



Hochschule Aalen

***Konzeption und Entwicklung eines Chatbots zur
Informationsbeschaffung für Studierende des
Bachelorstudiengangs Wirtschaftsinformatik an
der Hochschule Aalen***

Bachelorarbeit

im Studiengang
Wirtschaftsinformatik

vorgelegt von

Tim Konle

Matr.-Nr.: 76737

am 18.08.2022

an der Hochschule Aalen

Erstprüfer:

Prof. Dr. Manfred Rössle

Zweitprüfer:

Prof. Dr. Marc Fernandes

Eidesstattliche Erklärung

Hiermit erkläre ich, **Tim Konle**, dass ich die Bachelorarbeit mit dem Thema "Konzeption und Entwicklung eines Chatbots zur Informationsbeschaffung für Studierende des Bachelorstudiengangs Wirtschaftsinformatik an der Hochschule Aalen" selbständig verfasst habe.

Weiterhin versichere ich, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, dass alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind. Dasselbe gilt für beigefügte Skizzen und Darstellungen. Des weiteren versichere ich, dass die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Ort, Datum

Unterschrift

Kurzfassung

Die Informationsbeschaffung ist für Studierende meist ein sehr mühsamer Prozess. Vor allem für Studienanfänger, die noch keine Orientierung auf den Webseiten der Hochschulen haben, kann dies meist viel Zeit in Anspruch nehmen. Um Studierende bei dieser Informationsbeschaffung zu unterstützen, wurde in dieser Abschlussarbeit ein Chatbot konzipiert, entwickelt und evaluiert. Dieser Chatbot soll den Studierenden des Studiengangs Wirtschaftsinformatik an der Hochschule Aalen eine schnellere Informationsbeschaffung ermöglichen.

Zusammengefasst beschreibt diese Abschlussarbeit den Prozess der Entwicklung eines Chatbots zur Informationsbeschaffung für Studierende. Dieser Prozess wird von der Schaffung der Wissensbasis und der Anforderungsanalyse, über die Konzeption und Entwicklung bis hin zur Evaluation beschrieben und ausgewertet.

Abstract

Obtaining information is usually a very tedious process for students. Especially for first-year students, who have no orientation on the websites of the universities, this can take a lot of time. In order to support students in this process, a chatbot was designed, developed and evaluated in this thesis. This chatbot is intended to enable students of the Information Systems program at Aalen University to obtain information more quickly.

In summary, this thesis describes the process of developing a chatbot for information retrieval for students. This process is described and evaluated from the creation of the knowledge base and the requirements analysis, through the conception and development, to the evaluation.

Inhaltsverzeichnis

EIDESSTATTLICHE ERKLÄRUNG	2
KURZFASSUNG.....	3
ABSTRACT.....	3
INHALTSVERZEICHNIS	4
ABBILDUNGSVERZEICHNIS	6
TABELLENVERZEICHNIS	6
LISTINGVERZEICHNIS	7
ABKÜRZUNGSVERZEICHNIS.....	8
1 EINLEITUNG	9
1.1 MOTIVATION UND PROBLEMSTELLUNG	9
1.2 ZIELSETZUNG & FORSCHUNGSFRAGEN.....	9
1.3 WISSENSCHAFTLICHE METHODIK.....	10
2 GRUNDLAGEN	11
2.1 CHATBOTS	11
2.1.1 Definition Chatbot.....	11
2.1.2 Geschichte der Chatbots	11
2.1.3 Arten von Chatbots	12
2.2 NATURAL LANGUAGE PROCESSING	14
2.2.1 Vorbearbeitung der Daten	14
2.2.2 Vektorisierung (TF-IDF)	16
2.3 MACHINE LEARNING.....	16
2.3.1 Machine Learning Grundlagen	16
2.3.2 Klassifikationsmodelle	17
2.3.3 Evaluation der Klassifikationsmodelle	19
3 ANFORDERUNGSANALYSE	22
4 KONZEPTION UND ENTWICKLUNG DES PROTOTYPS	23
4.1 PLANUNG UND KONZEPTION	23
4.2 VERWENDETE TECHNOLOGIEN UND BIBLIOTHEKEN	25
4.3 ENTWICKLUNG DES PROTOTYPS	27
4.3.1 Datenbank und Datenbankstruktur	27
4.3.2 Datenerhebung.....	28
4.3.3 Verbindung zur Telegram-API.....	29
4.3.4 Textvorverarbeitung	30
4.3.5 Implementierung der ML-Algorithmen	30
4.3.6 Historie der Benutzerinteraktion.....	31
4.4 EINORDNUNG DES PROTOTYPS.....	32
5 EVALUATION DES PROTOTYPS.....	34

6	FAZIT, LIMITATIONEN UND ZUKUNFTSAUSBLICK	36
6.1	FAZIT	36
6.2	LIMITATIONEN	36
6.3	ZUKUNFTSAUSBLICK	37
	ANHANG A: AUSZÜGE AUS DEM SOURCECODE	38
A.1	FUNKTIONEN DER DATENERHEBUNG	38
A.1.1	<i>Extrahierung der Vorlesungsdaten</i>	<i>38</i>
A.1.2	<i>Abrufen der Informationen von der Webseite</i>	<i>39</i>
A.1.3	<i>Abrufen des RSS-Feeds</i>	<i>40</i>
A.2	VERBINDUNG ZUR TELEGRAM API	40
A.2.1	<i>Einbindung der Command- und Message-Handler</i>	<i>40</i>
A.2.2	<i>Funktionen, die durch Handler aufgerufen werden können</i>	<i>41</i>
A.3	NLP-FUNKTIONEN	43
A.3.1	<i>Vorverarbeitung der Textdaten</i>	<i>43</i>
A.4	ML-FUNKTIONEN	43
A.4.1	<i>Datengrundlage</i>	<i>43</i>
A.4.2	<i>Vorbereitung der Datengrundlage</i>	<i>44</i>
A.4.3	<i>Training und Vergleich der Klassifikationsalgorithmen</i>	<i>44</i>
A.4.4	<i>Einbindung der SVM</i>	<i>45</i>
A.4.5	<i>Abspeicherung der Klassifikationshistorie</i>	<i>45</i>
	ANHANG B: EVALUATION UND EVALUATIONSERGEBNISSE	46
B.1	EVALUATION DURCH USABILITY TEST UND FRAGEBOGEN	46
B.2	AUSWERTUNG DER EVALUATION	49
B.2.1	<i>Fragen und Antworten</i>	<i>49</i>
B.2.2	<i>Feedback der Teilnehmer</i>	<i>49</i>
	QUELLENVERZEICHNIS	51

Abbildungsverzeichnis

Abbildung 1.1: DSR Framework und Zyklen nach Hevner [3].....	10
Abbildung 2.1: Eigene Preprocessing Pipeline auf Basis von Vijayarani et al. [18].....	15
Abbildung 2.2: Hyperebenen im zwei- und dreidimensionalen Merkmalsraum [27].....	18
Abbildung 2.3: Darstellung des KNN-Algorithmus mit $k = 3$ [31].....	19
Abbildung 2.4: Einfache Kontingenztafel nach Olson und Delen [33].	20
Abbildung 4.1: Chatbot Design nach Meshram et al. [35].....	23
Abbildung 4.2: Eigenes Konzept des Chatbots.....	24
Abbildung 4.3: Darstellung der verwendeten Technologien und Bibliotheken.	25
Abbildung 4.4: Darstellung der Datenbankstruktur.....	27
Abbildung 5.1: Frage 9 der Evaluation mit Verhältnis der Antworten.....	35
Abbildung B.1: Frage zur Einordnung der Befragten.	46
Abbildung B.2: Aufgaben zur Durchführung eines Usability Tests.....	47
Abbildung B.3: Frage nach Feedback im Fragebogen.....	48

Tabellenverzeichnis

Tabelle 2.1: Klassifizierung von Chatbots nach Adamopoulou und Moussiades [15].....	13
Tabelle 4.1: Vergleich der Machine Learning Klassifikatoren.	31
Tabelle 4.2: Einordnung des Chatbots nach Adamopoulou und Moussiades [15].....	32
Tabelle B.1: Behauptungen zur Evaluation.....	48
Tabelle B.2: Auswertung der Fragen bei 14 Teilnehmern.	49
Tabelle B.3: Feedback und Verbesserungsvorschläge der Teilnehmer.	49

