client-Server-Kommunikation hatten. Diese Fehler werden im Folgenden als solche gekennzeichnet.

7.7.1 Bugs im Training und Spiel

Der Button, um ein neues Spiel zu starten, ist fehlerhaft: Das Scoreboard wird nicht aktualisiert und der Ball ist nicht im Holster zu finden, während das Scoreboard unbeabsichtigterweise im Trainingsraum zu sehen ist. Hierbei handelt es sich um einen Fehler, der vor Beginn der Nutzerstudie bereits behoben wurde, dessen Fix jedoch noch nicht auf der verwendeten Version des Frontends implementiert war.

7.7.2 Ungewollte Verhaltensmuster und Fehler des KI-Gegenspielers

Die Projektgruppe konnte im Rahmen der Nutzerstudie beobachten, dass der Bot unter Umständen den Ball nicht korrekt zurückspielt beziehungsweise dies gar nicht versucht. Ein ausbleibender Versuch ist als solcher zu erkennen, da sich der Bot in diesem Fall nicht in Richtung des Balls bewegt. Dies führt dazu, dass das Verhalten des Bots ausgenutzt werden kann, um Punkte zu erzielen, zum Beispiel, indem der Ball über einen der Seitenränder der gegnerischen Plattenhälfte gespielt wird.

Wird der Schläger mithilfe der Regler rotiert, rotiert der Schläger vom Bot in gleicher Art und Weise mit. Auch dies könnte durch Spieler ausgenutzt werden, um die Fähigkeit des Bots das Spiel zu spielen einzuschränken. Die Auswirkungen, die dieser Fehler haben kann, konnten im Rahmen der Nutzerstudie nicht ermittelt werden und wurden aus Zeitgründen nach der Studie nicht näher untersucht.

7.7.3 Ergebnisse und Bugs des Chatbots

Zu Beginn der Studie hat ein Proband den Chatbot gefragt, wie er den Ball trifft. Auf diese Frage hat er keine Antwort bekommen, auch nicht, dass der Chatbot ihn nicht verstanden hat. Außerdem wird die Verabschiedung nicht korrekt verstanden, dadurch lässt sich der Chatbot nicht sprachlich beenden. Stattdessen läuft die Sprachaufnahme weiter und der Chatbot antwortet auf das, was er verstanden hat.

Diese Bugs können Probleme bei der Sprachaufnahme per Mikrofon auf akustische Gegebenheiten im Core IML, Rauschen und Hintergrundgeräusche zurückzuführen sein. Denn zum Zeitpunkt des Studienbeginns wurde die Sprachaufnahme per Mikrofon durchgeführt, ohne die Hintergrundgeräusche zu behandeln. In einem entsprechenden Bugfix ist eine Funktion der Bibliothek Speech Recognition[48] eingebunden worden, die Umgebungsgeräusche händelt. Außerdem wurde der programmatische Ablauf der Verarbeitung der Nutzernachricht leicht angepasst: Um die Antwortzeiten des Chatbots zu minimieren und die Konversation flüssiger zu gestalten, wurde die Nachricht nur dann an den Rasa[25]-Server gesendet, wenn diese nicht leer ist. Bisher entstand dadurch eine Wartezeit, auch wenn der Nutzer nichts gesagt hat.

Dieser Fix wurde nach dem ersten Studientag implementiert. Die Studienergebnisse werden dadurch vermutlich nur bezüglich des Chatbots beeinflusst. Das Ziel ist, dass die Probanden so überhaupt mit dem Chatbot kommunizieren können. Wenn der Fix erfolgreich war, sollte sich eine Erhöhung der Nutzerfreundlichkeit des Chatbots feststellen lassen.

Außerdem bereitete die Fixierung des Mikrofons Schwierigkeiten. Bei einem kleinen Teil der Probanden wurde das Mikrofon so platziert, dass es beweglich war und hin und her schwingen konnte. Dies hat die Sprachaufnahme erschwert. Dieser Aspekt der Fixierung ist aber einer, der während der Studie bzw. für jeden Proband unterschiedlich sein kann. Daher sollte das Ziel sein, für jeden Probanden eine möglichst ähnlich gute Fixierung des Mikrofons zu erreichen und so vergleichbare Ergebnisse bezüglich der Kommunikation mit dem Chatbot zu erhalten. Wenn dies nicht der Fall ist, sollte bei der Auswertung der Studienergebnisse die Qualität der Mikrofonfixierung berücksichtigt werden.

Weiterhin sind die Antwortzeiten des Chatbots sehr lang. Mögliche Ursachen dafür

können die verwendeten Toolkits Rasa[25], Speech Recognition[48] und gTTS[32] sein. Einerseits nimmt das Abspielen der erstellten Audiodatei Zeit in Anspruch und andererseits dauert es lange, bis das Abspielen der Audiodatei beginnt. Dies ist ein Problem hinsichtlich der Nutzerfreundlichkeit, weil der Nutzer einen Gesprächsverlauf erwarten würde, der zwischen zwei Menschen bestehen würde. Dabei wäre es unüblich, wenn ein Gesprächspartner erst stark verzögert antwortet.

Außerdem ist der implementierte Chatbot nur auf ein geschlossenes Wissensgebiet bzw. vordefinierte Fragen trainiert. So besteht die Gefahr, dass einige Fragen von Probanden in der Studie nicht beantwortet werden können, wenn diese nicht von den Trainingsdaten abgedeckt wurden. Sinn eines solchen Chatbots in unserer Anwendung wäre ein Assistent, der jegliche Fragen beantworten kann. Dies ist in der implementierten Variante nicht möglich.

7.7.4 Bugs in der Klassifizierung und der Fehlerevaluierung

Für die Klassifizierung wird das json-Objekt des Schlages an das Backend gesendet. Dieser Prozess soll nur im Spiel durchgeführt werden. Jedoch werden im Trainingsraum ebenfalls json-Objekte an das Backend gesendet, das entsprechende json-Objekt ist aber leer. In diesem Fall könnte das Senden komplett eingestellt werden.

7.7.5 Weitere Erkenntnisse aus der Studie

Bezüglich der Fehlerevaluierung wurde erkannt, dass der Fehlerschwellwert, der initial auf fünf gesetzt ist, zu niedrig angesetzt wurde. Hier bietet sich vor allem beim Aufschlag an, diesen auf 15 hochzusetzen.

7.7.6 Diskussion des Chatbots

Hinsichtlich des eingeschränkten Bereichs an Fragen, die der Chatbot beantworten kann, muss die Verwendung des Frameworks Rasa[25] hinterfragt werden. Entweder müssen Trainingsdaten definiert werden, die möglichst viele Nutzerfragen abdecken, oder es müsste ein anderes Framework für den Chatbot verwendet werden. Trainingsdaten für alle denkbaren Fragen zu definieren ist aber nicht möglich, da nicht vorhersehbar ist, welche Fragen die Nutzer stellen könnten. Daher sollte rückblickend die Wahl des Frameworks für den Chatbot hinterfragt werden. Dementsprechend könnte die Nutzung des ChatGPT Sprachmodells sinnvoller gewesen sein, da dieses Modell, wie in Kapitel 3.2.4 beschrieben, schon auf einer großen Menge an Daten trainiert wurde. Mittlerweile kann es zusätzlich auf eigene Daten trainiert werden, wobei aber das Feintuning des ChatGPT-3.5 turbo Sprachmodells noch nicht möglich ist.[27][26]Mit dem Modell von ChatGPT könnte das Problem gelöst werden, auch auf allgemeine Nutzerfragen eingehen zu können.

Bezüglich der langen Antwortzeiten des Chatbots sind während der Implementierung einige Beobachtungen festgestellt worden:

1. Geringe Antwortzeiten bei kommandozeilenbasiertem Chatbot, bei dem die Nutzereingabe ausschließlich über die Kommandozeile geschieht
2. Langsamere Antwortzeiten als beim komplett textbasierten Chatbot (1.) für den kommandozeilenbasierten Chatbot, aber mit sprachlicher Eingabe des Nutzers,

die zuerst in Text umgewandelt werden muss, bevor sie an den Rasa[25]-Server geschickt wird

3. Längste Antwortzeiten des Voicebots im Vergleich zu den Chatbotvarianten 1. und 2.; hier wird sowohl die sprachliche Nutzereingabe verwendet als auch die Chatbotantwort sprachlich ausgegeben

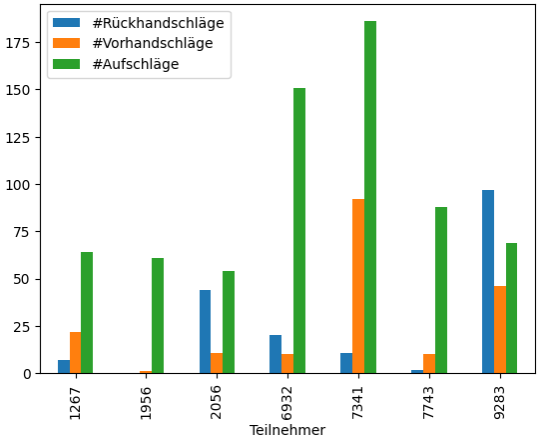
Basierend auf den Beobachtungen in der Liste könnte die Bibliothek gTTS[32] den größten Einfluss auf die Länge der Antwortzeiten haben. Denn die meisten Chatbotäußerungen, die ausgegeben werden, sind sehr lang. Daher braucht sowohl die Erstellung der Audiodateien als auch das Abspielen viel Zeit.

Außerdem kann die Aufnahme der sprachlichen Eingabe des Nutzers Zeit kosten. Aus den Tests wurde deutlich, dass die Mikrofonaufnahme erst dann stoppt, wenn eine Zeit lang keine Sprache aufgenommen werden konnte. Ob es eine Konstante gibt, die angibt, wie lange die Sprachaufnahme laufen soll, ist nicht bekannt. Die Umwandlung der Sprache in Text schien schnell zu funktionieren, vor allem wenn die Nutzereingaben kurz sind.

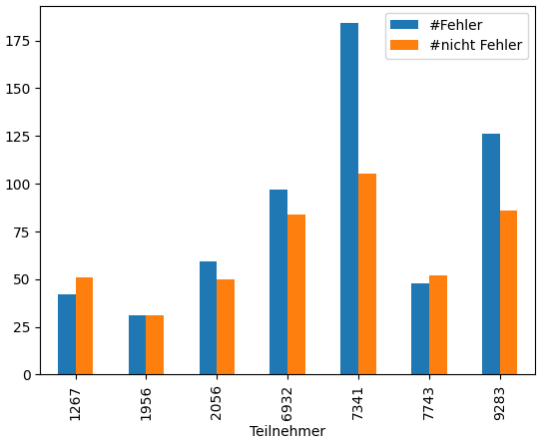
Letztendlich lässt sich festhalten, dass die gewählte Kombination aus den Bibliotheken (Rasa[25], speech recognition[48] und gTTS[32]) nicht für unsere Anwendung geeignet ist, aufgrund der langen Antwortzeiten und der eingeschränkten Antwortmöglichkeiten wegen des beschränkten Wissensbereichs des Chatbots.

Außerdem ist problematisch, dass die Chatbot-Bugs erst während der Studie erkannt worden sind. Grund dafür ist, dass es nur einen Playtest im Studiensetting während des letzten Meilensteintreffens gab. Dort wurden nicht alle Funktionalitäten des Chatbots ausreichend getestet, weshalb der Chatbot zu Beginn der Studie noch nicht ausreichend getestet und anfällig für Fehler war. Während der lokalen Tests zu Hause, in ruhiger Umgebung, mit der eigenen VR-Ausstattung traten diese Fehler nicht auf. Daraus lässt sich schließen, solche Chatbotanwendungen in Zukunft frühzeitig in solch einem Setting zu testen, in dem es genutzt werden soll.

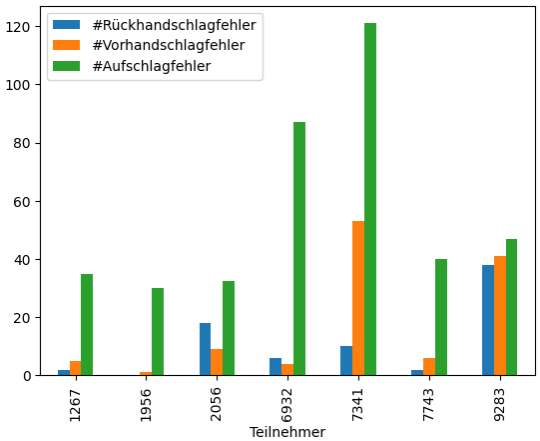
Der implementierte Chatbot ist also nicht nur ein Sprachbefehl, sondern ermöglicht eine Kommunikation mit dem Nutzer. Aber wegen der TTS-Funktionalitäten der Tutorials ist das System bezüglich der Tutorials eines, das einfach Sprachbefehle ausgibt und hier keine weitere Kommunikation mit dem Nutzer ermöglicht.