Listingverzeichnis

<i>Listing 1: Funktion, um Module aus iCal zu extrahieren.</i>	38
<i>Listing 2: Funktion, um Veranstaltungsdaten aus iCal zu extrahieren.</i>	39
<i>Listing 3: Funktion, um Informationen von der Webseite des Studiengangs zu erhalten.</i>	39
<i>Listing 4: Funktion, um RSS-Feed des Studiengangs auszulesen.</i>	40
<i>Listing 5: Einbindung der Command- und Message-Handler.</i>	40
<i>Listing 6: Start-Funktion.</i>	41
<i>Listing 7: Help-Funktion.</i>	41
<i>Listing 8: Funktion zur Modulübersicht.</i>	42
<i>Listing 9: Funktion, um Module zum Stundenplan hinzuzufügen.</i>	42
<i>Listing 10: Funktion, um Module aus dem Stundenplan zu entfernen.</i>	42
<i>Listing 11: Echo-Funktion, um auf Nachrichten ohne Befehl zu antworten.</i>	42
<i>Listing 12: Funktion zur Vorverarbeitung von Textdaten.</i>	43
<i>Listing 13: Auszug aus der Trainingsdaten CSV-Datei.</i>	43
<i>Listing 14: Vorbereitung der Datengrundlage.</i>	44
<i>Listing 15: Einbindung, Training und Vergleich der ML-Algorithmen.</i>	44
<i>Listing 16: Funktion, um Eingabe des Benutzers zu klassifizieren.</i>	45
<i>Listing 17: Funktion zum Abspeichern der Klassifikationshistorie.</i>	45

Abkürzungsverzeichnis

API	Application Programming Interface
AIML	Artificial Intelligence Markup Language
HTML	Hyper Text Markup Language
MIT	Massachusetts Institute of Technology
ML	Machine Learning / maschinelles Lernen
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
SVM	Support Vector Machine
URL	Uniform Resource Locator
XML	Extensible Markup Language

1 Einleitung

1.1 Motivation und Problemstellung

Während des Studiums stoßen Studierende immer wieder auf große Herausforderungen und Hürden. Vor allem der (selbst-) organisatorische Aufwand kann für die Studierenden, insbesondere zu Beginn des Studiums, erdrückend sein. Hierbei ist vor allem die Informationsbeschaffung eine große Hürde. Meist verbringen die Studierenden viel Zeit damit, Informationen über Vorlesungen, Termine, Abgaben etc. zu suchen. Zurzeit ist die Informationsbeschaffung über die Webseite der Hochschule Aalen, die einzige Möglichkeit, die die Studierenden haben, um an diese Informationen zu gelangen. Da diese Webseite nicht intuitiv gestaltet ist, ist dies vor allem für Studienanfänger ein beschwerlicher und oft auch überfordernder Prozess. Eine Lösung, um die Informationsbeschaffung für die Studierenden zu erleichtern, wäre eine Applikation, welche von einem mobilen Endgerät erreichbar ist und in natürlicher Sprache mit den Studierenden kommuniziert.

1.2 Zielsetzung & Forschungsfragen

Im Rahmen dieser Abschlussarbeit soll ein Prototyp eines Chatbots zur Informationsbeschaffung für die Studierenden des Bachelorstudiengangs Wirtschaftsinformatik der Hochschule Aalen konzipiert und entwickelt werden. Der Arbeit liegt eine Semesterarbeit im Wahlmodul Natural Language Processing zugrunde. Hierbei wurde ein vereinfachter Prototyp eines Informationschatbots für Studierende entwickelt. Diese Arbeit soll den Prototypen des Chatbots von Grund auf neu konzipieren und entwickeln. Vor allem die Zeitersparnis soll hierbei im Vordergrund stehen. Der Chatbot soll es den Studierenden ermöglichen Zeit bei der Informationsbeschaffung zu sparen, um sich so auf die wichtigen Dinge im Studium konzentrieren zu können. Diese Ziele führen zu folgenden Forschungsfragen:

1. Wie kann ein Chatbot entwickelt werden, um die Studierenden des Bachelorstudiengangs Wirtschaftsinformatik der Hochschule Aalen bei der Informationsbeschaffung zu unterstützen?
2. Kann ein Chatbot, die Dauer der Informationsbeschaffung für Studierende senken und damit die Effektivität erhöhen?

Der Ansatz zur Entwicklung eines Informationschatbots für den Studiengang Wirtschaftsinformatik an der Hochschule Aalen wird hierbei in Kapitel 4 beschrieben.

Ob dieser es schafft die Dauer der Informationsbeschaffung zu senken, wird in der Evaluation (Kapitel 5) erörtert.

1.3 Wissenschaftliche Methodik

Diese Arbeit folgt der Forschungsmethodologie Design Science Research (DSR) nach Hevner et al. [1]. Ziel des DSR Ansatzes ist es Ergebnisse, sogenannte Artefakte zu entwickeln, die ein praktisches Problem lösen sollen [2]. DSR nach Hevner et al. enthält ein Framework, welches die Design Science Paradigmen darstellt. Hierbei wird auf die Umgebung des Projekts, die Wissensbasis und die eigentliche Entwicklung des Artefakts und deren Zusammenspiel eingegangen. Aufbauend auf dem Framework und den Leitlinien veröffentlichten Hevner et al. 2007 ein Prozessmodell bestehend aus 3 Forschungszyklen, welche iterativ durchlaufen werden [3]. Diese Forschungszyklen sind: Relevance Cycle, Design Cycle und Rigor Cycle.

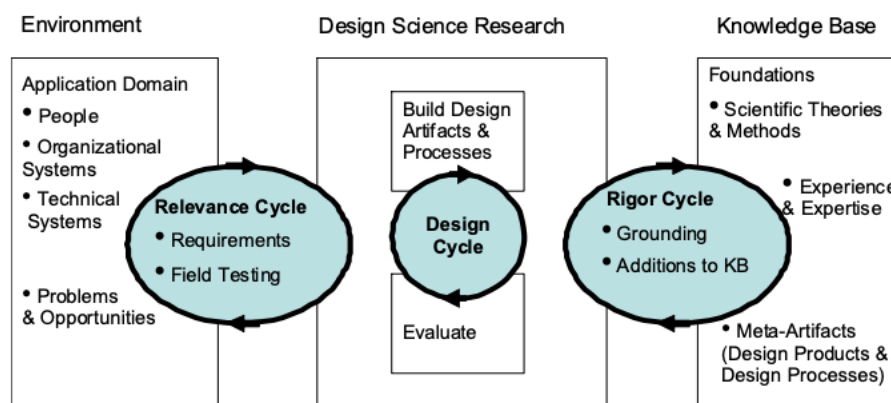


Abbildung 1.1: DSR Framework und Zyklen nach Hevner [3].

Ziel des DSR Ansatzes ist eine Verbesserung des Problems durch die Entwicklung eines Artefakts. Um dies zu erreichen, müssen im Vorhinein die Anforderungen an das Artefakt definiert werden. Dies geschieht im **Relevance Cycle**. In diesem Projekt wurde hierfür eine Anforderungsanalyse durchgeführt (Kapitel 3). Dabei wurde auf das Umfeld, in dem die Applikation verwendet werden soll, eingegangen. Es wurden die betroffenen Personen, die organisatorischen sowie technischen Systeme und die zu lösenden Probleme eruiert.

Dem Relevance Cycle steht der **Rigor Cycle** gegenüber. Design Science stützt sich auf eine breite Wissensbasis. Diese wird in diesem Zyklus geschaffen. Um diese Basis zu schaffen, wurde in dieser Arbeit eine Literaturrecherche durchgeführt. Aus dieser Literaturrecherche entstand das Grundlagen Kapitel (Kapitel 2).

Das Herz des DSR-Ansatzes bildet der **Design Cycle**. In diesem Zyklus wird das Artefakt entwickelt und anschließend evaluiert. Die Konzeption und Entwicklung dieses Chatbots wird in Kapitel 4 beschrieben. Evaluiert wurde der Prototyp, indem er von Probanden getestet wurde und diese im nachhinein Fragen dazu beantworten mussten. Die genaue Auswertung der Evaluation ist in Kapitel 5 zu finden.

2 Grundlagen

In diesem Kapitel werden die Grundlagen erläutert, welche zum weiteren Verständnis dieser Abschlussarbeit benötigt werden. Hierzu gehören die Grundlagen der Chatbot-Technologie, Natural Language Processing sowie Grundlagen im Bereich des maschinellen Lernens.