(a) Anzahl Schläge pro Proband

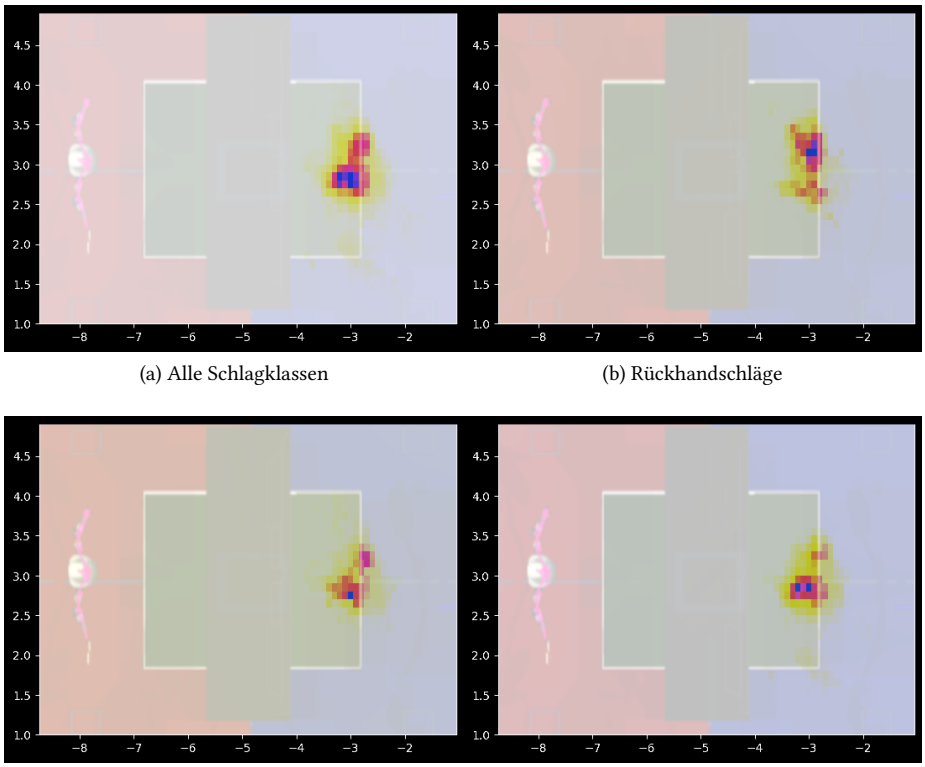


(b) Anzahl der gesamten Fehler pro Proband



(c) Anzahl der artenspezifischen Schläge pro Proband

Abbildung 7.7: Darstellung der verschiedenen Schlag- und Fehlerarten aller Probanden



(a) Alle Schlagklassen

(b) Rückhandschläge

(c) Vorhandschläge

(d) Aufschläge

Abbildung 7.8: Heatmap der Positionsdaten des Schlägers während der aufgezeichneten Schlagbewegungen. Darstellung erfolgt mit den Daten aller Probanden außer 5236. Heatmap wurde über eine Abbildung des Spielfeldes gelegt und passend auf diese skaliert. x- und y-Achse zeigen die Koordinaten innerhalb des Unity-Environments

Kapitel 8

Fazit

Im Folgenden werden die zu Beginn aufgestellten Forschungsfragen noch einmal genannt und anschließend betrachtet, wie die Projektteilnehmer diese im Rahmen des hier Projektes beantworten konnten.

Forschungsfragen:

RQ1 **Wie lässt sich ein VR-Serious-Game für den Sportsektor, hier am Beispiel von Tischtennis, konstruieren?**

RQ2 **Wie lässt sich ein Konversationsinterface bzw. Chatbot in der Nutzung eines VR-Serious-Games für den Sportsektor integrieren und welchen Einfluss hat dies auf die Usability des Systems?**

RQ3 **Wie lassen sich mit Unity ML-Agents konstruierte Agenten in ein höherdimensionales Environment, wie einem VR-Serious-Game integrieren und welche Probleme wirft das ML-Agents Toolkit dabei auf?**

RQ4 **Lassen sich mit synthetischen Daten trainierte Machine-Learning-Komponenten für die Klassifikation von Spielerbewegungen im Rahmen eines VR-Environments nutzen?**

RQ1 Die Benutzerstudie liefert Anlass zur Annahme, dass die Umsetzung eines VR-Serious-Games für Tischtennis möglich ist. Die nahe am Durchschnitt liegende System-Usability (66 gegen einen Durchschnitt von 68) sowie die positiv eingestufte Attraktivität im Rahmen der UEQ-Bewertung dienen hierbei als Indikator. Die Aussagekraft der Ergebnisse dieser Arbeit sind jedoch kritisch zu hinterfragen. Grund hierfür ist die Herangehensweise an die Konstruktion des Systems, mit Hinblick auf Simulationseigenschaften des Spiels und Integration sowie Fähigkeiten des Chatbots. Diese wird von den Projektteilnehmern als suboptimal angesehen, was auf mangelnde Vorkenntnisse mit besagten Technologien zurückzuführen ist. Die Bezeichnung des SmartTableTennis-Systems als SG ist hierbei nicht zutreffend. Zum einen, da ein Wissenstransfer des im Spiel vermittelten Wissens auf das reale Spiel nur bei einer Erfüllung von Simulationseigenschaften gegeben ist, was nach Auswertung der Spielerperformance nicht der Fall ist (siehe Abschnitt 7.6) und zum anderen, da keine Daten durch das System generiert werden, die anderweitig genutzt werden können. Darüber

hinaus gilt es auch den Anwendungsfall des Tischtennis zu hinterfragen. Dieser hat sich im Rahmen der Implementierung aufgrund der Geschwindigkeit und Bewegungsintensität des Realwelt-Sports mehrfach als Herausforderung dargestellt. Hierbei wäre im Rahmen zukünftiger Iterationen des Systems zu untersuchen, ob diese Herausforderungen durch mangelnde Expertise mit den verwendeten Technologien oder gar Hard- und Softwarelimitationen (siehe Abschnitt 5.1.3) handelt.

RQ2 Die Integration des Dialoginterfaces in das VR-Environment war erfolgreich. Limitationen entstanden bei der Umsetzung durch das begrenzte Wissen und Vokabular des Chatbots und im Rahmen der Nutzerstudie traten Hardware- und Setup-bedingte Probleme mit der Spracherkennung auf. Dies führte dazu, dass im Rahmen der Studie diese Funktionalität des Chatbots wenig genutzt wurde und sich somit keine definitive Aussage über den Beitrag der Frage-Antwort-Funktionalität auf die Usability des Systems treffen lässt. Das Feedback der Nutzer im Rahmen der Fragebögen gibt jedoch Grund zur Annahme, dass die hohe Antwortzeit und hardwarebezogene Verständlichkeitsprobleme die größten negativen Faktoren beim Umgang mit dem Dialogsystem waren. Im Rahmen zukünftiger Arbeiten beziehungsweise Iterationen dieses Systems sollten, trotz Erfolg der Implementierung, alternative Dialoginterfaces, wie beispielsweise textbasierte Chatbotausgaben, untersucht werden.

RQ3 Eine Implementierung des KI-Gegenspielers mit dem Unity ML-Agents Toolkit war trotz Vorkenntnissen im Arbeiten mit diesem im Rahmen dieses Projektes nicht erfolgreich. Die Gründe hierfür sind komplex und nicht eindeutig. Eines der Hauptprobleme war im Zusammenhang mit der übrigen Implementierung des Spiels. So war die Instanz, welche den Spielzustand erfasste, nötig für die Verteilung von Rewards jedoch als globaler Singleton implementiert, sodass eine Parallelisierung des Trainings nicht möglich war. Dies führte zu einer Erhöhung der Trainingsdauer und verlangsamte den iterativen Prozess der Anpassungen des Trainings immens. Da jedoch dem Bot erfolgreich Verhaltensweisen antrainiert werden konnten (Bewegung innerhalb einer vorgegebenen Fläche, Halten des Schlägers innerhalb eines bestimmten Bereiches neben dem „Körper“), ist die Möglichkeit der Integration von ML-Agents basierten Agenten in höherdimensionale VR-Environments nicht gänzlich auszuschließen. Über eine grundlegende Eignung des Toolkits für die Anwendung in Unity VR-Environments lässt sich jedoch keine Aussage treffen. Darüber hinaus gilt es den Tischtennis-Use-Case hier erneut zu hinterfragen, da die Geschwindigkeit der Spielobjekte sowie die nötige Präzision der Bewegung einen komplexen Trainingsprozess erfordert.

RQ4 Im Rahmen dieser Arbeit konnte beobachtet werden, dass das Training von Machine-Learning-Komponenten mit synthetischen Daten möglich und erfolgreich war. Hierzu wurde auf Basis von zuvor aufgenommenen Realwelt-Daten und mithilfe zweier Python-Packages ein synthetischer Datensatz generiert, mit welchem die neuronalen Netze anschließend trainiert wurden. Die so trainierten Netze ließen sich sowohl mit einem Validierungsdatensatz testen als auch im Rahmen der Benutzerstudie erfolgreich verwenden. Die Performance des schließlich verwendeten RNN bei der Auswertung des Validierungsdatensatzes, welcher nur aus Realweltdaten bestand, bietet einen quantitativen Anhaltspunkt zugunsten der Anwendbarkeit solcher Komponenten zur Bewegungs-

klassifikation in Unity-VR-Environments. Die Probleme während der Nutzerstudie lassen sich nicht auf eine fehlerhafte Bewegungsklassifikation zurückführen. Ebenso lassen sich keine Aussagen über die Genauigkeit der Klassifikation durch das RNN während der Studie treffen.

Zusammenfassend konnte die **RQ1** teilweise beantwortet werden, indem eine Herangehensweise präsentiert und durch außenstehende Nutzer getestet wurde, jedoch sind nähere Untersuchungen anderer Anwendungsfälle mit umfangreicheren Implementierungen für eine umfassendere Antwort nötig. **RQ2** konnte erfolgreich beantwortet werden, indem auch hier ein Ansatz vorgestellt und evaluiert wurde. Dieser Ansatz jedoch deutet darauf hin, dass alternative und höher entwickelte Umsetzungen zukünftig untersucht werden sollten. Die **RQ3** konnte nicht erfolgreich beantwortet werden, da die Implementierung nicht erfolgreich umgesetzt werden konnte. Aufgrund mangelnder Literatur der Verwendbarkeit und Eignung des ML-Agents Toolkits für VR-Umgebungen lässt sich ebenfalls nicht sagen, ob die erfolglose Implementierung mangelnder Erfahrung oder Ressourcen zuzuschreiben ist, oder einer mangelnden Optimierung des Toolkits selbst. Abschließend konnte im Rahmen der Beantwortung von **RQ4** gezeigt werden, dass für den beschriebenen Anwendungsfall die Verwendung von synthetischen für das Training von Klassifikatoren möglich und effektiv ist. Somit konnte der Großteil der vorgestellten Forschungsfragen durch das Vorgehen dieses Projektes erfolgreich beantwortet werden.

Kapitel 9

Ausblick

Im Rahmen dieses Kapitels werden verschiedene Möglichkeiten, auf die geleistete Arbeit aufzubauen, dargelegt. Hierbei werden besagte Aufbaumöglichkeiten nach Komponenten des Projektes geordnet und organisatorische Verbesserungsmöglichkeiten separat genannt.

9.1 Anpassungen des Projektmanagements

Für zukünftige Projekte, die auf den hier genutzten Ansatz aufbauen, sollten auch Änderungen und Erweiterungen des Projektmanagementkonzeptes erfolgen. Es sollte von Projektbeginn an mit einem iterativen Vorgehensmodell gearbeitet werden. Grund hierfür ist, dass bei einem System auf diversen Komponenten (hier: Tischtennis-Spiel, Machine-Learning Komponenten, KI-Gegenspieler, Chatbot) die flexible Planung und regelmäßige Anforderungserhebung mehr Vorteile mit sich bringt als die Planungssicherheit des Wasserfallmodells - insbesondere dann, wenn noch keine ausgiebigen Vorkenntnisse mit den zu verwendenden Technologien vorliegen. Im Falle geringer Vorkenntnisse sollte ebenfalls eine umfangreichere Einarbeitungsphase durch das Projektmanagement geplant werden. Ebenso sollte in Betracht gezogen werden, technologiespezifisches Wissen innerhalb der Gruppe in Form kleiner Präsentationen oder Leitfäden zu teilen und so vergleichbare Wissensstände innerhalb der Gruppe sicherzustellen und mehr Flexibilität bei der Aufgabenverteilung zu erzielen.

Mit Hinblick auf die Aufgabenorganisation sollte ebenfalls eine umfangreichere Einarbeitung in die zu nutzende Organisationsinfrastruktur erfolgen. Dies soll dazu beitragen, den potenziellen Verlust der Zugänge zu vermeiden und so zu einer aktiveren Nutzung der Infrastruktur zu führen. Ebenfalls nötig ist hierfür jedoch Transparenz der Zugangsvorgänge zur Infrastruktur, sowie gut zugängliche Kommunikationskanäle im Fehlerfall.

Für zukünftige Iterationen dieses Projektes wäre eine Umstrukturierung der Meetings ebenfalls sinnvoll. Beispielsweise sollten mehr Meetings in Präsenz gehalten werden sowie die Moderation wöchentlich wechseln. Ursprünglich wurde von der Projektgruppe der Mehrwert einer wöchentlich wechselnden Moderation nicht erkannt, jedoch zeigte sich im Laufe des Projektes, dass die aktiven Beiträge in Gruppenmeetings

zunehmend weniger wurden. Wöchentlich wechselnde Moderationen und mehr Präsenzmeetings werden von den Projektteilnehmern als eine Möglichkeit, diese zu erhöhen, angesehen.

Während des Projektes stellte sich die Kommunikation und Bereitstellung von Informationen und somit auch die Aufgabenorganisation wiederholt als Herausforderung heraus. Die Formalisierung des Inhaltes von Statusupdates, beziehungsweise Updates zum aktuellen Stand diverser Aufgaben durch Gruppenmitglieder, und eine Formalisierung des Protokollaufbaus und -inhaltes, könnte dies verbessern. Während der Projektdurchführung kam es wiederholt dazu, dass der jeweils aktuelle Stand einer Aufgabe entweder unvollständig und schwer nachvollziehbar wiedergegeben wurde, oder im anderen Extremfall ausschweifend und repetitiv zu vorherigen Wochen formuliert war. Bei den Protokollen ließen sich ebenfalls starke Unterschiede mit Hinblick auf Ausführlichkeit, Präzision und Korrektheit, abhängig davon welches Mitglied dieses verfasste, feststellen. Eine Formalisierung der Updates und des Protokolls könnte diesen Problemen vorbeugen.

Auch die Arbeit mit *Feature-Branches* sollte für zukünftige Iterationen des Projektes überdacht werden. Während aufseiten des Backends diese es ermöglichten, unabhängig von anderen Aufgaben und Features Aufgaben zu bearbeiten und der größte Koordinationsaufwand in der korrekten Reihenfolge abzuschließender Merges bestand, waren die Probleme im Frontend gravierender. Insbesondere bei umfangreicheren Frontend-Projekten und sollten wenige Vorkenntnisse im Umgang mit Unitys Git-Integration vorliegen, sollte vom Arbeiten mit *Feature Branches* für Unity-VR-Projekte abgesehen werden.

9.2 Anpassung der Studie

Mithilfe der Studie konnten einige Fehler und Verbesserungsmöglichkeiten des entwickelten Tischtennisspiels festgestellt werden. Dennoch gab es einige Probleme bei der Antragstellung und Planung der Studie, die für zukünftige Studien berücksichtigt werden sollten.

Der Antrag zur Studie wurde von der Kommission für Forschungsfolgenabschätzung und Ethik der Universität Oldenburg genehmigt. Dieser Antrag beinhaltet viele Punkte zur Durchführung, Datenschutzerklärung, Probandeninformation usw., die detailreich beschrieben wurden. Dadurch musste viel Zeit in die Recherche und das Verfassen des Antrags investiert werden. Zusätzlich wurde lange auf eine Antwort der Kommission gewartet, sodass der Antrag erst nach der Studie genehmigt wurde. Eine Alternative ist, die Studie stattdessen über das DFKI bewilligen zu lassen. Der Studienantrag vom DFKI ist kürzer als der Antrag von der Universität Oldenburg und die Einwilligungserklärung für die Probanden lässt sich dabei mit einem Generator schnell erstellen. So könnte mehr Aufwand in andere Bereiche des Projekts investiert werden.

Ein weiteres Problem war, dass während der Studie noch einige Fehler behoben werden mussten, sodass am Ende nur zwei Probanden das Spiel mit den besten Voraussetzungen getestet haben. Grund dafür war, dass der finale Test erst einige Tage vor dem Studienbeginn stattgefunden hat. Um dieses Problem zukünftig zu vermeiden, sollte der finale Test mindestens eine Woche früher stattfinden. Dadurch lassen sich kleinere und größere Bugs noch vor der Studie beheben. Die Zeit kann ebenfalls verwendet

werden, um komplexere Merge-Konflikte zu lösen.

9.3 Chatbot

Der Chatbot bietet einige Möglichkeiten zur Weiterentwicklung:
Weitere Chatbotäußerungen zu bisher nicht erkannten Fehlern könnten in den Trainingsdaten definiert werden. Dabei handelt es sich um Fehler bezüglich des Zeitpunkts des Balltreffpunkts oder der Länge des Schlages. Voraussetzung dafür ist, dass die Fehlererkennung auf diese Fehler erweitert wird.

Außerdem kann in Zukunft eine Custom Action ans Frontend gesendet werden, wenn der Nutzer den Trainingsraum betreten und ein Tutorial basierend auf seinen Fehlern wiederholen möchte. Grundlegend ist dies im Backend im entsprechenden *Feature Branch* schon implementiert. Dies würde ein automatisiertes Betreten des Trainingsraums ermöglichen. So könnte eine Konversation mit dem Nutzer geführt werden, anstatt ihm nur auf TTS-Basis den Vorschlag zu präsentieren, den Trainingsraum zu betreten. Dies könnte die Nutzerfreundlichkeit erhöhen.

Außerdem könnten die Chatbotfunktionen zum Trainieren des Sprachmodells, zum Starten des Rasa[25]-Servers und zum Ausführen der Rasa[25]-Tests auch für das Betriebssystem macOS umsetzt werden. Die Funktionen starten einen Subprozess, der die entsprechenden Terminal-Befehle ausführt, und sind nur für Linux- und Windowssysteme definiert. Um die Chatbotfunktionalitäten und damit auch das Backend in Zukunft auch in macOS ausführbar zu machen, könnten diese Funktionen um den entsprechenden Terminalbefehl erweitert werden.

Zuletzt kann die Umsetzung des Chatbots grundsätzlich hinterfragt werden. Ziel des Chatbots war dem Nutzer eine Assistenz zu bieten, die ihm bei Fragen weiterhilft oder erkannte Probleme im Spiel benennt. Um diese Assistenzfunktion zu realisieren, wäre es auch denkbar, visuelle Hinweise in Pop-up-Fenstern in der virtuellen Umgebung zu präsentieren. Dabei können auch weitere Informationen geliefert werden, beispielsweise wie viele Fehler einer Schlagart erkannt worden sind. Mithilfe einer solchen Assistenzfunktion wären die Interventionen nicht so invasierend, wie die Nutzer sie bezüglich des Chatbots in der Studie empfunden haben.

9.4 Klassifikation

In diesem Abschnitt werden Möglichkeiten aufgezeigt, wie die Klassifikation weiterentwickelt werden kann. Bei der Auswertung der Klassifizierung wurde die Genauigkeit des Netzes bei der Klassifizierung der Schläge im Spiel nicht evaluiert. Dies stellt einen möglichen Ansatzpunkt für künftige Untersuchungen dar. Dafür sollte ein Verfahren entwickelt werden, welches eine gleiche Zeitsynchronisation von Video und Daten bietet. Anhand der Daten könnten der Anfangs- und Endzeitpunkt eines Schlages ermittelt werden. Mittels der Zeitpunkte könnten die einzelnen Videosequenzen geschnitten und den entsprechenden Daten sowie dessen Klassifizierung zugeordnet werden. Danach müsste manuell jede Videosequenz betrachtet und mit den Ergebnissen der Klassifizierung verglichen werden. Dadurch könnte die Genauigkeit des Netzes bei der Klassifizierung der Schläge während eines Spieles ermittelt werden.

Ebenfalls kann die Klassifikation an sich erweitert werden. Es wurde bei der Klassi-

fizierung bisher zwischen Aufschlägen, Rückhand- und Vorhandschlägen unterschieden. In Zukunft könnten weitere Schlagarten wie Schmetterbälle oder auch der Schupf betrachtet werden. Dafür müssten diese Arten von Schlägen neu aufgenommen werden, damit die Netze mit diesen trainiert werden können. Im Anschluss ist es erforderlich, eine Datengenerierung auf Grundlage der Daten der neuen Schläge durchzuführen, mit dem Ziel, eine erweiterte Datengrundlage für die jeweiligen Schlagarten zu generieren. Gleichzeitig sollte in den Netzen die Ausgabeschicht um eine Anzahl von Neuronen erweitert werden, die der Anzahl der neu hinzugekommenen Schlagarten zur Klassifikation entspricht. Danach muss das Training neu durchgeführt werden. Werden keine zufriedenstellenden Ergebnisse erreicht, muss eine neue *Hyperparameteroptimierung* durchgeführt werden.

Eine weitere Option wäre, die Netze mit der Aktivierungsfunktion *Leaky-ReLU* zu testen, ob sowohl bei der Modellperformanz als auch bei der Trainingsgeschwindigkeit bessere Ergebnisse erzielt werden.

Literaturverzeichnis

[Adamopoulou and Moussiades, 2020] Adamopoulou, E. and Moussiades, L. (2020). An overview of chatbot technology. pages 373–383.

[Alcañiz Raya et al., 2020] Alcañiz Raya, M., Marín-Morales, J., Minissi, M. E., Teruel Garcia, G., Abad, L., and Chicchi Giglioli, I. A. (2020). Machine learning and virtual reality on body movements' behaviors to classify children with autism spectrum disorder. *Journal of Clinical Medicine*, 9(5).

[Alvarez et al., 2011] Alvarez, J., Djaouti, D., et al. (2011). An introduction to serious game definitions and concepts. *Serious games & simulation for risks management*, 11(1):11–15.

[Angelov et al., 2020] Angelov, V., Petkov, E., Shipkovenski, G., and Kalushkov, T. (2020). Modern virtual reality headsets. In *2020 International congress on human-computer interaction, optimization and robotic applications (HORA)*, pages 1–5. IEEE.

[Bansal et al., 2018] Bansal, T., Pachoki, J., Sidor, S., Sutskever, I., and Mordatch, I. (2018). Emergent complexity via multi-agent competition.

[Bekele and Champion, 2019] Bekele, M. K. and Champion, E. (2019). A comparison of immersive realities and interaction methods: Cultural learning in virtual heritage. frontiers in robotics and AI 6 (sept. 2019), 14.

[Bengio et al., 2017] Bengio, Y., Goodfellow, I., and Courville, A. (2017). *Deep learning*, volume 1. MIT press Cambridge, MA, USA.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Bowman and McMahan, 2007] Bowman, D. A. and McMahan, R. P. (2007). Virtual reality: How much immersion is enough? *Computer*, 40(7):36–43.

[Checa and Bustillo, 2020] Checa, D. and Bustillo, A. (2020). A review of immersive virtual reality serious games to enhance learning and training. *Multimedia Tools and Applications*, 79:5501–5527.

[Clow, 2012] Clow, D. (2012). The learning analytics cycle: Closing the loop effectively. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, LAK '12, page 134–138, New York, NY, USA. Association for Computing Machinery.

[Craighead et al., 2008] Craighead, J., Burke, J., and Murphy, R. (2008). Using the unity game engine to develop sarge: A case study. In *Proceedings of the 2008 Simulation Workshop at the International Conference on Intelligent Robots and Systems (IROS 2008)*, volume 4552.

[Cummings and Bailenson, 2016] Cummings, J. J. and Bailenson, J. N. (2016). How immersive is enough? A meta-analysis of the effect of immersive technology on user presence. *Media psychology*, 19(2):272–309.

[Dinh et al., 1999] Dinh, H. Q., Walker, N., Hodges, L. F., Song, C., and Kobayashi, A. (1999). Evaluating the importance of multi-sensory input on memory and the sense of presence in virtual environments. In *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*, pages 222–228. IEEE.

[Freire et al., 2016] Freire, M., Serrano-Laguna, Á., Iglesias, B. M., Martínez-Ortiz, I., Moreno-Ger, P., and Fernández-Manjón, B. (2016). *Game Learning Analytics: Learning Analytics for Serious Games*, pages 1–29. Springer International Publishing, Cham.

[Gao et al., 2019] Gao, J., Galley, M., and Li, L. (2019). Neural approaches to conversational AI.

[Göbl et al., 2021] Göbl, B., Kriglstein, S., and Hlavacs, H. (2021). Conversational interfaces in serious games: Identifying potentials and future research directions based on a systematic literature review. *CSEDU (1)*, pages 108–115.

[Goutte and Gaussier, 2005] Goutte, C. and Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European conference on information retrieval*, pages 345–359. Springer.

[Günther and Fritsch, 2010] Günther, F. and Fritsch, S. (2010). Neuralnet: Training of neural networks. *R J.*, 2(1):30.

[Haarnoja et al., 2018] Haarnoja, T., Pong, V., Hartikainen, K., Zhou, A., Dalal, M., and Levine, S. (2018). Soft actor critic-deep reinforcement learning with real-world robots.

[Haji and Abdulazeez, 2021] Haji, S. H. and Abdulazeez, A. M. (2021). Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 18(4):2715–2743.

[Hecker et al., 2023] Hecker, D., Voss, A., Paaß, G., and Wirtz, T. (2023). Big Data 2.0 – mit synthetischen Daten KI-Systeme stärken. *Wirtschaftsinformatik & Management*, 15(2):161–167.

[Hittmeir et al., 2019] Hittmeir, M., Ekelhart, A., and Mayer, R. (2019). On the utility of synthetic data: An empirical evaluation on machine learning tasks. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, New York, NY, USA. Association for Computing Machinery.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

[Jadeja and Varia, 2017] Jadeja, M. and Varia, N. (2017). Perspectives for evaluating conversational AI.

[Jain et al., 1996] Jain, A., Mao, J., and Mohiuddin, K. (1996). Artificial neural networks: A tutorial. *Computer*, 29(3):31–44.

[Jiao, 2020] Jiao, A. (2020). An intelligent chatbot system based on entity extraction using RASA NLU and neural network. *Journal of Physics: Conference Series*, 1487(1):012014.

[Juliani et al., 2018] Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., and Lange, D. (2018). Unity: A general platform for intelligent agents.

[Jurafsky and Martin, 2000] Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, USA, 1st edition.

[Kalay, 2022] Kalay, A. F. (2022). Generating synthetic data with the nearest neighbors algorithm. *arXiv preprint arXiv:2210.00884*.

[Lewis and Sauro, 2018] Lewis, J. R. and Sauro, J. (2018). Item benchmarks for the system usability scale. *Journal of Usability Studies*, 13(3).

[Lewis and Jacobson, 2002] Lewis, M. and Jacobson, J. (2002). Game engines. *Communications of the ACM*, 45(1):27.

[Lin, 2022] Lin, R. (2022). Analysis on the selection of the appropriate batch size in CNN neural network. In *2022 International Conference on Machine Learning and Knowledge Engineering (MLKE)*, pages 106–109.

[Masetti and Di Giandomenico, 2020] Masetti, G. and Di Giandomenico, F. (2020). Analyzing forward robustness of feedforward deep neural networks with LeakyReLU activation function through symbolic propagation. In *ECML PKDD 2020 Workshops*, pages 460–474, Cham. Springer International Publishing.