2.1 Chatbots

2.1.1 Definition Chatbot

Einer der wichtigsten Aspekte des menschlichen Lebens ist die Kommunikation. Vor allem der Informationsaustausch, welcher mit ihr einhergeht, ist für das menschliche Leben von wichtiger Bedeutung. Diese Kommunikation ist mittlerweile aber nicht nur zwischenmenschlich möglich. Durch Fortschritte in der Computertechnik wurden Technologien entwickelt, welche Kommunikation in natürlicher Sprache simulieren lassen. So kann die Informationsbeschaffung zwischen einem Menschen und einer Maschine zum Beispiel über natürliche Sprache stattfinden. Diese Technologie wird Chatbot-Technologie genannt.

„Ein Chatbot ist ein Computerprogramm, das Eingaben des Nutzers in natürlicher Sprache verarbeitet und intelligente Antworten erzeugt, welche dann dem Nutzer zurückgesendet werden“ [4]. Die meisten Chatbots sind heutzutage entweder regelbasiert oder auf ML aufgebaut. Die Interaktion zwischen dem Nutzer und dem Programm läuft meistens über eine textbasierte Schnittstelle ab. Hierzu werden Messenger Dienste wie zum Beispiel Facebook, Slack oder Telegram und deren API's verwendet [4].

Es gibt unzählige Einsatzbereiche für diese Art von Technologie. Sie können zum Beispiel im First-Level Support ihren Einsatz finden, bei der Beratung von Menschen oder bei der Informationsbeschaffung. Im letzteren Punkt findet sich diese Arbeit wieder.

2.1.2 Geschichte der Chatbots

Im Jahre 1950 ging Alan Turing der Frage auf den Grund, ob es möglich wäre, dass ein Computerprogramm mit einer Gruppe von Menschen kommuniziert, ohne dass diese merken, dass es sich um einen künstlichen Gesprächspartner handelt. Diese Frage nennt man den sogenannten Turing Test, welcher von vielen Forschern als Grundlage moderner Chatbot-Technologien angesehen wird [5].

Ein erster Ansatz hierzu wurde 1966 von Joseph Weizenbaum veröffentlicht [6]. Der Chatbot, der den Namen ELIZA trägt, wurde von 1964 bis 1966 im Artificial Intelligence Laboratory des MIT entwickelt. Er sollte den Benutzern vortäuschen, dass diese eine Konversation mit einem Menschen führen [7]. Hierfür simulierte er die Tätigkeit eines Psychotherapeuten und gab die Eingabe des Nutzers als Frage formuliert wieder. Eliza

hatte nur eine begrenzte Fähigkeit zu Kommunikation, wurde aber durchaus eine Inspiration für spätere Projekte [8].

Eine weitere Entwicklung in diesem Bereich war PARRY. Dieser Chatbot wurde 1972 von Kenneth Marc Colby, einem Psychiater und Informatiker, veröffentlicht. Er wurde in der psychiatrischen Abteilung von Stanford eingeführt. Vor allem in der Funktionsweise bzw. dem Verhalten unterscheidet dieser sich von ELIZA. Der Chatbot sollte nicht wie ein Arzt reagieren, sondern eher wie ein schizophrener Patient. Er sollte somit auch als Übung für angehende Psychiater verwendet werden [9].

Der erste online Chatbot wurde 1995 vorgestellt. Mit dem Namen ALICE war dieser ein Vorreiter der heutigen Chatbots. Er gewann den Loebner-Preis als „the most human Computer“ in den Jahren 2000, 2001 und 2004. Der Loebner-Preis ist eine Auszeichnung, welche auf dem Turing Test basiert und den menschenähnlichsten Computer sucht. Bei der Entwicklung von ALICE, einem open source Projekt, waren bis zu 500 Entwickler aus der ganzen Welt beteiligt [10]. ALICE basiert auf der Technologie der AIML. AIML ist eine auf XML basierende Markup Language, welche Tag-basiert funktioniert. Diese dient zur Modellierung von Gesprächsmustern [11]. Außerdem folgt der Chatbot dem Prinzip des überwachten Lernens (supervised learning), da aus vorhandenen Konversationen weitere AIML Inhalte generiert werden.

In den 2010er Jahren kamen nach und nach die wohl modernste Form der Chatbots auf den Markt. Die führenden Technologieunternehmen veröffentlichten smarte, sprachgesteuerte Assistenten, welche vor allem das Internet der Dinge revolutionierten. Diese Assistenten sind zum Beispiel Apples Siri [12], Amazons Alexa [13] oder der Google Assistant [14].

2.1.3 Arten von Chatbots

Um Chatbots zu unterscheiden, müssen verschiedene Faktoren betrachtet werden. Adamopoulou und Moussiades definieren hierfür sechs Parameter mit denen Chatbots klassifiziert werden können [15]. Diese werden nachfolgend beschrieben.

Einen der zu klassifizierenden Parameter stellt die Wissensdomäne (knowledge domain) dar. Hierbei wird unterschieden, ob ein Chatbot über mehrere Wissensbereiche reden kann (open domain) oder ob er auf einen Bereich begrenzt (closed domain) ist und keine Informationen über andere Bereiche geben kann.

Ein weiterer Punkt in dem unterschieden werden kann, ist die Dienstleistung, die ein Chatbot anbietet (service provided). Interpersonelle (interpersonal) Chatbots befinden sich im Bereich der Kommunikation und bieten zum Beispiel Dienste wie Buchungen oder FAQ's an. Sie erhalten Informationen von einer Schnittstelle und geben diese an den Benutzer weiter. Die sich im persönlichen Bereich befindenden Programme werden intrapersonelle (intrapersonal) Chatbots genannt. Sie besitzen eine Persönlichkeit und sind Begleiter des Nutzers im Alltag. Vermittler Chatbots (inter-agent) dienen zur Kommunikation zwischen Chatbots, sozusagen als Schnittstelle.

Auch durch die Ziele (goals), welche die Chatbots verfolgen sollen, können sie unterschieden werden. Informative (informative) Chatbots sollen den Benutzer mit Informationen versorgen. Sie suchen diese Informationen nach der Anfrage des Nutzers entweder aus gespeicherten Quellen oder beziehen diese von einer Schnittstelle. Unterhaltungsbasierende (chat-based/conversational) Chatbots sprechen hingegen wie ein echter anderer Mensch. Sie sollen exakt, vor allem auch kontextuell richtig, auf die Eingabe des Benutzers antworten. Aufgabenbasierende (task-based) Chatbots sind auf ihre Aufgabe fixiert und dienen nur dessen Zweck. Sie haben konkrete Aufgaben wie zum Beispiel eine Flugbuchung.

Eine weitere Klassifizierung kann durch die Betrachtung der Art der Verarbeitung des Inputs und Erstellung des Outputs (input processing and response generation method) erfolgen. Die Architektur der regelbasierten (rule-based) Chatbots wurde vor allem in den ersten Ansätzen dieser Technologie verfolgt. Hierbei wird die Antwort des Chatbots auf Basis von vordefinierten Regeln ausgesucht. Antworten sind fest vordefiniert und werden nicht generiert. Die abrufbasierenden (retrieval-based) Modelle bieten mehr Flexibilität. Sie generieren die Antworten aus Abfragen von API's oder anderen Ressourcen. Beim generativen (generative) Ansatz werden die Antworten basierend auf die vorherigen Gesprächsdaten generiert. Dies geschieht durch die Anwendung von Machine Learning und Deep Learning Algorithmen.

Auch der Anteil an menschlicher Hilfe (human-aided), kann bei der Unterscheidung berücksichtigt werden. Hierbei wird analysiert ob und wie weit ein Chatbot durch menschliches Know-how unterstützt wird. Welche Entwicklungsplattform genutzt wurde, kann ebenfalls eine Rolle spielen. Bei einer Open-Source Plattform hat der Entwickler zum Beispiel vollen Eingriff bei der Implementierung. Werden jedoch verschlossene Plattformen (closed platforms) von großen Unternehmen verwendet, haben die Entwickler nur einen kleinen Einblick und können die internen Funktionen der Plattform nicht beeinflussen. Ein Vorteil dieser Plattformen ist aber meist die große Menge an bereitgestellten Testdaten. [15].

Tabelle 2.1: Klassifizierung von Chatbots nach Adamopoulou und Moussiades [15].