[Miles et al., 2012] Miles, H. C., Pop, S. R., Watt, S. J., Lawrence, G. P., and John, N. W. (2012). A review of virtual environments for training in ball sports. *Computers & Graphics*, 36(6):714–726.

[Moore and DeNero, 2011] Moore, R. C. and DeNero, J. (2011). L1 and L2 regularization for multiclass hinge loss models.

[Neumann et al., 2018] Neumann, D. L., Moffitt, R. L., Thomas, P. R., Loveday, K., Watling, D. P., Lombard, C. L., Antonova, S., and Tremeer, M. A. (2018). A systematic review of the application of interactive virtual reality to sport. 22(3):183–198.

[Nimavat and Champaneria, 2017] Nimavat, K. and Champaneria, T. (2017). Chatbots: An overview. Types, architecture, tools and future possibilities.

[Oviatt et al., 2018] Oviatt, S., Grafsgaard, J., Chen, L., and Ochoa, X. (2018). *Multimodal Learning Analytics: Assessing Learners' Mental State during the Process of Learning*, page 331–374. Association for Computing Machinery and Morgan & Claypool.

[Parsons and Cobb, 2016] Parsons, S. and Cobb, S. (2016). State-of-the-art of virtual reality technologies for children on the autism spectrum. In *Technology and Students with Special Educational Needs*, pages 77–88. Routledge.

[Pascanu et al., 2013] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.

[Petrov et al., 2018] Petrov, E. V., Mustafina, J., Alloghani, M., Galiullin, L., and Tan, S. Y. (2018). Learning analytics and serious games: Analysis of interrelation. In *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*, pages 153–156.

[Qiumei et al., 2019] Qiumei, Z., Dan, T., and Fenghua, W. (2019). Improved convolutional neural network based on fast exponentially linear unit activation function. *Ieee Access*, 7:151359–151367.

[Rodriguez, 2008] Rodriguez, A. (2008). Restful web services: The basics. *IBM developerWorks*, 33(2008):18.

[Rubio-Tamayo et al., 2017] Rubio-Tamayo, J. L., Gertrudix Barrio, M., and García García, F. (2017). Immersive environments and virtual reality: Systematic review and advances in communication, interaction and simulation. *Multimodal Technologies and Interaction*, 1(4):21.

[Sharma and Joshi, 2020] Sharma, R. K. and Joshi, M. (2020). An analytical study and review of open source chatbot framework, Rasa. *Int. J. Eng. Res*, 9(06):1011–1014.

[Sharma et al., 2017] Sharma, S., Sharma, S., and Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316.

[Shintani et al., 2021] Shintani, M., Lee, J. H., and Okamoto, S. (2021). Digital pen for handwritten alphabet recognition. In *2021 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE.

[Smagulova and James, 2019] Smagulova, K. and James, A. P. (2019). A survey on LSTM memristive neural network architectures and applications. *The European Physical Journal Special Topics*, 228(10):2313–2324.

[Sonnet, 2022] Sonnet, D. (2022). Neuronale Netze. In *Neuronale Netze Kompakt*, IT kompakt, pages 17–70. Springer Fachmedien Wiesbaden GmbH, Germany.

[Su et al., 2018] Su, P.-H., Mrkšić, N., Casanueva, I., and Vulić, I. (2018). Deep learning for conversational AI. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 27–32, New Orleans, Louisiana. Association for Computational Linguistics.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

[Tan et al., 2015] Tan, M., Santos, C. d., Xiang, B., and Zhou, B. (2015). LSTM-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

[Vakalopoulou et al., 2023] Vakalopoulou, M., Christodoulidis, S., Burgos, N., Colliot, O., and Lepetit, V. (2023). Deep learning: basics and convolutional neural networks (CNN). In Colliot, O., editor, *Machine Learning for Brain Disorders*. Springer.

[Van Houdt et al., 2020] Van Houdt, G., Mosquera, C., and Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, 53:5929–5955.

[Vandeput, 2021] Vandeput, N. (2021). Underfitting. In *Data Science for Supply Chain Forecasting*, pages 37–40. Walter de Gruyter GmbH, Germany.

[Werner, 2020] Werner, M. (2020). Flache neuronale Netze für die Klassifizierung: Shallow neural networks for clasifications. In *Digitale Bildverarbeitung*, pages 349–382. Springer Fachmedien Wiesbaden, Wiesbaden.

[Xu et al., 2019] Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. (2019). Modeling tabular data using conditional gan. In *Advances in Neural Information Processing Systems*.

[Young et al., 2013] Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013). Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

[Zacharias et al., 2018] Zacharias, J., Barz, M., and Sonntag, D. (2018). A survey on deep learning toolkits and libraries for intelligent user interfaces.

Anhang A

Leitfaden zum Arbeiten mit Git

How to GitLab

How do you use GitLab properly?

This small Pamphlet will guide the PG MMI 2 through the correct way of working with Git, Issues, Branches and more.

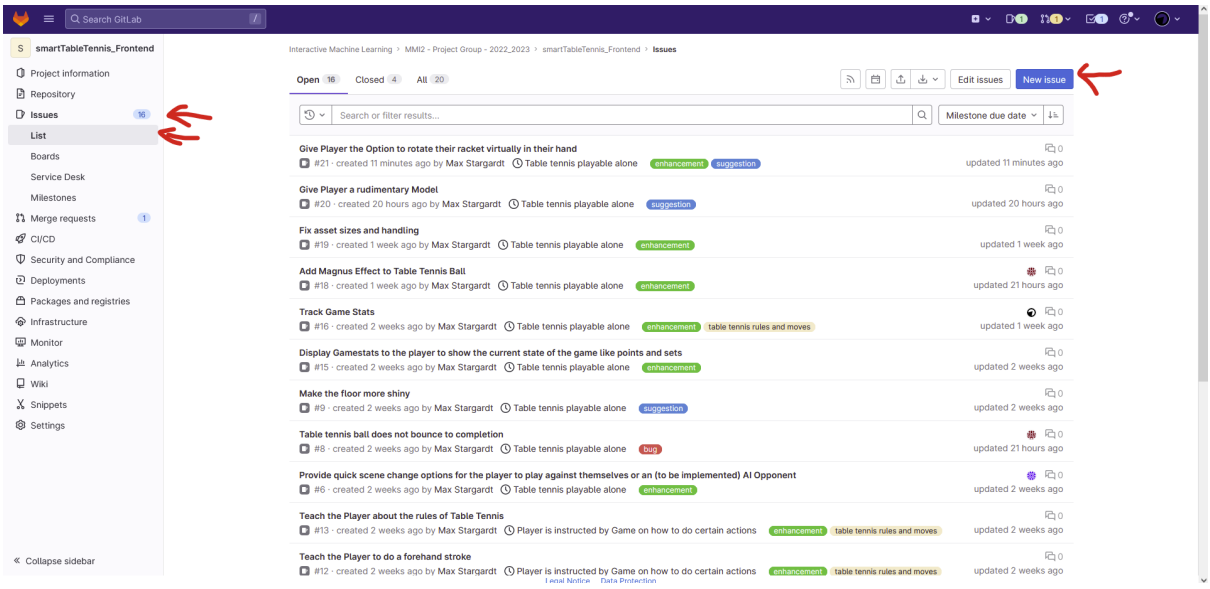
What is Git?

Git is a Version Control System (**VCS**) that tracks all Project Progress made as so-called Commits or Artifacts. A **Commit** is a Snapshot of the current Changes to the Codebase since the last Commit. This Commit-Chain goes down to the very first commit of the codebase. Each Commit corresponds to a **Branch**, which is a copy of a code base from a certain point in time on. A Branch can split from any Branch at any time if needed so. That means two Branches, that split from the same host Branch at the same time, can have a different code progress if different commits have been made to these branches.

If you are unfamiliar with the general usage of Git, you can find multiple online Tutorials on the web to further your Knowledge of Git as a whole. One suitable tutorial can be found under:
https://www.simplilearn.com/tutorials/git-tutorial/what-is-git

How to Issue

Issues are used to describe work that needs to be done. This could be an enhancement of the project or a bugfix or something else. In Gitlab you create an Issue by clicking the “New Issue” button on the Issues-List page on Gitlab.



An Issue should always have a short but indicative title of the work package it describes. The Description should always contain every needed information for the work package. This means ways of reproducing an error for a bug, or a verbose description of what an enhancement should do and how it should do it, or a general outline of what a suggestion suggests.
This leads us to **Labels**. A Label is a very short, colored flair that will be displayed next to an Issue to display what the issue is about. These labels should at least include but are not limited to:

- enhancement
- suggestion
- bug
- critical
- documentation
- user study

Further Labels can be added as needed under the Project Information Tab. But it should always be communicated to the team on how to use added labels.

Each label corresponds to some part of a project.

- **Enhancements** represent any and all features that are added to the project
- **Suggestions** represent any and all features, documentations, ideas, etc. that are not needed for the project to progress but could make it look better or do some minute detail better.
- **Bugs** represent any feature, that is not working as intended and have to be fixed
- **Critical** represents any work package that is of utmost importance, which has to be done ASAP
- **Documentation** represents code refactorings or general documentation of the Project. Be it in code or off code
- **User Study** represents any work package that is directly related to the User Study that has to be created.

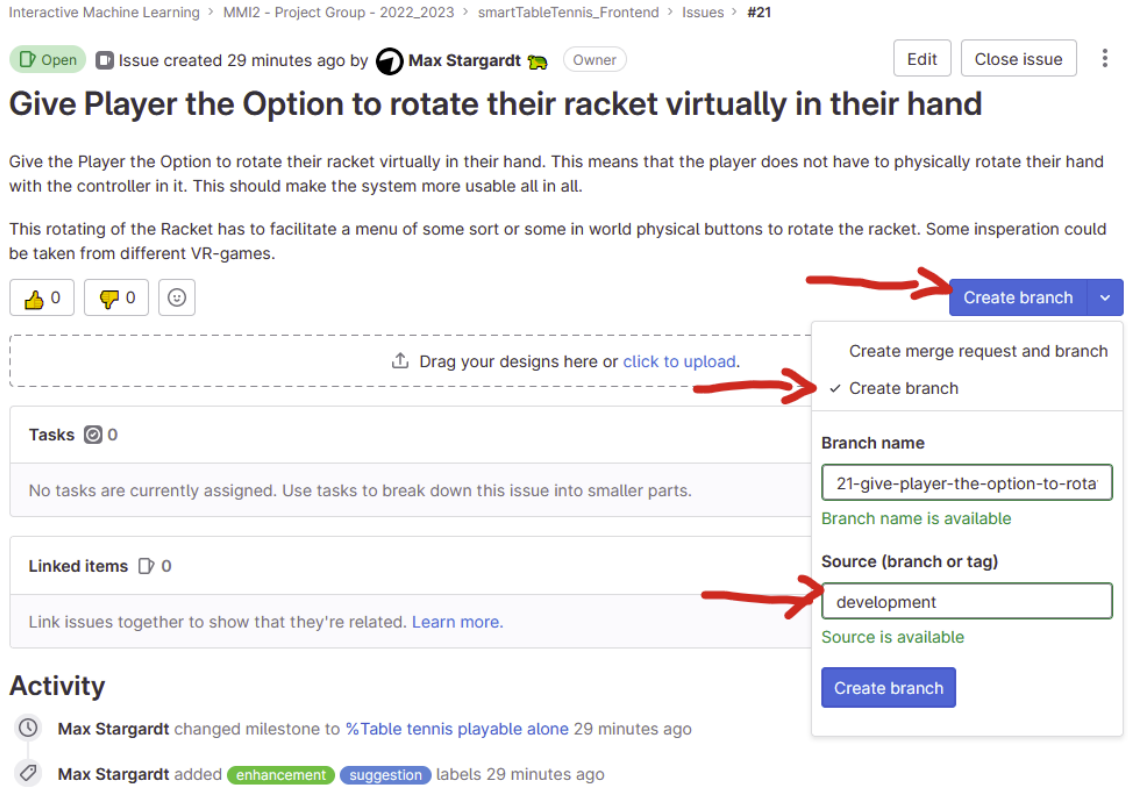
If a label has been selected, a Milestone can be selected. The selected Milestone should always relate to the Issue at hand.

How to Milestone

A Milestone defines a Work package that is too big to be defined as a single Issue. Therefore, a Milestone contains many Issues, a short but descriptive title and a more extensive description of the to be completed work package. Milestones creations are analogous to issue creation.
After each Milestone has been reached, a Merge Request that merges the Development Branch with the Master Branch has to be created and approved.

How to Branch

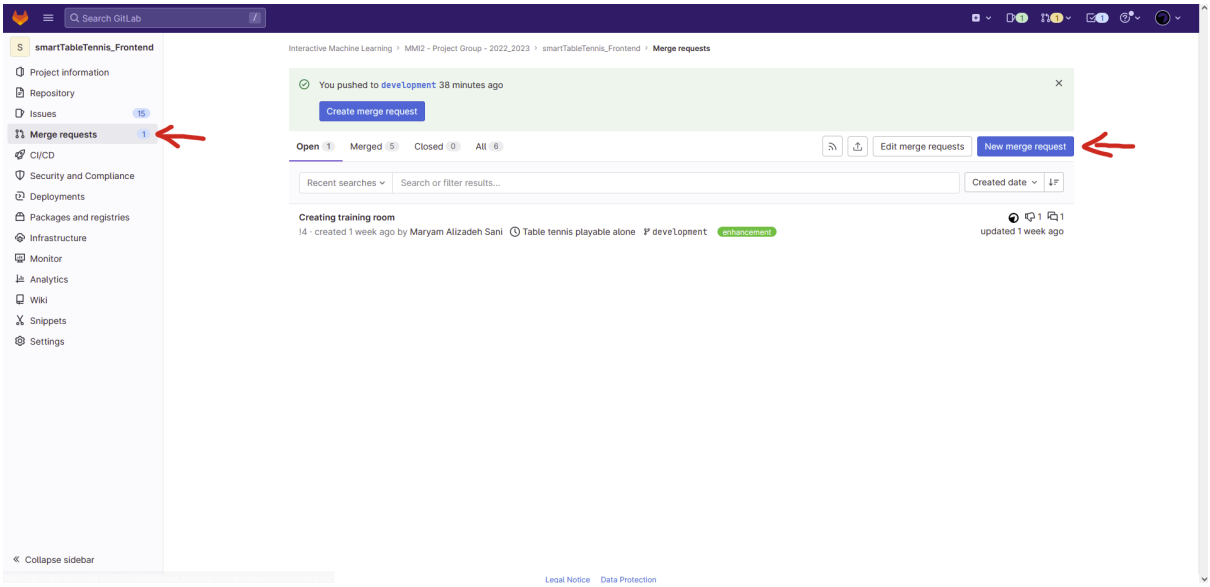
We agreed to work together using feature Branches. Feature Branches are Branches that directly link to an Issue and try to solve said Issue. This means that a Branch is only ever used to fix exactly one Issue. You are *forbidden* to fix multiple issues in one Branch.
You can easily create a Feature Branch by clicking on the “Create Branch” button in the Issue View.



Always make sure to check the “Create Branch” option instead of the “Create Merge Request and Branch” Option. You also always have to make sure to use the “development”-Branch as a source. You can use your Branch as normal after creating it.

How to Merge Request

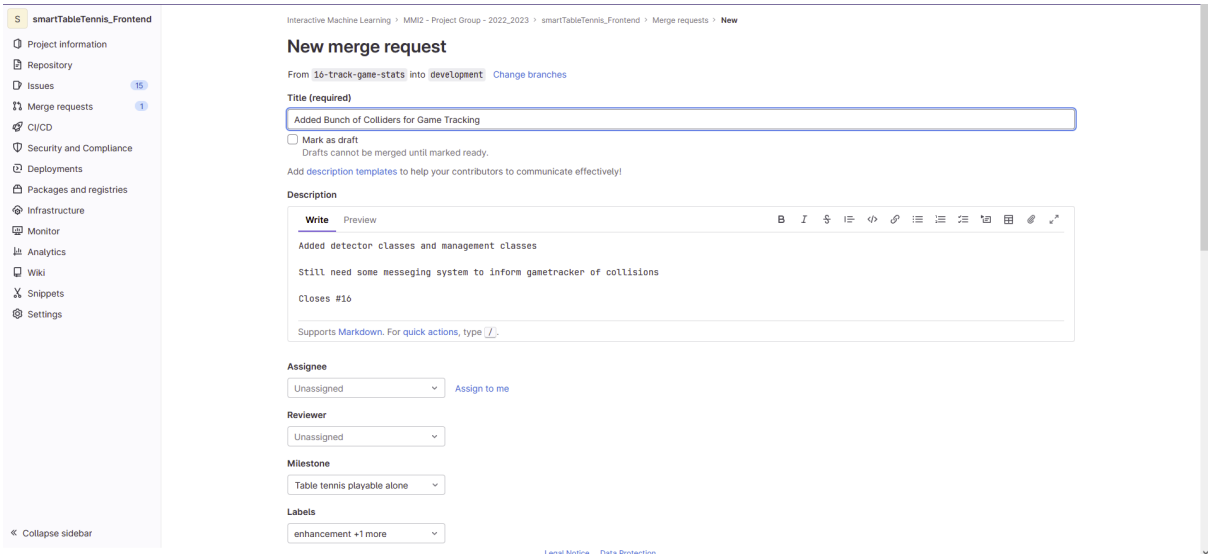
After finishing an Issue, a Merge Request should be created that merges a feature Branch with the development Branch.
You should always locally merge the development Branch with your feature Branch before trying to merge the feature Branch with the development Branch, as this indicates merge conflicts that need to be solved safely.



Creating a Merge Request is analogous to creating an Issue.

When creating a Merge Request, select your feature Branch as source and always make sure to select the development Branch as target.'

After Creating a Merge Request that solves an Issue, you should see the view below.



The Description should contain your last commit as well as a "Closes #00" Sticker, that signifies which issue has been solved.

A Merge Request should always be labeled identically to its issue. It should also have an Assigned Reviewer that has to review the done work, to verify that the corresponding Issue has indeed been solved to completion and without bugs.

When investigating the Merge Options, always make sure to **uncheck** "Delete source branch when merge request is accepted" as well as to **uncheck** "Squash commits when merge request is accepted". This ensures an intact history of the repository.

When reviewing a Merge Request, give (dis-)approval to indicate whether you're ok to merge the branch. If you disapprove a branch, write a comment why you disapproved and what needs to be fixed. This could be a broken code convention, logic errors, or something else.

After a Branch has been merged, always make sure to go the its corresponding Issue and mark it as closed.

Because our GitLab is hosted by the DFKI we cannot change the GitLab settings to only allow approved Merge Requests to be merged. So be responsive and only merge Branches that have the approval of the assigned Reviewer.

How to organize work

- If issues relate to one another, describe them as linked and actually link them via "#00" in their description. This tells others to watch out for possible feature collisions.
- Tell others to review your request if they are assigned as a reviewer
- Use Issues appropriately and don't inflate their usage to every little workpiece
- Assign reviewers that somewhat know about the issue, so that they don't have to guess what a feature might do, or how it might work.
- Try to avoid branching a feature branch from another feature branch, this creates messy merge conflicts.
- When solving merge conflicts, always be vigilant to not overwrite another developer's progress
- The Master Branch should always contain a deployable state of the project with little to no known bugs. The Development Branch can contain bugs, that should be fixed, and is used to store completed features until a milestone has been reached.
- You are **forbidden** to directly commit to the Master or Development Branch. This ensues an intact repository

Anhang B

Studienantrag

Carl von Ossietzky
Universität
Oldenburg

Kommission für Forschungsfolgenabschätzung und Ethik

Begleitendes Übersichtsformular für Anträge

1. Name und Anschrift: Bengt Lüers

Ist eine/r der Antragsteller/-innen Arzt/Ärtzin: ja ☐ nein ☒

2. Titel der Studie

Evaluation eines VR-Tischtennisspiels mit intelligenten Assistenzfunktionen

3. Es handelt sich um einen

Erstantrag ☒ Folgeantrag ☐

Drs. Nr. des Erstantrags: **Klicken Sie hier, um Drs. Nr. einzugeben**

4. Zusammenfassung (max 250 Wörter)

In der Studie soll die Nutzbarkeit und Funktionalität eines VR-Tischtennisspiel evaluiert werden. Bei VR (Virtuelle Realität) handelt es sich um eine computergenerierte Wirklichkeit in einem dreidimensionalen Raum. Zusätzlich soll eine intelligente Assistenzfunktion in Form eines Chatbots bewertet werden.

Dabei durchlaufen die Probanden drei verschiedene Phasen in einer virtuellen Welt. In der ersten Phase sollen sich die Teilnehmer an die virtuelle Umgebung gewöhnen und die Steuerung lernen. Danach beginnen die Probanden in Phase 2 mit sich selbst Tischtennis zu spielen. In Phase 3 werden die Benutzer gegen eine künstliche Intelligenz antreten. Zwischen Phase 2 und 3 gibt es eine 10-minütige Pause zur Erholung. Während des Aufenthalts in der virtuellen Welt haben die Probanden die Möglichkeit mit einem Chatbot zu interagieren, um Hilfestellungen und Rückmeldung zu bekommen.

Zum Schluss werden die Teilnehmer unterschiedliche Fragebögen ausfüllen, um die Nutzbarkeit und Funktionalität des Tischtennisspiels und Chatbots zu evaluieren.

5. Fragen zum Forschungsvorhaben

Werden Patienten untersucht, die im Rahmen der Studie ambulant oder stationär behandelt werden? ja ☐ nein ☒

Werden den Patienten/Probanden Medikamente verabreicht oder werden sie invasiven Prozeduren

unterzogen? ja ☐ nein ☒

Fällt die Studie unter gesetzliche Regelungen (AMG, MPG, Strahlenschutzgesetz etc.)?
ja ☐ nein ☒

Aus welchem Wissenschaftsgebiet entstammt die Fragestellung?
Informatik – Mensch-Maschine-Interaktion und Künstliche Intelligenz

Handelt es sich um eine Studie an Minderjährigen? ja ☐ nein ☒

Handelt es sich um eine Studie an nicht-einwilligungsfähigen Erwachsenen? ja ☐ nein ☒

6. Eingereichte Dokumente (die Punkte unten zum Ankreuzen)
Antrag ☒
Probandeninformation ☒
Einverständniserklärung ☒
In ihrer jeweils aktuellen Form in einer pdf-Datei (mit durchnummerierten Seiten) ☒

Antrag auf Stellungnahme der Kommission für Forschungsfolgenabschätzung und Ethik

1. Bezeichnung des Forschungsvorhabens

Evaluation eines VR-Tischtennisspiels mit intelligenten Assistenzfunktionen

2. Name und Kontaktdaten des Antragstellers[1] (Dienstanschrift):

Bengt Lüers

Applied Artificial Intelligence (AAI) – Uni Oldenburg

Marie-Curie-Straße 1

26129 Oldenburg

bengt.lueers@uni-oldenburg.de

+49 441 998337444

3. Angaben zu den Rahmenbedingungen des Vorhabens

Es handelt sich um einen Antrag ohne Finanzierung durch eine *Förderinstitution*. Eine Stellungnahme der Ethikkommission wird *nicht verlangt*.

Die Datenerhebungen finden im Zeitraum von 07.08.2023 bis 27.08.2023 im Core IML in der Heiligengeiststraße 6, 26121 Oldenburg statt.

4. Gegenstand und Verfahren des Vorhabens

Gegenstand. Das Ziel der Studie ist, die Nutzbarkeit eines VR-Tischtennisspiels zu evaluieren. Es soll getestet werden, wie realitätsnah das Tischtennisspiel ist und ob eine intelligente Assistenzfunktion in Form eines Chatbots den Benutzern helfen kann.

Methoden. Messung der Spielerbewegung anhand Hand-, Augen- und Schlägerbewegung für die Fehleranalyse. Zusätzlich werden Fragebögen ausgefüllt.

Experimentelle Aufgaben. Die Probanden werden gebeten, in einem virtuellen Raum Tischtennis zu spielen. Es beginnt mit einer 10-minütigen Einführungsphase, wo die Teilnehmer sich an das virtuelle Umfeld gewöhnen und sich mit der Steuerung vertraut machen. Daraufhin werden die Teilnehmer 10 Minuten lang mit sich selbst Tischtennis spielen. Nach dem

[1] Im Fall von studentischen Abschlussarbeiten muss der betreuende Hochschullehrer mit als Antragsteller auftreten und den Antrag mit unterschreiben

Spielen gibt es eine 5-10-minütige Pause, um sich vom Aufenthalt in der virtuellen Welt zu erholen. Zum Abschluss folgt ein 20-minütiges Spiel gegen einen Bot. Während des Aufenthalts in VR sollen die Probanden mit einem Chatbot interagieren, um Rückmeldung und Tipps zum Spielverlauf bekommen. Der spielerische Bot und der Chatbot werden hierbei von einer künstlichen Intelligenz kontrolliert. Nach dem Spiel werden die Teilnehmer Fragebögen für die Evaluation auszufüllen.

Durchführung. Am Anfang werden die Teilnehmer über Evaluationsziele und datenschutzrechtliche Implikationen aufgeklärt. Die Probanden werden ebenfalls darauf hingewiesen, dass sie zu jeder Zeit die VR-Brille abnehmen können. Der erste Fragebogen wird dabei schon ausgefüllt. Danach beginnt der Hauptteil der Studie, der in drei Phasen aufgeteilt ist (siehe Experimentelle Aufgaben). Während des Aufenthalts in VR haben die Probanden zu jeder Zeit die Möglichkeit die VR-Brille abzunehmen. Nach den 3 Phasen werden die Probanden dazu gebeten 4 verschiedene Fragebögen auszufüllen.

Auswertung. Für die Evaluation der Studie werden verschiedene standardisierte Fragebögen verwendet. Für die Evaluation der Nutzbarkeit des Tischtennisspiels werden die Fragebögen „System Usability Scale (SUS)“ und „User Experience Questionaire (UEQ)“ ausgefüllt. Der SUS-Fragebogen bezieht sich dabei mehr auf die Funktionalität des Produkts, während der UEQ sich auf die Benutzerfreundlichkeit bezieht. Da sich die Teilnehmer in einer Trainings- und Lernumgebung befinden werden wir ebenfalls den NASA-TLX-Fragebogen verwenden, um mentale und physische Beanspruchung zu evaluieren.

Des Weiteren erstellen wir einen weiteren Fragebogen, wo u.a. nach Rückmeldung und Verbesserungsvorschlägen gefragt wird. Zusätzlich wird überprüft ob Vorkenntnisse in Tischtennis und VR, Auswirkungen auf die Leistungen der Teilnehmer haben.

Körperliche Beanspruchung. Wir erwarten körperliche Anstrengung durch den gesamten Aufenthalt in der VR. Da die Teilnehmer mit sich selbst und gegen einen Bot Tischtennis spielen, kann es zu körperlicher Anstrengung und Ermüdung kommen. Dazu kommt, dass nicht alle Teilnehmer den Aufenthalt in VR gut vertragen. Da die Belastungen sehr individuell verschieden sind, laden wir die Teilnehmenden dazu ein, auf diese Belastungssymptome selbst zu achten und bei den Durchführenden der Studie Pause einzufordern, sofern erforderlich.