Parameter	Klassifizierungsmöglichkeiten
knowledge domain	<ul style="list-style-type: none"> • open domain • closed domain
service provided	<ul style="list-style-type: none"> • interpersonal • intrapersonal • inter-agent
goals	<ul style="list-style-type: none"> • informative • chat-based/conversational • task-based
input processing and response genera-	<ul style="list-style-type: none"> • rule-based

tion method	<ul style="list-style-type: none">• retrieval-based• generative
human-aided	<ul style="list-style-type: none">• level of human support
build	<ul style="list-style-type: none">• open source• closed platform

2.2 Natural Language Processing

Damit ein Chatbot und somit ein Computerprogramm die menschliche Sprache verstehen bzw. verarbeiten kann, muss die Eingabe des Nutzers zuerst bearbeitet werden. Dieser Prozess der Verarbeitung natürlicher Sprache wird Natural Language Processing (NLP) genannt. NLP beschreibt ein Gebiet der Informatik, welches sich mit der menschlichen Kommunikation beschäftigt. Durch die Verwendung von verschiedenen Computertechniken soll hierbei die Analyse und Darstellung menschlicher Sprache durch Maschinen gelingen. Es gibt verschiedene Methoden, um die natürliche Sprache zu verstehen (NLU) oder natürliche Sprache zu erzeugen (NLG). Bei NLP handelt sich um ein großes Forschungsfeld, in dem viele Methoden noch erforscht oder weiterentwickelt werden müssen. So sind Suchmaschinen ein Beispiel für Algorithmen, welche schon sehr gut entwickelt sind. Ein Beispiel an dem noch weitergeforscht werden muss, ist das Themenfeld der Interpretation von Sätzen und deren Bedeutungen, da es hier nur eine begrenzte Anzahl an Algorithmen gibt [16], [17].

2.2.1 Vorbearbeitung der Daten

Wie zu Beginn des Kapitels bereits erwähnt, müssen verschiedene Verfahren durchgeführt werden, damit ein Computer die menschliche Sprache verarbeiten kann. Hierzu ist die Vorbearbeitung (preprocessing) der Daten ein wichtiger Punkt. Es werden dabei verschiedene Schritte durchlaufen, die die Eingabe der Benutzer so bearbeitet, dass verschiedene Verfahren des maschinellen Lernens zur Klassifikation angewendet werden können. Diese Verfahren werden in einem späteren Kapitel genauer beschrieben. Die Vorbearbeitung der Daten kann über eine Reihe von verschiedenen Methoden geschehen [17]. Hierzu gibt es verschiedene Ansätze zur Auswahl und Anordnung dieser Methoden. Nachfolgend wird eine Pipeline auf Basis von Vijayarani et al. beschrieben [18]. Dieser Vorbearbeitungsprozess wird in diesem Projekt verwendet.

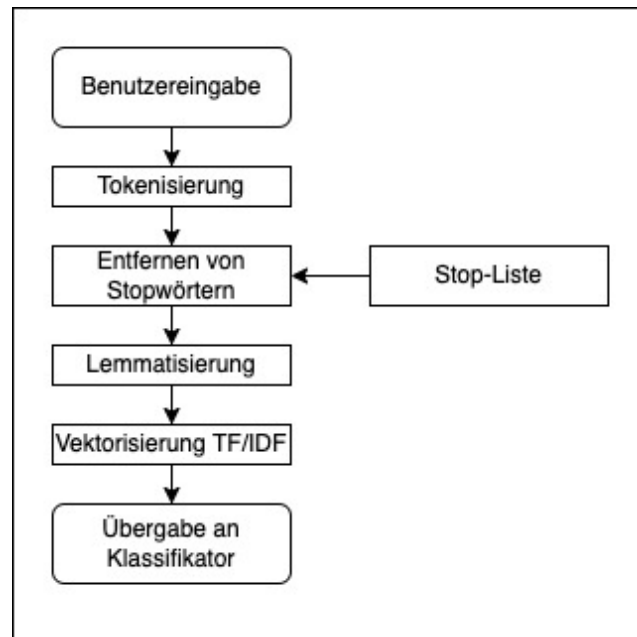


Abbildung 2.1: Eigene Preprocessing Pipeline auf Basis von Vijayarani et al. [18].

Tokenisierung:

Bei der Tokenisierung wird ein Satz oder ein Text in einzelne Elemente wie Wörter, Zahlen oder Satzzeichen zerlegt. Diese einzelnen Elemente werden als Tokens bezeichnet. Zur Weiterverarbeitung wird eine Liste dieser Tokens verwendet. Der Hauptzweck bei diesem Verfahren ist die Identifizierung der Schlüsselwörter [19]. Ein Beispiel der Tokenisierung kann wie folgt aussehen:

Wann habe ich heute Vorlesungen? → [Wann, habe, ich, heute, Vorlesungen, ?]

Entfernen von Stopwörtern:

In einem Text kommen verschiedene Wörter sehr häufig vor. Diese Wörter sind aber im Kontext meist völlig bedeutungslos und tragen zum Inhalt des Textes kaum bei. Ein Beispiel für solche Wörter sind zum Beispiel Verbindungswörter wie „oder“ oder „und“. In den meisten NLP-Bibliotheken gibt es für verschiedene Sprachen eine Liste an solchen Stopwörtern. Über einen Vergleich, ob ein Token in dieser Liste vorhanden ist, können diese dann im Vorbearbeitungsprozess entfernt werden [19].

Lemmatisierung:

Die Lemmatisierung beschreibt die Suche nach der Grundform eines Wortes. Das Verfahren ist eng mit dem Stemming verwandt, unterscheidet sich aber in gewissen Punkten. Bei der Lemmatisierung werden Flexionen und Suffixe von Wörtern entfernt um diese dann in ihr Lemma (Wörterbuchform) umzuwandeln. Beim Stemming werden nur die Endungen abgeschnitten, es resultieren in manchen Fällen also keine richtigen Wörter. Bei der Lemmatisierung werden die Grundformen in ihrer lexikalischen Form

dargestellt. Hierzu werden aber verschiedene Wörterbücher zum Vergleich gebraucht. Dies wirkt sich auf die Geschwindigkeit bei der Verwendung der Lemmatisierung aus. [16].

2.2.2 Vektorisierung (TF-IDF)

Die Vektorisierung ermöglicht Maschinen den Text der natürlichen Sprache zu verstehen und zu verarbeiten. Dies geschieht durch eine Umwandlung der Wörter in eine numerische Darstellung. Es ist somit der wichtigste Schritt, um Textdokumente maschinenlesbar zu machen.

Die am häufigsten verwendete Methode zur Vektorisierung ist die Term Frequency – Inverse Document Frequency Methode. TF-IDF gibt an, wie wichtig ein Wort für ein Dokument in einer Sammlung von Dokumenten ist. Der TF-IDF Wert steigt abhängig von der Häufigkeit des Wortes innerhalb des zu untersuchenden Dokuments. Er wird aber durch die Häufigkeit in der kompletten Sammlung ausgeglichen. Es wird also die Begriffshäufigkeit mit der inversen Dokumentenhäufigkeit multipliziert, wobei die inverse Dokumentenhäufigkeit darstellt wie oft ein Wort im Gesamten vorkommt [20], [21]. Der TF-IDF Wert lässt sich wie folgt berechnen:

TF-IDF:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Wobei t die Terme, d das einzelne Dokument und D eine Sammlung von Dokumenten bezeichnet.

Termfrequenz (tf):

$$tf(t, d) = \frac{\text{Anzahl der Vorkommen eines Terms } t \text{ in einem Dokument } d}{\text{Gesamtzahl der Begriffe im Dokument } d}$$

Inverse Dokumentenhäufigkeit (idf):

$$idf(t, D) = \log \frac{\text{Gesamtanzahl der Dokumente } D}{\text{Anzahl der Dokumente in denen der Begriff } t \text{ vorkommt}}$$

2.3 Machine Learning

2.3.1 Machine Learning Grundlagen

Das Gebiet des maschinellen Lernens (ML) ist ein Teilgebiet der künstlichen Intelligenz. Es stellt ein multidisziplinäres Themengebiet dar, welches viele Forschungsbereiche enthält. Ziel ist das Trainieren von KI-Systemen, damit diese sich selbstlernend an die Umgebung anpassen können [22].

Machine Learning untersucht Algorithmen, die sich mit zunehmender Erfahrung verbessern. Ein System ist also lernfähig, wenn sich bei einer bestimmten Aufgabe T , die Leistung P durch die Erfahrung E verbessert [23]. Es gibt verschiedenste Gebiete, in

denen Machine Learning Algorithmen einen positiven Beitrag leisten können. Beispiele hierfür sind die natürliche Sprachverarbeitung, die Robotik, Computer Vision oder die Kundenzuordnung.