Mentale Beanspruchung. Durch den längeren Aufenthalt in der VR erwarten wir eine gewisse mentale Beanspruchung. Da die Teilnehmer gegen einen Bot spielen, können kompetitive Teilnehmer zusätzlichen Stress spüren.

Preisgabe persönlicher Informationen. Folgende persönliche Informationen müssen preisgegeben werden, um in der Studie auf die Probanden eingehen zu können:

- Name

- Kontaktdaten
- Alter
- Geschlecht
- VR-Kenntnisse
- Tischtennis-Kenntnisse
- Rechts- oder Linkshänder

Täuschung und Aufklärung. Wir arbeiten nicht mit Täuschung. Die Teilnehmer werden vor dem Beginn der Studie über den Ablauf der Studie, unser Evaluationsziel und die datenschutzrechtlichen Implikationen aufgeklärt.

5. Angaben zu Aufzeichnung, Aufbereitung, Speicherung und Löschung der Daten

Personenbezogene Daten. Name, Kontaktdaten, Alter, Geschlecht, VR-Kenntnisse, Tischtennis-Kenntnisse, Rechts- oder Linkshänder

Datenschutz. Verwendung einer Kodierliste

Kodierliste und persönliches Codewort. Datum der Löschung: 27.08.2024

Es existiert eine Kodierliste, welche die Namen der Probanden einem Pseudonym zuordnet. Die Daten sind nur den Pseudonymen zugeordnet und nicht den Namen direkt. Wenn eine Löschung der Daten gewünscht wird, benötigt die Person lediglich ihren vollständigen Namen und die Daten zum dazugehörigen Pseudonym werden gelöscht. Die Kodierliste wird separat von den in der Studie erhobenen Daten aufbewahrt. Die erhobenen Daten werden beim DFKI in einem Repository gespeichert.

Löschung der Daten. Die Kodierliste wird ein Jahr nach Beendigung der Studie gelöscht. Die anonymisierten Daten werden nach guter wissenschaftlicher Praxis 10 Jahre aufbewahrt.

6. Gewinnung der Personenstichprobe und Vergütung von Probanden

Rekrutierung. Ausschreibung am schwarzen Brett der Universität Oldenburg. Zusätzlich werden in Tischtennisvereinen nach Teilnehmern gesucht.

Merkmale der Personenstichprobe. 15-20 Personen mit unterschiedlicher Erfahrung in Tischtennis.

Einschluss- und Ausschlusskriterien.

- Keine diagnostizierte neurologische oder psychische Erkrankung (z.B. Depressionen, ADHS, etc.)
- Keine nicht durch eine Sehhilfe zu korrigierende Augenerkrankung (z.B. Schielen, Rot-Grün-Schwäche, etc.)

Die Teilnehmer werden vor dem Beginn der Studie, von dem Versuchsleiter befragt, ob eine oder mehrere der Ausschlusskriterien auf Sie zutreffen.

Teilnahmevergütung. Die Teilnehmer erhalten eine Vergütung von 12€ pro Stunde. Die Vergütung erfolgt über eine Überweisung des DFKI. Dazu muss die Kontonummer angegeben werden, die separat von den Untersuchungsergebnissen aufbewahrt wird. Falls die Teilnehmer den Versuch früher abbrechen, erfolgt eine Vergütung aufgerundet für die bis dahin angetretene Zeit.

7. Freiwilligkeit der Teilnahme und Rücktritt

Freiwilligkeit. Alle Teilnehmer bekommen vorweg Teilnehmerinformationen zugesendet. Zusätzliche Erwähnung der Freiwilligkeit bei der Begrüßung der Probanden. Alle Teilnehmer bekommen vorab die gleichen Startbedingungen.

Rücktritt. Die Teilnehmer werden vorab darüber informiert, dass eine Rücktrittsmöglichkeit ohne Nachteile und das Recht der eigenen Daten bis zum Zeitpunkt der Anonymisierung der Daten jederzeit möglich ist.

9. Informiertheit und Einwilligung

Informiertheit. Es ist eine vollständige Informiertheit der Teilnehmer gewährt.

Einwilligung. Nach Information der Probanden wird deren Einwilligung eingeholt. Die Einwilligungserklärung ist dem Antrag beigefügt.

Ort, Datum Unterschrift Antragsteller

Carl von Ossietzky Universität Oldenburg

Ammerländer Heerstraße 114-118, 26129 Oldenburg

Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)

DFKI GmbH, Trippstadter Strasse 122, 67663 Kaiserslautern

Projektleiter: Bengt Lüers

Ansprechpartner für eventuelle Rückfragen:

Versuchsleiter: Philipp Peikenkamp

E-Mail: philipp.peikenkamp@uni-oldenburg.de

Allgemeine Teilnehmerinformation über die Untersuchung
DFKI

Titel der Studie: Evaluation eines VR-Tischtennisspiels mit intelligenten Assistenzfunktionen

Herzlich willkommen bei unserer Studie mit dem Titel "Evaluation eines VR-Tischtennisspiels mit intelligenten Assistenzfunktionen“! Wir danken Ihnen für Ihr Interesse an dieser Studie.
Die Studie ist für unsere Projektgruppe, die aus der Kooperation zwischen der Uni Oldenburg und dem DFKI entstanden ist.
Das Ziel der Studie ist, die Nutzbarkeit eines VR-Tischtennisspiels zu evaluieren. Bei VR (Virtuelle Realität) handelt es sich um eine computergenerierte Wirklichkeit in einem dreidimensionalen Raum. Es soll getestet werden, wie realitätsnah das Tischtennisspiel ist und ob eine intelligente Assistenzfunktion in Form eines Chatbots die Benutzer helfen kann.

Ablauf der Studie

Das folgende Experiment besteht aus 40 Minuten Aufenthalt in einer virtuellen Welt mit einer 10-minütgen Pause. Vor und nach dem Experiment werden Fragebögen ausgefüllt. Insgesamt dauert das Experiment 1-1,5 Stunden.
Ihre Aufgabe ist es, drei verschiedene Phasen zu durchlaufen. In der ersten Phase werden

Sie sich an die VR-Umgebung gewöhnen und die Steuerung lernen. In der zweiten Phase werden Sie Tischtennis mit sich selbst spielen. Dabei wird eine geklappte Tischtennisplatte verwendet. Abschließen werden Sie gegen einen Bot spielen und versuchen möglichst viele Punkte zu erzielen. Während dieser Phasen haben Sie die Möglichkeit mit einem Chatbot zu reden, der Fragen beantworten kann und Rückmeldung zu vergangenen Spielzügen gibt. Während der VR-Zeit tragen Sie ein VR-Headset und spielen Tischtennis mit zwei Controllern. Das VR-Headset umfasst den gesamten Kopf oberhalb der Nase, die Ohren können aber bei Bedarf freigelassen werden. Falls Sie sich in der VR unwohl fühlen, haben Sie jederzeit die Möglichkeit, das Headset abzunehmen und den Versuch zu unterbrechen. Der Aufenthalt in VR wird im Stehen verbracht. Die Versuchsleitung achtet darauf, dass es zu keinen Kollisionen mit anderen Objekten kommt.
Über den Fragebogen werden mehrere personenbezogene Daten erfasst. Bei den Daten handelt es sich um das Geschlecht, Alter, Händigkeit, VR- und Tischtenniskenntnisse.

Datenschutz

Kategorien personenbezogener Daten, die verarbeitet werden

Von der Datenverarbeitung sind folgende personenbezogene Daten umfasst:

- Name
- Kontaktdaten
- Geschlecht
- Alter
- Händigkeit
- VR-Kenntnisse
- Tischtennis-Kenntnisse
- Daten aus dem Spielverlauf
- Beantwortete Fragebögen

Es werden keine Videodaten erhoben.

Verfahren der Datenverarbeitung

Zur Verschlüsselung der Daten wird eine Kodierliste verwendet. Sie können eine Löschung der Daten bis zum 27.08.2024 beantragen. Wenden Sie sich dafür an das Sekretariat des DFKIs (iml-sek@dfki.de). Bei Nennung ihres Namens werden die Daten zum dazugehörigen Pseudonym gelöscht. Die Liste der Namen und Pseudonymen wird separat von den erhobenen Daten aufbewahrt. Nach Ablauf eines Jahres wird die Kodierliste gelöscht. Danach sind die Daten keiner Person mehr zuzuordnen und von Ihnen nicht mehr löschbar.
Sollten Sie noch Fragen haben, wenden Sie sich damit bitte an den Versuchsleiter.

Freiwilligkeit und Anonymität

Die Teilnahme an der Studie ist freiwillig. Sie können jederzeit und ohne Angabe von Gründen die Teilnahme an dieser Studie beenden, ohne dass Ihnen daraus Nachteile entstehen. Auch wenn Sie die Studie vorzeitig abbrechen, haben Sie Anspruch auf *eine entsprechende*

Vergütung / entsprechende Versuchspersonenstunden für den bis dahin erbrachten Zeitaufwand.

Die im Rahmen dieser Studie erhobenen, oben beschriebenen Daten und persönlichen Mitteilungen werden vertraulich behandelt. So unterliegen diejenigen Projektmitarbeiter, die durch direkten Kontakt mit Ihnen über personenbezogene Daten verfügen, der Schweigepflicht. Des Weiteren wird die Veröffentlichung der Ergebnisse der Studie in anonymisierter Form erfolgen, d. h. ohne dass Ihre Daten Ihrer Person zugeordnet werden können.

Datenschutz

Kodierliste: Die Erhebung und Verarbeitung Ihrer oben beschriebenen persönlichen Daten erfolgt pseudonymisiert im DFKI unter Verwendung einer Nummer und ohne Angabe Ihres Namens. Es existiert eine Kodierliste auf Papier, die Ihren Namen mit der Nummer verbindet. Die Kodierliste ist nur den Versuchsleitern und dem Projektleiter zugänglich; das heißt, nur diese Personen können die erhobenen Daten mit meinem Namen in Verbindung bringen. Die Kodierliste wird in einem abschließbaren Schrank aufbewahrt und spätestens am 27.08.2024 vernichtet. Ihre Daten sind dann anonymisiert. Damit ist es niemandem mehr möglich, die erhobenen Daten mit Ihrem Namen in Verbindung zu bringen. Die anonymisierten Daten werden mindestens 10 Jahre gespeichert. Solange die Kodierliste existiert, können Sie die Löschung aller von Ihnen erhobenen Daten verlangen. Ist die Kodierliste aber erst einmal gelöscht, können wir Ihren Datensatz nicht mehr identifizieren. Deshalb können wir Ihrem Verlangen nach Löschung Ihrer Daten nur so lange nachkommen, wie die Kodierliste existiert.

Vergütung

Für die Teilnahme an der Untersuchung erhalten Sie eine Vergütung in Höhe von 12 € pro Stunde. Die Vergütung wird im Nachgang per Überweisung ausgezahlt. Bei einer Überweisung der Vergütung müssen Sie Ihre Kontoverbindung angeben. Alle diesbezüglichen Informationen werden völlig separat von den Untersuchungsdaten aufbewahrt. Sollten Sie den Versuch früher abbrechen, so erfolgt eine Vergütung aufgerundet für die bis dahin angetretene Zeit.

Dauer der Verarbeitung

Nach Auswertung aller Daten und Abschluss der Studie, spätestens jedoch nach Wegfall des Forschungszwecks, werden Ihre Daten schnellstmöglich – **insbesondere bevor eine Veröffentlichung zu wissenschaftlichen Zwecken** (z.B. Fachartikel, Tagungsbeiträge, wissenschaftliche Datenbanken [Open Data Repositories]) **stattfindet** – anonymisiert. Hierzu ist die Verantwortliche nach § 13 Absatz 2 Satz 1 Niedersächsisches Datenschutzgesetz (NDSG) verpflichtet. Anonymisierung bedeutet, dass niemand mehr Ihre Daten Ihrer Person zuordnen kann. Ihre Daten sind dann nicht mehr „personenbezogen“ im Sinne der datenschutzrechtlichen Rechtsvorschriften.

Ihre Abrechnungsdaten bleiben bis zu ihrer Löschung nach zehn Jahren nur noch für das Buchhaltungs-/Abrechnungssystem und dessen Mitarbeiter sichtbar.

Verwendung der Daten

Diese Studie dient ausschließlich Forschungszwecken. Die erhobenen Daten sollen für die Studenten der Projektgruppe verwendet werden, um das Produkt zu evaluieren.

Die Sie betreffenden personenbezogenen Daten werden ohne Ihre Einwilligung nicht an Dritte weitergegeben.
Die Sie betreffenden personenbezogenen Daten werden nicht zu anderen als den angegebenen Zweck weiterverarbeitet

Kontaktdaten der Verantwortlichen und des Datenschutzbeauftragten

Verantwortliche	**Datenschutzbeauftragter**
Carl von Ossietzky Universität Oldenburg (KdöR), gesetzlich vertreten durch den Präsidenten Ammerländer Heerstr. 114-118 26129 Oldenburg Telefon: +49 441 798-0 Telefax: +49 441 798-3000 E-Mail: internet@uol.de Internet: https://uol.de	Carl von Ossietzky Universität Oldenburg Der Datenschutzbeauftragte Ammerländer Heerstr. 114-118 26129 Oldenburg Tel.: 0441-798-4196 E-Mail: dsuni@uol.de Internet: https://uol.de/datenschutz/

Ansprechpartner
Zur Kontaktaufnahme, insbesondere zur Wahrnehmung Ihrer Betroffenenrechte, wenden Sie sich bitte an dem Briefkopf genannten Versuchsleiter

Rechtsgrundlage
Die Rechtsgrundlage für die Erhebung Sie betreffenden personenbezogener Daten ist z.B. Einwilligung gem. Art. 6 Abs. 1 lit. a DSGVO.

Rechte als Betroffener
- Sie haben ein **Recht auf Auskunft** über die Sie betreffenden personenbezogenen Daten (Art. 15 DSGVO).
- Sie können unverzüglich von dem Verantwortlichen **Berichtigung** Sie betreffender unrichtiger oder **Vervollständigung** unvollständiger personenbezogener Daten verlangen (Art. 16 DSGVO).
- Sie sind hiermit darüber informiert worden, dass Sie jederzeit eine **Löschung** der Sie betreffenden personenbezogenen Daten verlangen können (Art. 17 DSGVO).
- Sie können die **Einschränkung der Verarbeitung** verlangen, soweit die gesetzlichen Voraussetzungen vorliegen (Art. 18 DSGVO).

- Sie haben das Recht, die Sie betreffenden personenbezogenen Daten, **in einem strukturierten, gängigen und maschinenlesbaren Format zu erhalten** und diese Daten einem anderen Verantwortlichen zu übermitteln (Art. 20 DSGVO).
- Sie können jederzeit gegen die Verarbeitung Sie betreffender personenbezogener Daten **Widerspruch einlegen**, die aufgrund von Artikel 6 Abs. 1 lit. e oder f DSGVO erfolgt (Art. 21 DSGVO).
- Sie können die erteilte **Einwilligung jederzeit mit Wirkung für die Zukunft widerrufen,** ohne, dass die Rechtmäßigkeit der aufgrund der Einwilligung bis zum Widerruf erfolgten Verarbeitung berührt wird (Art. 7 Abs. 3 DSGVO) *Sofern Rechtsgrundlage auf Einwilligung basiert.*

Bereitstellung der Daten und Folgen der Nichtbereitstellung
Die Bereitstellung der Sie betreffenden personenbezogenen Daten ist weder vertraglich noch gesetzlich vorgeschrieben. Sie sind nicht dazu verpflichtet, Sie betreffende personenbezogene Daten bereitzustellen. Die Nichtbereitstellung hätte zur Folge, dass Sie an der Studie nicht teilnehmen können.

Beschwerderecht bei einer Aufsichtsbehörde
Falls Sie der Ansicht sind, dass die Verarbeitung Ihrer personenbezogenen Daten gegen Datenschutzvorschriften verstößt, wenden Sie sich bitte an die/den Datenschutzbeauftragte/n der Verantwortlichen (s.o.). Unabhängig hiervon haben Sie ein Recht auf **Beschwerde** bei der zuständigen Aufsichtsbehörde. Die zuständige Aufsichtsbehörde ist:

Die Landesbeauftragte für den Datenschutz Niedersachsen
Prinzenstraße 5
30159 Hannover

Telefon: 0511 120-4500
Telefax: 0511 120-4599
Email: poststelle@lfd.niedersachsen.de

Carl von Ossietzky Universität Oldenburg

Ammerländer Heerstraße 114-118, 26129 Oldenburg

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

DFKI GmbH, Trippstadter Strasse 122, 67663 Kaiserslautern

Projektleiter: Bengt Lüers

Ansprechpartner für eventuelle Rückfragen:

Versuchsleiter: Philipp Peikenkamp

E-Mail: philipp.peikenkamp@uni-oldenburg.de

Einwilligungserklärung

DFKI

Titel der Studie: *Evaluation eines VR-Tischtennisspiels mit intelligenten Assistenzfunktionen*

Ich (Name des Teilnehmers /der Teilnehmerin in Blockschrift)

_____

bin schriftlich über die Studie und den Versuchsablauf aufgeklärt worden. Ich willige ein, an dem mir beschriebenen Versuch teilzunehmen. Sofern ich Fragen zu dieser vorgesehenen Studie hatte, wurden sie von Herrn/Frau _____ vollständig und zu meiner Zufriedenheit beantwortet.

„Kodierliste“

> *Mit der beschriebenen Erhebung und Verarbeitung der Daten Geschlecht, Alter, Händigkeit, VR-Kenntnisse und Tischtennis-Kenntnisse bin ich einverstanden. Die Aufzeichnung und Auswertung dieser Daten erfolgt pseudonymisiert im DFKI unter Verwendung einer Nummer und ohne Angabe meines Namens. Es existiert eine Kodierliste auf Papier, die meinen Namen mit dieser Nummer verbindet. Diese Kodierliste ist*

nur den Versuchsleitern und dem Projektleiter zugänglich, das heißt, nur diese Personen können die erhobenen Daten mit meinem Namen in Verbindung bringen. Spätestens am 27.08.2024, wird die Kodierliste gelöscht. Meine Daten sind dann anonymisiert. Damit ist es niemandem mehr möglich, die erhobenen Daten mit meinem Namen in Verbindung zu bringen. Mir ist bekannt, dass ich mein Einverständnis zur Aufbewahrung bzw. Speicherung dieser Daten widerrufen kann, ohne dass mir daraus Nachteile entstehen. Ich bin darüber informiert worden, dass ich jederzeit eine Löschung all meiner Daten verlangen kann. Wenn allerdings die Kodierliste bereits gelöscht ist, kann mein Datensatz nicht mehr identifiziert und also auch nicht mehr gelöscht werden. Meine Daten sind dann anonymisiert. Ich bin einverstanden, dass meine anonymisierten Daten zu Forschungszwecken weiter verwendet werden können und mindestens 10 Jahre gespeichert bleiben.

Ich hatte genügend Zeit für eine Entscheidung und bin bereit, an der o.g. Studie teilzunehmen. Ich weiß, dass die Teilnahme an der Studie freiwillig ist und ich die Teilnahme jederzeit ohne Angaben von Gründen beenden kann. Ich weiß, dass ich in diesem Fall Anspruch auf das Geld (12€/h) für die bis dahin erbrachten Stunden habe.

Eine Ausfertigung der Teilnehmerinformation über die Untersuchung und eine Ausfertigung der Einwilligungserklärung habe ich erhalten. Die Teilnehmerinformation ist Teil dieser Einwilligungserklärung.

Ort, Datum & Unterschrift des Teilnehmers: Name des Teilnehmers in Druckschrift:

_____ _____

Ort, Datum & Unterschrift des Versuchsleiters: Name des Versuchsleiters in Druckschrift:

_____ _____

Anhang C

Studienskript

Studienskript

Vor der Studie:

1. Foto vom Raum machen und Bengt und Ray schicken
2. Kontroller und VR-Headset desinfizieren
3. PC starten und einloggen
4. Frontend starten
    a. Unity Editor starten
    b. Kamera Einstellungen anpassen
    c. OBS starten und überprüfen, ob das Spiel aufgenommen wird
    d. Start drücken
5. Backend starten
    a. Hinweis: Rasa Server braucht etwas länger (1-2 Minuten)
    b. Über Docker:
        i. Docker starten
        ii. Folgende Commands ausführen:
        iii. Pull Docker image:

```
1. docker logout
2. docker login git.ni.dfki.de:5050
3. docker rmi git.ni.dfki.de:5050/iml/pg-mmi2/mmi2.api-gateway:python3.10-latest
4. docker pull git.ni.dfki.de:5050/iml/pg-mmi2/mmi2.api-gateway:python3.10-latest
```

Run Docker image:

```
docker run --rm -it --publish 1717:1717 git.ni.dfki.de:5050/iml/pg-mmi2/mmi2.api-gateway:python3.10-latest
```

    a. Über Pycharm:
        i. MMI2.API-Gateway klonen
        ii. Schritte der Readme Datei folgen
5. Fragebögen öffnen
6. Testen ob Backend und Frontend funktionsfähig sind
    a. Einmal eine Trainingssession starten, um zu gucken, ob alles funktioniert

Studie:

1. Begrüßung **(5 Minuten)**
    a. Einwilligungserklärung einholen
    b. Name und IBAN des Probanden eintragen
    c. Einmal grob das Ziel der Studie erklären
    d. Kurz den folgenden Ablauf der Studie beschreiben
    e. Folgende Hinweise dem Probanden mitgeben:
        i. Proband kann zu jeder Zeit die virtuelle Welt verlassen
        ii. Studie findet im Stehen statt
        iii. Ball nicht zu schnell schlagen
        iv. Lieber im Raum teleportieren als zu Fuß laufen

v. Spiel ist noch nicht ganz ausgereift und kann einige Bugs enthalten
vi. Bugs gerne mitteilen
vii. Bei Personen mit Brillen fragen, ob sie mit, oder ohne Brille spielen möchten
viii. Grob erklären, wie man den Chatbot benutzt und dass der Chatbot etwas Zeit braucht, um Fragen zu beantworten

2. Erster Bogen wird ausgefüllt (Fragebogen 1) **(5 Minuten)**
3. Folgende Dinge beachten, während der Proband in der VR ist
    a. Genug Platz, sodass es zu keinen Kollisionen kommt
    b. Eine Person hält die Kabel fest, sodass die Probanden sich nicht verheddern
4. 1.Phase **(10 Minuten)**
    a. VR-Brille aufsetzen und richtig einstellen
    b. Steuerung erklären/Tutorial starten
    c. Proband gewöhnt sich an der VR und stellt Rotation des Schlägers ein
5. 2.Phase **(10 Minuten)**
    a. Proband spiel Tischtennis gegen die Platte
6. Pause **(5 Minuten)**
7. 3.Phase **(20 Minuten)**
    a. Gametracking wird gestartet und Proband verlässt den Trainingsraum
    b. Proband spielt gegen Bot
8. VR-Brille abnehmen
9. OBS beenden
10. Fragebogen werden ausgefüllt (Fragebogen 2, NASA TLX, SUS, UEQ) **(10 Minuten)**
11. Für die Teilnahme bedanken und nach weiteren Anmerkungen fragen **(? Minuten)**

Nach der Studie:

1. Überprüfen, ob alle Fragebogen gesendet worden sind.
2. Alle Daten vom Frontend und Backend mit der Probanden ID separat speichern
    a. Frontend Daten: Werden im Videos Ordner gespeichert
    b. Backend Daten: csv Dateien sind im evaluation Ordner gespeichert
3. Frontend und Backend beenden
4. Alle Geräte ausschalten
5. Foto vom Raum machen und Bengt und Ray schicken

Anhang D

Fragebögen

D.1 Fragebogen 1

Erfahrungen in VR und Tischtennis

Als erstes Werden ein paar Fragen zu ihrer Erfahrung in Tischtennis und Virtual Reality gestellt

* Gibt eine erforderliche Frage an

1. Wie viel Erfahrung haben sie im Tischtennisspielen? *

Markieren Sie nur ein Oval.

	Ich habe noch nie Tischtennis gespielt				Ich spiele regelmäßig Tischtennis
	1	2	3	4	5

2. Wie bewerten Sie Ihre Fähigkeiten im Tischtennis? *

Markieren Sie nur ein Oval.

- Anfänger/in
- Fortgeschritten
- Experte/in

3. Wie viel Erfahrung haben sie mit VR-Brillen?

Markieren Sie nur ein Oval.

Ich habe noch nie eine VR-Brille verwendet

1
2
3
4
5

Ich verwende regelmäßig VR-Brillen

4. Wie geschickt sind Sie darin, sich in der virtuellen Welt zu bewegen *

Markieren Sie nur ein Oval.

Keine Erfahrung/ungeschickt

1
2
3
4
5

Sehr geschickt

5. Haben Sie bereits Tischtennis in VR gespielt? *

Markieren Sie nur ein Oval.

- Ja
- Nein

6. Wie beurteilen Sie diese Erfahrung im Vergleich zum Spielen von Tischtennis in der realen Welt? *

Markieren Sie nur ein Oval.

- Besser als in der realen Welt
- Gleich gut wie in der realen Welt
- Schlechter als in der realen Welt
- Keine Meinung/ Keine Erfahrung

D.2 Fragebogen 2

Bewertung der VR-Tischtennis und Chatbot-Nutzung

* Gibt eine erforderliche Frage an

1. Wie wichtig finden Sie die folgenden Eigenschaften eines Tischtennisspiels in der Virtuellen Realität? *

Markieren Sie nur ein Oval pro Zeile.

	Völlig unwichtig	Eher unwichtig	Mittelmäßig	Eher wichtig	Sehr wichtig
Bedienbarkeit	◯	◯	◯	◯	◯
Realismus	◯	◯	◯	◯	◯
Levelanzahl	◯	◯	◯	◯	◯
Mehrspieler	◯	◯	◯	◯	◯
Avatardetails	◯	◯	◯	◯	◯

2. Wie hilfreich war der Chatbot, um das Spiel zu erlernen *

Markieren Sie nur ein Oval.

Gar nicht hilfreich

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Sehr hilfreich

3. Wie einfach bzw. schwierig ist es, die Anweisungen des Chatbots zu verstehen? *

Markieren Sie nur ein Oval.

Sehr einfach

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Sehr schwierig

4. Wie würden Sie die Reaktionsfähigkeit des Chatbots bewerten? *

Markieren Sie nur ein Oval.

	Sehr schnell
1	
2	
3	
4	
5	
	Zu langsam

5. Bewerten Sie folgende Aussage: Ich hatte das Gefühl, dass der Chatbot dazu beigetragen hat, meine Fähigkeiten im Tischtennis zu verbessern. *

Markieren Sie nur ein Oval.

Stimmte voll und ganz zu

1
2
3
4
5

Stimme gar nicht zu

6. Wie einfach bzw. schwierig war es, mit dem Chatbot zu interagieren? *

Markieren Sie nur ein Oval.