Im Bereich des maschinellen Lernens werden dabei verschiedene Lernansätze verfolgt. Nachfolgend wird eine Auswahl dieser Kategorien erläutert.

Überwachtes Lernen (Supervised Learning):

Beim Ansatz des überwachten Lernens wird eine Reihe von Beispielen mit den richtigen Ausgaben bereitgestellt. Auf Grundlage dieser Trainingsmenge lernt der Algorithmus. Dieses Verhalten wird auch als lernen von Beispielen bezeichnet. Der Algorithmus versucht also Beziehungen und Abhängigkeiten zwischen den Eingabedaten und der richtigen Antwort zu erkennen, um diese dann auf unbekannte Daten anzuwenden und diese zu klassifizieren. Anwendung findet dies zum Beispiel in der Vorhersage (Forecasting) auf Grundlage von vorangegangenen Daten. Ein Beispiel hierfür ist der Produktvorschlag auf Basis des vorherigen Kaufverhaltens eines Kunden. Zur Vorhersage können hierbei Regressions- und Klassifizierungsmethoden verwendet werden [24], [25].

Unüberwachtes Lernen (Unsupervised Learning):

Bei diesem Ansatz sind die Kategorien, in welche die Daten unterteilt werden können, vorher nicht bekannt. Die Trainingsdaten sind somit unbeschriftet. Es wird versucht, nicht identifizierte, bestehende Muster aus den Daten zu erkennen. Es sollen verborgene Strukturen aus den nicht beschrifteten Daten gefunden werden. Das Ziel ist es also Regeln und Muster zu suchen und die Datenpunkte so zu gruppieren, dass Erkenntnisse daraus gewonnen werden können [24], [25].

Halbüberwachtes Lernen (Semi-supervised Learning):

Dieser Lernansatz befindet sich zwischen den beiden zuvor beschriebenen. Beim überwachten Lernen sind die Kategorien, in welche die Daten zu klassifizieren sind, bekannt. Beim unüberwachten Lernen sind diese nicht bekannt. So sind beim halbüberwachten Lernen einige Daten beschriftet, die Mehrzahl der Daten sind aber unbeschriftet. Hierfür kann ein Grund zum Beispiel mangelndes Fachkenntnis über die Daten sein [24], [25].

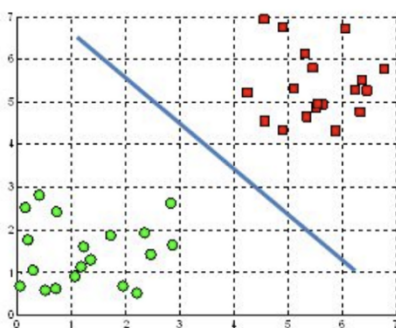
2.3.2 Klassifikationsmodelle

Im Nachfolgenden werden Algorithmen zur Klassifikation beschrieben, welche bei der Entwicklung des Chatbots implementiert und verglichen wurden.

Support Vector Machine (SVM):

Bei der Support Vector Machine handelt es sich um einen Algorithmus zur Klassifikation von Daten. Dieser führt die Einteilung durch, indem er eine n -dimensionale Hyperebene konstruiert. Diese Ebene trennt die Daten in zwei Klassen. Dabei gibt es viele mögliche Hyperebenen, um diese Klassen zu trennen. Ziel ist es die Ebene zu finden, die den größten Abstand zwischen den Datenpunkten der einzelnen Klassen aufweist. Die Dimension der Hyperebene hängt dabei von der Anzahl der Merkmale ab. So ist bei einer Merkmalsanzahl von zwei die Hyperebene zum Beispiel nur eine Linie. Bei drei berücksichtigten Merkmalen wird sie eine zweidimensionale Ebene [26], [27].

A hyperplane in \mathbb{R}^2 is a line



A hyperplane in \mathbb{R}^3 is a plane

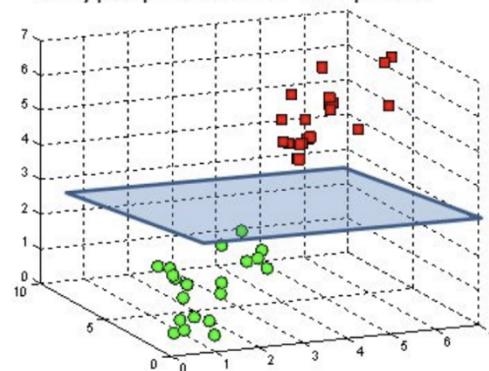


Abbildung 2.2: Hyperebenen im zwei- und dreidimensionalen Merkmalsraum [27].

Naive Bayes (NB):

In diesem Ansatz wird die lineare Klassifizierung verwendet. Hierbei wird vor allem von der Annahme ausgegangen, dass die Attribute unabhängig voneinander sind. Dies ist in der Realität aber meist nicht der Fall. Trotzdem liefert dieser einfache und effiziente Algorithmus meist sehr gute Ergebnisse. Der Algorithmus basiert, wie der Name schon verrät, auf dem Bayes Theorem. Dies ist ein Wahrscheinlichkeitsmodell, welches von Thomas Bayes formuliert wurde. In Zusammenhang mit einer Klassifikation wird untersucht, wie groß die Wahrscheinlichkeit ist, dass ein bestimmtes Objekt auf Basis seiner Merkmale zu einer Klasse x gehört. Es wird hierbei die Wahrscheinlichkeit für jede Klasse berechnet und die wahrscheinlichste ausgewählt [28], [29].

k-Nearest Neighbor (kNN):

k-Nearest Neighbor ist ein weit verbreiteter Klassifikator, welcher auch häufig im Bereich der Textklassifizierung zu finden ist. Vor allem seine Einfachheit und Effizienz stechen dabei hervor. Dieser Algorithmus wird oft als „lazy learner“ bezeichnet, da die Modellbildung nicht während der Trainingsphase, sondern erst bei Anfrage geschieht. Um ein unbekanntes Objekt zu klassifizieren, wird das Objekt in der Trainingsmenge

eingesortet. Dann wird die Vorhersage auf Basis der Klassenbezeichnungen der k nächsten Nachbarn ermittelt [30], [31].

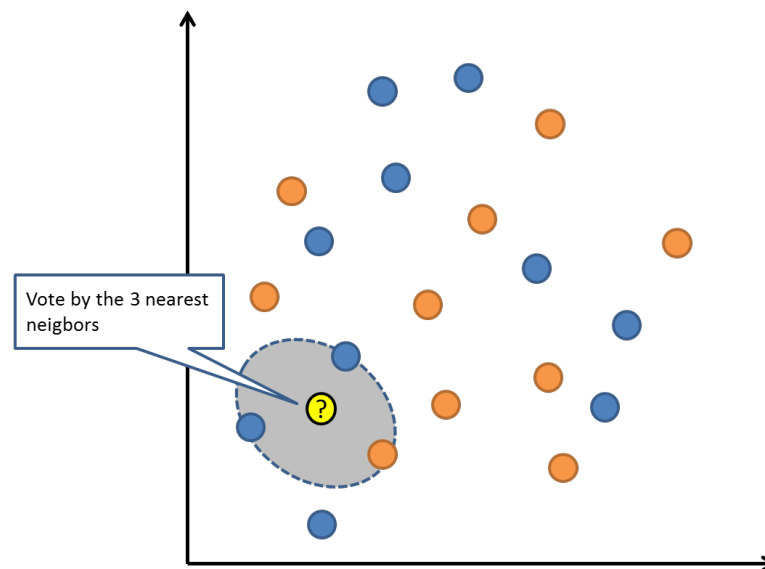


Abbildung 2.3: Darstellung des KNN-Algorithmus mit $k = 3$ [31].

Decision Tree:

Beim Decision Tree Algorithmus handelt es sich um ein Klassifizierungsverfahren unter Verwendung eines Entscheidungsbaums. Hierbei werden die Daten kontinuierlich nach bestimmten Parametern aufgeteilt. Es ergibt sich daraus eine baumähnliche Struktur. Dieser Baum besteht aus sogenannten Knoten, an denen der Wert eines bestimmten Attributs des zu klassifizierenden Objekts überprüft wird. Eine Verzweigung entspricht dem Ergebniss des Tests, welcher am Knoten durchgeführt wurde. Sie verbindet Knoten mit den nächsten Knoten oder Blättern. Die Klassifizierungsergebnisse werden durch Blattknoten dargestellt, welche für die einzelnen Klassen stehen. Bei einer Klassifizierung wird bei der Wurzel des Baums begonnen und sich so lange durch den Baum bewegt, bis ein Blatt erreicht wird [32].

2.3.3 Evaluation der Klassifikationsmodelle

Nach der Implementierung und dem Training der Klassifikationsmodelle müssen diese getestet werden. Hierbei wird getestet, wie gut das Modell mit ungesehenen Daten umgehen kann. Dies kann über verschiedene Leistungsmetriken geschehen. Über diese Metriken können die verschiedenen Modelle miteinander verglichen werden, um zu sehen, welches Modell am besten zum Anwendungsfall passt [33]. Nachfolgend werden verschiedene Kennzahlen zur Evaluation der Leistung des Modells genannt und erläutert.

Kontingenztabelle:

Die Kontingenztabelle (Confusion Matrix) stellt die primäre Quelle für die Leistungsmessung von Klassifikationsmodellen dar. Die Kreuztabelle ist die Basis einer Vielzahl an anderen Kennzahlen. Hierbei werden die vorhergesagten Werte mit den tatsächlichen Werten gegenübergestellt. Klassen werden in den Zeilen in gleicher Reihenfolge aufgeführt wie in den Spalten. Dies hat zur Folge, dass sich die richtig klassifizierten Objekte auf einer Diagonalen von links oben nach rechts unten befinden [34].

		True Class	
		Positive	Negative
Predicted Class	Positive	True Positive Count (TP)	False Positive Count (FP)
	Negative	False Negative Count (FN)	True Negative Count (TN)

Abbildung 2.4: Einfache Kontingenztabelle nach Olson und Delen [33].

Precision:

Die Präzision ist die Anzahl an richtig positiv klassifizierten Elementen geteilt durch die Gesamtzahl an positiv vorhergesagten Werten. Dieser Wert gibt also an wie sehr man einem Modell vertrauen kann, wenn es ein Element als positiv vorhersagt [34].

$$Precision = \frac{TP}{TP + FP}$$

Recall:

Bei der Berechnung des Recall-Werts werden die richtig positiv klassifizierten Elemente durch die Gesamtzahl der eigentlich positiven Werte geteilt. Es zeigt also die Vorhersagegenauigkeit für die positive Klasse [34].

$$Recall = \frac{TP}{TP + FN}$$

Accuracy:

Die Genauigkeit wird ebenfalls auf Basis der Kontingenztabelle berechnet. Hierfür steht die Anzahl aller richtig klassifizierten Elemente im Zähler und die Gesamtzahl der klassifizierten Elemente im Nenner. Dieser Wert beschreibt also die Wahrscheinlichkeit, dass ein unbekanntes Objekt richtig klassifiziert wird [34].

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

F1-Score:

Der F1-Wert ermittelt das harmonische Mittel zwischen Precision und Recall. Es ist ein gewichteter Durchschnitt zwischen diesen Werten. Es gibt also einen Überblick über die Leistung des Modells [34].

$$F1 = 2 * \left(\frac{Precision * Recall}{Precision + Recall} \right)$$

3 Anforderungsanalyse

Um den Nutzen des zu entwickelnden Chatbots für die Studierenden zu maximieren, wurde vor der Entwicklung eine Anforderungsanalyse durchgeführt. Hierbei stand die Auseinandersetzung mit dem Ziel des Bots und der Zielgruppe im Mittelpunkt. Die Zielgruppe sind Studierende des Studiengangs Wirtschaftsinformatik an der Hochschule Aalen. Ihnen soll durch den Chatbot die Möglichkeit geboten werden, Informationen über das Studium zu erlangen. Das Ziel ist die Maximierung der Effektivität bei der Informationsbeschaffung durch Studierende. Diese Informationsbeschaffung läuft bislang über die manuelle Suche auf der Webseite der Hochschule Aalen ab und ist, vor allem für Studierende in jüngeren Semestern, meist mühsam und zeitaufwendig. Der Chatbot soll Informationen der Studiengangsseite abdecken, welche im Laufe des Studiums häufig gebraucht werden.

Folgende Informationen sollen durch den Chatbot abgefragt werden können:

- Allgemeine Informationen zum Studienprozess wie die Anerkennung von Leistungen, das Studium Generale oder das Praxissemester.
- Informationen über die Vorlesungen und Termine. Hierbei soll der Vorlesungsplan durch den Studierenden konfiguriert werden können. Anschließend sollen die Termine der gewählten Veranstaltungen abgefragt werden können.
- Dokumente, welche im Laufe des Studiums benötigt werden. Beispiele hierfür sind das Modulhandbuch, die Studien- und Prüfungsordnung oder ein Antrag auf Verlängerung der Bachelorarbeit.
- Aktuelle Informationen vom Schwarzen Brett.

Um den Studierenden die Informationsbeschaffung so einfach wie möglich zu machen, soll der Chatbot in einem Messengerdienst auf mobilen Endgeräten verfügbar sein. Außerdem soll der Chatbot in einer modernen objektorientierten Programmiersprache entwickelt werden. Die Konzeption und Entwicklung dieses Chatbots wird im Kapitel 4 beschrieben.

4 Konzeption und Entwicklung des Prototyps

In diesem Kapitel wird die Konzeption und Entwicklung des Prototyps beschrieben. Die verwendeten Technologien werden vorgestellt und die Vorgehensweise bei der Entwicklung wird beschrieben. Außerdem wird der finale Prototyp eingeordnet und klassifiziert.

4.1 Planung und Konzeption

Vor der Entwicklung des Prototyps wurde dieser geplant und konzipiert. Um mit dem Benutzer interagieren zu können muss ein Chatbot auf verschiedene Dienste und Technologien zurückgreifen. So gibt es verschiedene Ansätze wie ein Chatbot konzipiert werden kann. Meshram et al. stellen ein Konzept vor, wie ein Chatbot grundsätzlich aufgebaut werden kann. Dabei kommuniziert der Benutzer über eine Schnittstelle mit dem Chatbotprogramm. Nachdem der Benutzer eine Frage über die Schnittstelle versendet hat, wird diese über eine NLP-Software analysiert. Im Nachgang werden die für die Antwort benötigten Daten aus der Datenbank abgefragt und eine Antwort wird generiert. Die Antwort wird dann wieder über die Schnittstelle an den Benutzer gesendet [35].

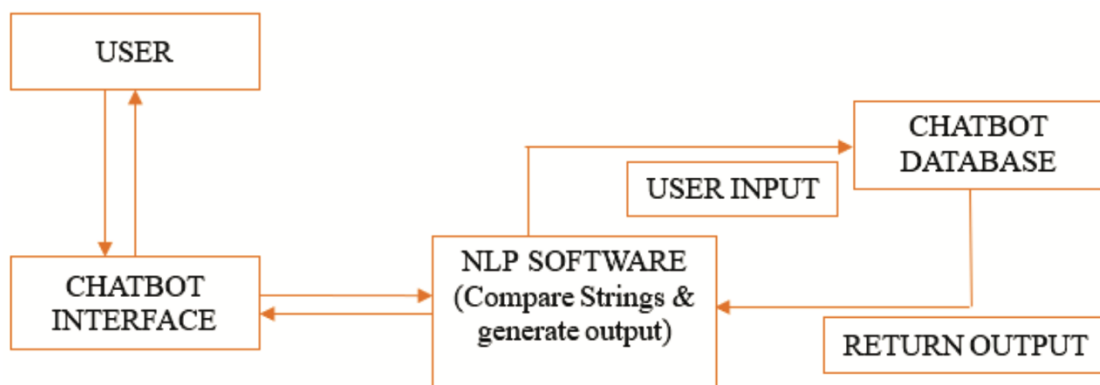


Abbildung 4.1: Chatbot Design nach Meshram et al. [35].

Aufbauend auf Meshram et al., wurde ein Konzept für die Entwicklung erstellt. Es wurden verschiedene Technologien ausgesucht, die zur Entwicklung benutzt wurden. Außerdem wurden die Bibliotheken, das Datenbankmanagement-System und die Benutzerschnittstelle recherchiert und ausgesucht. Nachfolgend wird dieses Konzept dargestellt und beschrieben.