Sehr einfach

1 ⬭

2 ⬭

3 ⬭

4 ⬭

5 ⬭

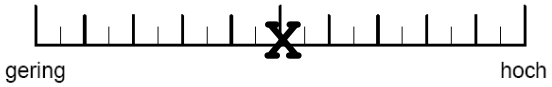
Sehr schwierig

D.3 NASA-TLX

Beanspruchungshöhe

Geben Sie jetzt für jede der unten stehenden Dimensionen an, wie hoch die Beanspruchung war. Markieren Sie dazu bitte auf den folgenden Skalen, in welchem Maße Sie sich in den sechs genannten Dimensionen von der Aufgabe beansprucht oder gefordert gesehen haben:

Beispiel:



Geistige Anforderungen
Wie viel geistige Anstrengung war bei der Informationsaufnahme und -verarbeitung erforderlich (z.B. Denken, Entscheiden, Rechnen, Erinnern, Hinsehen, Suchen...)? War die Aufgabe leicht oder anspruchsvoll, einfach oder komplex, erforderte sie hohe Genauigkeit oder war sie fehlertolerant?

Körperliche Anforderungen
Wie viel körperliche Aktivität war erforderlich (z.B. Ziehen, Drücken, Drehen, Steuern, Aktivieren,…)? War die Aufgabe leicht oder schwer, einfach oder anstrengend, erholsam oder mühselig?

Zeitliche Anforderungen
Wie viel Zeitdruck empfanden Sie hinsichtlich der Häufigkeit oder dem Takt, mit dem Aufgaben oder Aufgabenelemente auftraten? War die Abfolge langsam und geruhsam oder schnell und hektisch?

Leistung
Wie erfolgreich haben Sie Ihrer Meinung nach die vom Versuchsleiter (oder Ihnen selbst) gesetzten Ziele erreicht? Wie zufrieden waren Sie mit Ihrer Leistung bei der Verfolgung dieser Ziele?

Anstrengung
Wie hart mussten sie arbeiten, um Ihren Grad an Aufgabenerfüllung zu erreichen?

Frustration
Wie unsicher, entmutigt, irritiert, gestresst und verärgert (versus sicher, bestätigt, zufrieden, entspannt und zufrieden mit sich selbst) fühlten Sie sich während der Aufgabe?

Kontrollieren sie bitte, ob Sie zu allen Fragen Angaben gemacht haben. Bei Unklarheiten wenden Sie sich bitte an die Versuchsleiterin / den Versuchsleiter.

Subskalen:

Der Wert jeder Subskala ist ein Einzelmesswert. Je nach Fragestellung können Subskalen auch einzeln verwendet oder untereinander kombiniert werden.

Auswertung:

Jedem Kreuz wird ein ganzzahliger Wert von 0 bis 20 zugeordnet, wobei „0“ als gering und „20“ als hoch gewertet wird. Befindet sich ein Kreuz zwischen zwei Teilstrichen der Skala, wird derjenige ganzzahlige Wert vergeben, der sich näher am Kreuzungspunkt der Angabe des Teilnehmers befindet.

Quellen:

Hart, S. G. (2006). NASA-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting,* 904-908. Santa Monica: HFES.

D.4 User Experience Questionnaire

Bitte geben Sie Ihre Beurteilung ab.

Um das Produkt zu bewerten, füllen Sie bitte den nachfolgenden Fragebogen aus. Er besteht aus Gegensatzpaaren von Eigenschaften, die das Produkt haben kann. Abstufungen zwischen den Gegensätzen sind durch Kreise dargestellt. Durch Ankreuzen eines dieser Kreise können Sie Ihre Zustimmung zu einem Begriff äußern.

Beispiel:

attraktiv	○	⊗	○	○	○	○	○	unattraktiv

Mit dieser Beurteilung sagen Sie aus, dass Sie das Produkt eher attraktiv als unattraktiv einschätzen.

Entscheiden Sie möglichst spontan. Es ist wichtig, dass Sie nicht lange über die Begriffe nachdenken, damit Ihre unmittelbare Einschätzung zum Tragen kommt.

Bitte kreuzen Sie immer eine Antwort an, auch wenn Sie bei der Einschätzung zu einem Begriffspaar unsicher sind oder finden, dass es nicht so gut zum Produkt passt.

Es gibt keine „richtige“ oder „falsche“ Antwort. Ihre persönliche Meinung zählt!

Bitte geben Sie nun Ihre Einschätzung des Produkts ab. Kreuzen Sie bitte nur einen Kreis pro Zeile an.

	1	2	3	4	5	6	7		
unerfreulich	○	○	○	○	○	○	○	erfreulich	1
unverständlich	○	○	○	○	○	○	○	verständlich	2
kreativ	○	○	○	○	○	○	○	phantasielos	3
leicht zu lernen	○	○	○	○	○	○	○	schwer zu lernen	4
wertvoll	○	○	○	○	○	○	○	minderwertig	5
langweilig	○	○	○	○	○	○	○	spannend	6
uninteressant	○	○	○	○	○	○	○	interessant	7
unberechenbar	○	○	○	○	○	○	○	voraussagbar	8
schnell	○	○	○	○	○	○	○	langsam	9
originell	○	○	○	○	○	○	○	konventionell	10
behindernd	○	○	○	○	○	○	○	unterstützend	11
gut	○	○	○	○	○	○	○	schlecht	12
kompliziert	○	○	○	○	○	○	○	einfach	13
abstoßend	○	○	○	○	○	○	○	anziehend	14
herkömmlich	○	○	○	○	○	○	○	neuartig	15
unangenehm	○	○	○	○	○	○	○	angenehm	16
sicher	○	○	○	○	○	○	○	unsicher	17
aktivierend	○	○	○	○	○	○	○	einschläfernd	18
erwartungskonform	○	○	○	○	○	○	○	nicht erwartungskonform	19
ineffizient	○	○	○	○	○	○	○	effizient	20
übersichtlich	○	○	○	○	○	○	○	verwirrend	21
unpragmatisch	○	○	○	○	○	○	○	pragmatisch	22
aufgeräumt	○	○	○	○	○	○	○	überladen	23
attraktiv	○	○	○	○	○	○	○	unattraktiv	24
sympathisch	○	○	○	○	○	○	○	unsympathisch	25
konservativ	○	○	○	○	○	○	○	innovativ	26

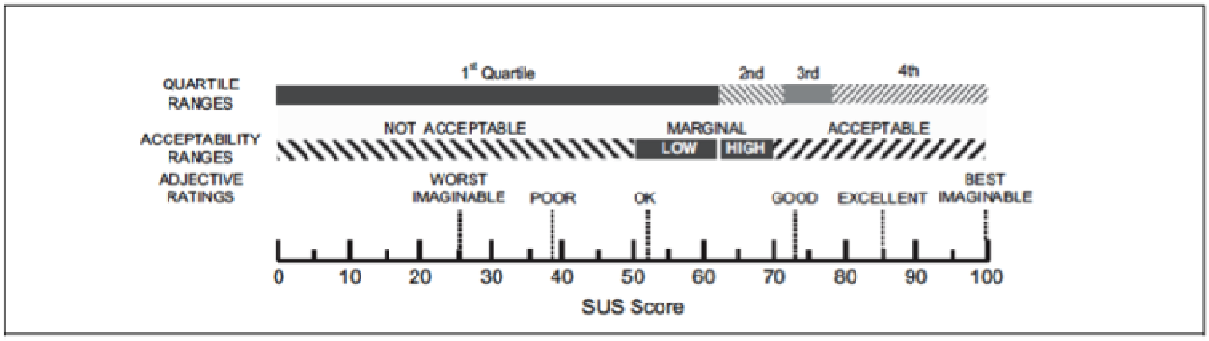
D.5 System Usability Scale

Bitte beurteilen Sie auf der folgenden Skala, inwieweit Sie den Aussagen zustimmen.

	lehne völlig ab				stimme völlig zu
1. Ich denke, ich würde die Website/ App regelmäßig nutzen.	0	1	2	3	4
2. Die Website/ App erscheint mir unnötig kompliziert.	0	1	2	3	4
3. Ich finde, die Website/ App ist einfach zu benutzen.	0	1	2	3	4
4. Ich denke, ich bräuchte technische Unterstützung um die Website/ App nutzen zu können.	0	1	2	3	4
5. Ich finde, dass die verschiedenen Funktionen der Website/ App gut integriert sind.	0	1	2	3	4
6. Die Website/ App erscheint mir zu uneinheitlich.	0	1	2	3	4
7. Ich glaube, dass die meisten Leute die Benutzung der Website/ App schnell erlernen können.	0	1	2	3	4
8. Die Website/ App erscheint mir sehr umständlich zu benutzen.	0	1	2	3	4
9. Ich fühle mich bei der Benutzung der Website/ App sehr sicher.	0	1	2	3	4
10. Ich musste einiges lernen, um mit der Website/ App zurecht zu kommen.	0	1	2	3	4

Auswertung:

Jedem Kreuz wird die entsprechende Zahl von 0 bis 4 zugeordnet.
Folgende Items, da sie negativ formuliert sind, müssen vor der weiteren Berechnung umgepolt werden: 2, 4, 6, 8 ,10.
Anschließend wird die Summe aus allen 10 Items berechnet und mit 2,5 multipliziert.
Durch diesen Schritt wir der Gesamtwert (SUS-Score) als Prozentwert angegeben. Dieser kann nach folgendem Schema interpretiert werden:



Quelle:

Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction, 24:6*, 574-594.

Anhang E

Werbetext

Studie zum Thema:
Evaluation eines VR-Tischtennisspiels mit intelligenten Hilfsfunktionen

Im Rahmen unserer Projektgruppe “Multimodal Multisensor Interaction” der Uni Oldenburg wurde ein VR-System (Virtual Reality) entwickelt, bei dem der Anwender gegen einen Bot Tischtennis spielen soll. Während dem Spiel wird der Anwender von einem Chatbot unterstützt. Ziel der Studie ist zu evaluieren, wie benutzerfreundlich ein Tischtennisspiel in der virtuellen Realität ist und ob eine intelligente Hilfsfunktion den Anwender unterstützen kann.

Voraussetzungen:

- Keine diagnostizierte neurologische oder psychische Erkrankung (z.B. Depressionen, ADHS, etc.)
- Keine körperliche Erkrankung (z.B. Knochenbrüche, etc.)
- Keine nicht durch eine Sehhilfe zu korrigierende Augenerkrankung (z.B. Schielen, Rot-Grün-Schwäche, etc.)
- Vollendung des 18ten Lebensjahr
- Rechtshändig
- Deutschkenntnisse

Dauer: 1-2 Stunden

Ort: Core IML Heiligengeiststraße 6, 26121 Oldenburg

Erhebungszeitraum: 07.08.2023-25.08.2023

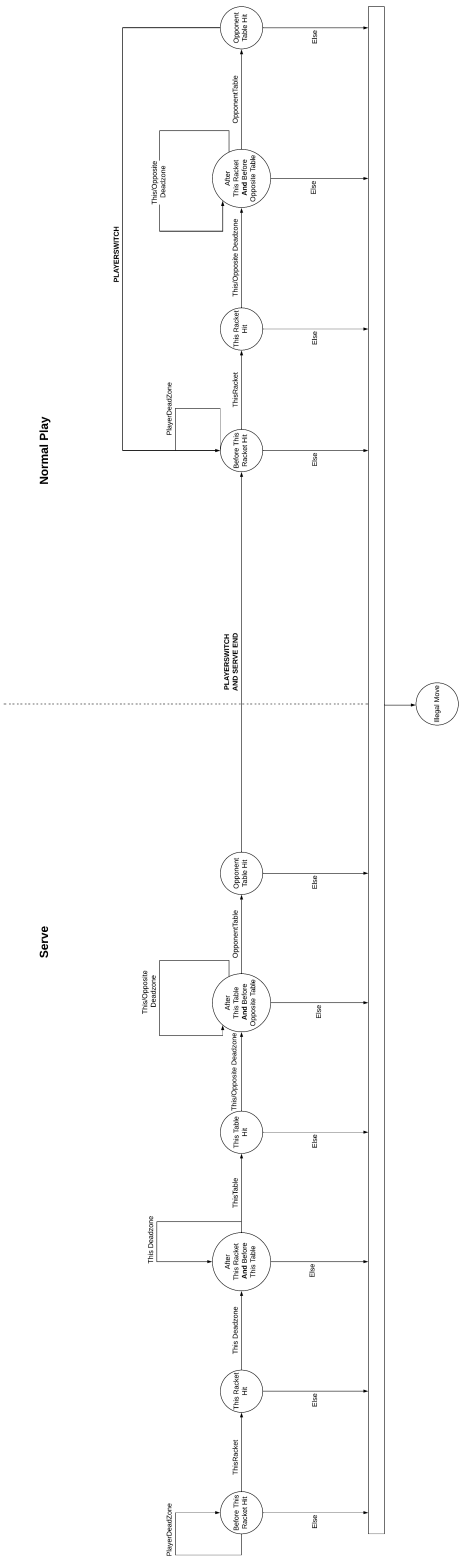
Vergütung: 12€ pro Stunde (Stunden werden aufgerundet)

Wenn Sie teilnehmen möchten, senden Sie eine E-Mail an philipp.peikenkamp@uni-oldenburg.de mit Terminvorschlägen im genannten Zeitraum. Anschließend erhalten Sie weitere Informationen zur Studie. Vielen Dank für Ihr Interesse!

Anhang F

Vorherige Entwürfe des GameStateTracker

F.1 Erster Entwurf des GameStateTrackers


Normal Play
Serve
PLAYERSWITCH
PLAYERSWITCH AND SERVE END
Illegal Move
Before This Racket Hit
After This Racket And Before This Table
After This Table And Before Opponent Table
Opponent Table Hit
This Racket Hit
This Table Hit
ThisRacket
ThisTable
OpponentTable
This Deadzone
This/Opposite Deadzone
Else

F.2 Zweiter Entwurf des GameStateTrackers