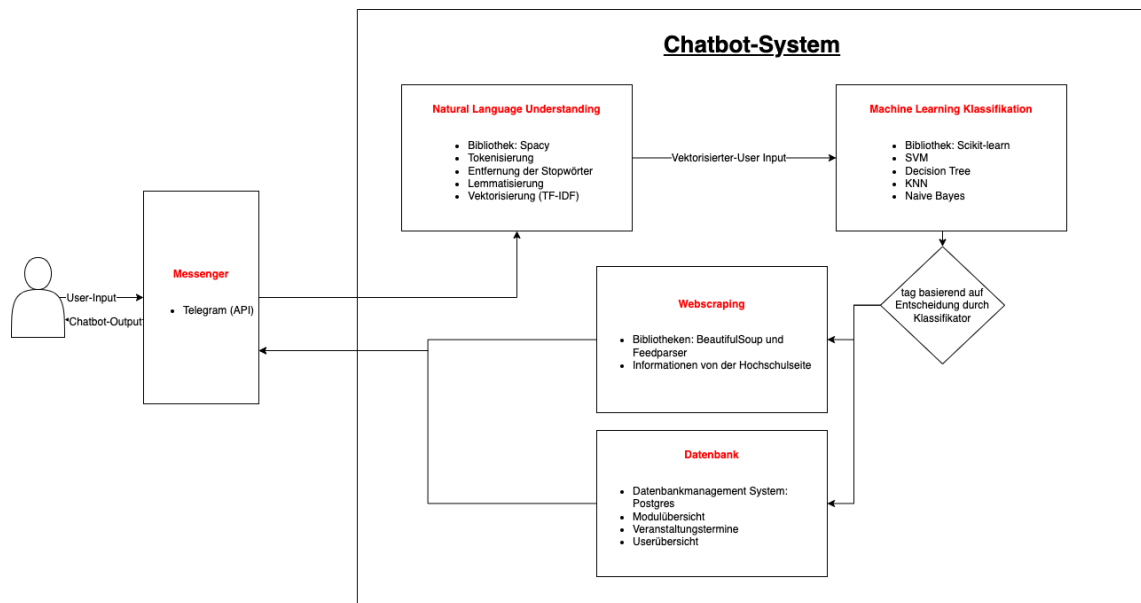


Abbildung 4.2: Eigenes Konzept des Chatbots.

Das entwickelte Konzept besteht hierbei aus zwei Teilen. Zum einen aus der Chatbot-Applikation, welche mit der Programmiersprache Python realisiert werden soll. Zum anderen aus dem Messengerdienst, der die Schnittstelle zwischen Benutzer und Chatbot darstellt. Das Konzept beinhaltet mehrere Bausteine, die geplant werden mussten. Nachfolgend werden diese erläutert.

Messenger:

Um den Studierenden die Informationsbeschaffung zu vereinfachen, sollte der Chatbot über einen Messenger erreichbar sein. Durch die Entwicklung der Smartphones wird heutzutage fast jeder Studierende von einem mobilen Endgerät durch den Tag begleitet. Kommunikation mit anderen Studierenden läuft hierbei meist über einen Messengerdienst. Als Messenger wurde Telegram ausgewählt, da dieser kostenlos zur Verfügung steht und eine einfach zu benutzende API besitzt. Außerdem gibt es zahlreiche Bibliotheken in verschiedenen Programmiersprachen, die die Kommunikation mit dieser API vereinfachen. Telegram stellt somit die Schnittstelle zwischen Benutzer und dem Programm dar.

Natural Language Understanding:

Um die Eingabe des Benutzers auf Seiten der Chatbot-Applikation verarbeiten zu können, muss die natürliche Sprache verarbeitet werden. Diese Verarbeitung geschieht über verschiedene Methoden, welche im Kapitel Grundlagen schon detaillierter behandelt wurden. Zur Verarbeitung in Python wurde die Bibliothek Spacy verwendet.

Klassifikation durch Machine Learning Algorithmen:

Um den Bot entscheiden zu lassen welche Information der Benutzer benötigt, werden ML-Algorithmen zur Klassifikation benötigt. Um diese Algorithmen einzusetzen, wurde die Bibliothek Scikit-Learn benutzt. Es wurden verschiedene Algorithmen zur Klassifikation getestet und miteinander verglichen.

Webscraping und Datenbank:

Nachdem die Eingabe des Nutzers klassifiziert und einem Thema zugeordnet wurde, müssen die erfragten Informationen bereitgestellt werden. Dies soll über zwei Wege geschehen. Zum einen soll eine Datenbank mit Informationen über Studierende, Vorlesungen und den gewählten Modulen aufgesetzt werden. Als Datenbankmanagement System wurde hierbei Postgres ausgewählt. Die allgemeinen und aktuellen Informationen zum Studium, sowie die für das Studium benötigten Dokumente werden über Webscraping beschaffen. Hierzu wurden die Bibliotheken BeautifulSoup und Feedparser ausgewählt.

4.2 Verwendete Technologien und Bibliotheken

Da der Chatbot vielseitige Aufgaben erledigen soll, wurden verschiedene Technologien und Bibliotheken bei der Entwicklung angewendet. Grundlage ist die Programmiersprache Python zusammen mit einer Postgres-Datenbank. Nachfolgend werden die verwendeten Technologien und Bibliothek kurz erklärt.

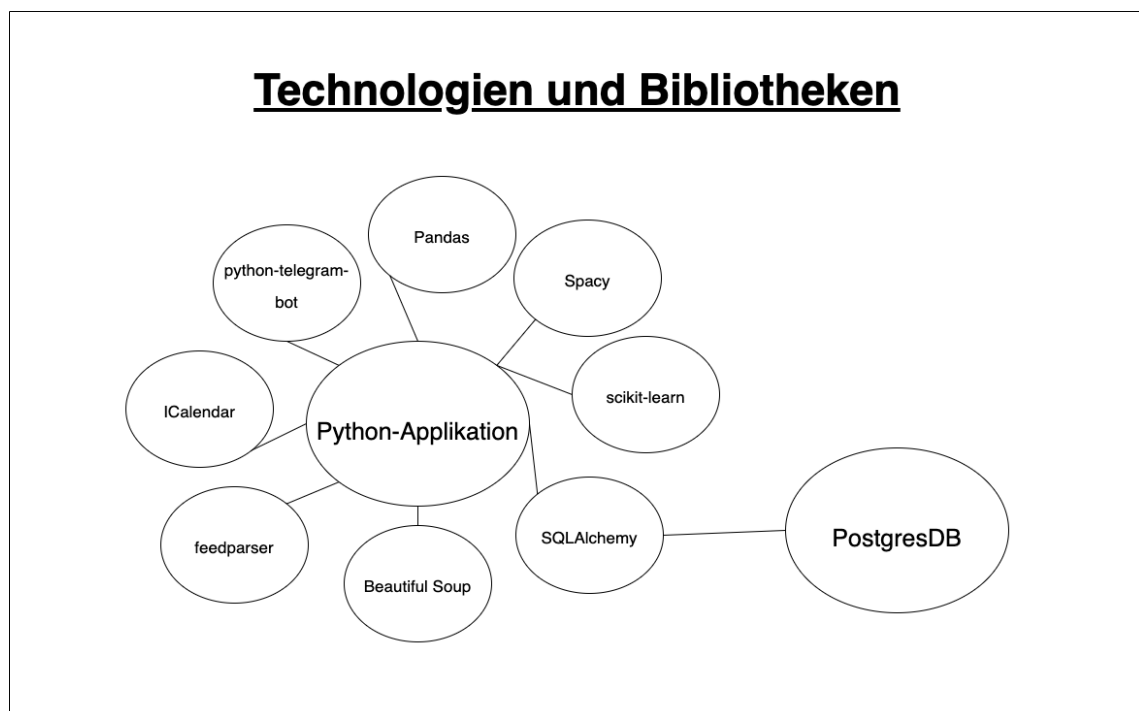


Abbildung 4.3: Darstellung der verwendeten Technologien und Bibliotheken.

Python ist eine objektorientierte und interpretierte Programmiersprache. Sie kombiniert Leistungsfähigkeit mit klarer und einfacher Syntax. Ziel ist die Steigerung der Effektivität bei der Entwicklung durch bessere Lesbarkeit. Sie ist vor allem für die schnelle Anwendungsentwicklung oder als Skriptsprache geeignet. Außerdem können bestehende Systeme einfach durch Python verbunden werden. Sie ist modular aufgebaut und be-

sitzt viele Bibliotheken, die die Sprache erweitern oder als Schnittstelle zu anderen Systemen funktionieren [36].

PostgreSQL ist ein objekt-relationales Datenbanksystem, das die SQL-Sprache um mehrere Funktionen erweitert. Dieses Open-Source System speichert Datenmengen sicher und skalierbar. Seine Ursprünge hat es 1986 als Teil des Postgres-Projekts an der University of California in Berkeley [37].

SQLAlchemy ist eine Python Bibliothek bestehend aus dem Python SQL-Toolkit und dem Object Relational Mapper (ORM). Sie bietet den Entwicklern den Vorteil, SQL vollumfänglich in ihren Programmen zu nutzen. Außerdem bietet sie vollständigen Datenbankzugriff, angepasst an die Programmiersprache Python. Der ORM bietet dabei die Möglichkeit Klassen auf die Datenbank abzubilden [38].

ICalender ist eine Python Bibliothek, welche die Möglichkeit bietet, ICalendar Dateien zu parsen, verarbeiten und generieren [39].

Die Python-telegram-bot Bibliothek ist eine Schnittstelle zur einfachen Nutzung der Telegram API in Python. Zusätzlich zur API-Implementierung bietet die Bibliothek Klassen an, die die Entwicklung eines Telegram-Bots vereinfachen sollen [40].

Pandas ist ein leistungsstarkes Open-Source Tool zur Datenanalyse und -manipulation. Das wesentliche Feature ist hierbei das DataFrame-Objekt, welches zur Datenbereitstellung und -transformation geeignet ist. Ziel des Projekts ist es, die Standardbibliothek für Datenanalyse in Python zu werden. 2008 wurde das Projekt ins Leben gerufen, bevor es 2009 als Open-Source Projekt veröffentlicht wurde [41].

Feedparser ist eine Bibliothek zum Herunterladen und Parsen von syndizierten Feeds wie zum Beispiel RSS. Hierbei werden Feeddaten von einer URL heruntergeladen und verarbeitet [42].

Beautiful Soup kann zum Auslesen und Verarbeiten von HTML- und XML-Dateien benutzt werden. Die Bibliothek ermöglicht das einfache Navigieren durch die Dateipfade und das simple Extrahieren der benötigten Daten [43].

Scikit-learn ist eine plattformunabhängige Bibliothek für das maschinelle Lernen mit Python. Sie ist unter der 3-Klausel-BSD lizenzfrei auf Github verfügbar. Es werden hierbei verschiedene Algorithmen für das maschinelle Lernen zur Verfügung gestellt. So zum Beispiel Regressions-, Clustering- oder Klassifikationsalgorithmen. Die erste Idee des Projekts entstand durch David Cournapeau im Google Summer of Code Projekt. Das erste öffentliche Release war dann 2010 [44].

Durch die freie, quelloffene Bibliothek Spacy wird die natürliche Sprachverarbeitung ermöglicht. Sie unterstützt bei der Entwicklung von Anwendungen in denen Text verarbeitet werden soll und kann zum Beispiel zur Informationsextraktion oder zur Vorbearbeitung benutzt werden. Dabei unterstützt Spacy mehr als 66 verschiedene Sprachen [45].

4.3 Entwicklung des Prototyps

Wie in den vorherigen Unterkapiteln bereits erwähnt, wurden bei der Entwicklung des Prototyps verschiedene Technologien und Bibliotheken auf Basis der Programmiersprache Python benutzt. Nachfolgend wird der Verlauf der Entwicklung des Chatbots beschrieben. Es wird auf die verwendete Datenbank und deren Datenbankstruktur eingegangen. Die Datenansammlung sowie die Verbindung zur Telegram-API werden beschrieben. Außerdem wird die Textverarbeitung und den Einsatz der Machine Learning Algorithmen beschrieben.

4.3.1 Datenbank und Datenbankstruktur

Zur Verwaltung der Informationen über die Benutzer, sowie die Vorlesungen und deren Verknüpfung, wurde PostgreSQL in Verbindung mit SQLAlchemy genutzt. Es wurde eine Postres-Datenbank mit dem Namen WI_HSAA_BOT erstellt und initiiert. Durch die Python Bibliothek SQLAlchemy wurde es ermöglicht, Klassen direkt in der Datenbank abzubilden. Somit entspricht die Klassenstruktur der Python Applikation der der Postgres-Datenbankstruktur.

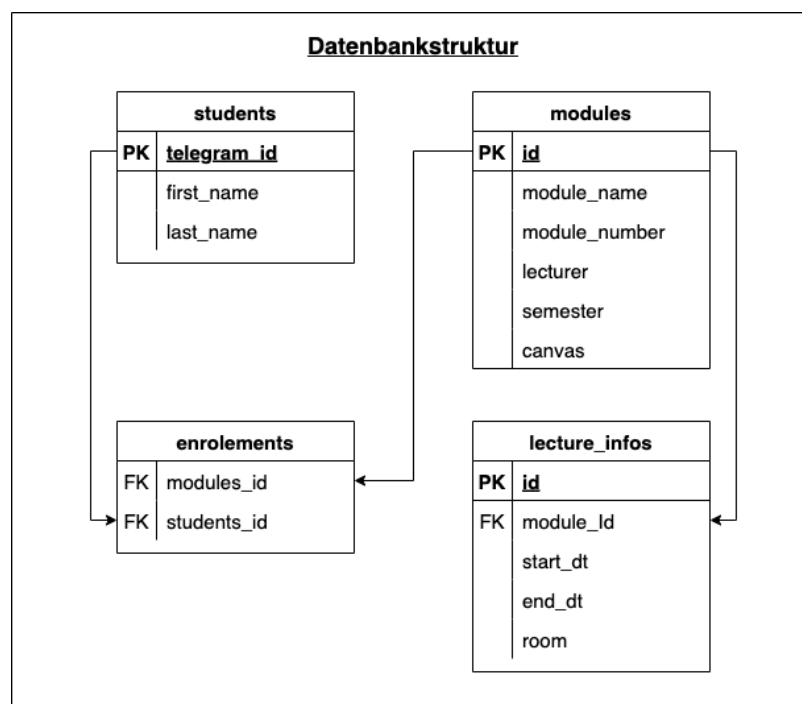


Abbildung 4.4: Darstellung der Datenbankstruktur.

Wie in Abbildung 4.4 dargestellt, werden in der Datenbank Informationen über die Studierenden, die Module und ihre Fächerwahl gespeichert. Die Studenten werden dabei mit ihrer Telegram-Id als Primary Key und ihrem Vor- sowie Nachnamen abgespeichert. Die Module werden mit einer eindeutigen Id und den Informationen zum jeweiligen Modul abgelegt. Die genauen Vorlesungstermine werden in der Tabelle lecture_infos abgespeichert, welche über einen Foreign Key auf das verwandte Modul

zeigt. Die Fächerwahl der einzelnen Studierenden wird über die Tabelle `enrolements` gespeichert. Hier werden die Studierenden mit den gewählten Modulen über die jeweiligen Id's verknüpft.

4.3.2 Datenerhebung

Um die Studierenden mit Informationen über das Studium versorgen zu können, musste eine Datengrundlage geschaffen werden. Für diesen Prototyp wurden deshalb die Informationen von der Hochschulwebseite über Webscraping erfasst, die aktuellen Informationen über ein RSS-Feed abgeholt und die Vorlesungsinformationen über ICal-Dateien extrahiert und in der Datenbank gespeichert.

Vorlesungsdaten:

Um an die Informationen zu den aktuell angebotenen Vorlesungen zu gelangen, mussten die Daten aus Starplan extrahiert werden. Starplan ist eine Software, über welche die Hochschule Aalen die Stundenpläne zur Verfügung stellt. Da Starplan leider keine öffentliche API zur Verfügung stellt, musste eine alternative Lösung gefunden werden. Es wurden die aktuellen Stundenpläne der Semester zwei, vier und sechs manuell als ICal-Datei heruntergeladen. Diese wurden dann per Python-Skript verarbeitet.

Dabei wurde die Bibliothek `ICalendar` verwendet. Zuerst wurde eine Funktion entwickelt, die aus den ICal-Dateien die verschiedenen Module, die in diesem Semester angeboten wurden, extrahiert. Hierfür wurde der Name, die Modulnummer, der Dozent, das zugehörige Semester und der Canvas-Link aus der ICal-Datei entnommen und in einem Dictionary gespeichert. Nach der gleichen Methode wurden dann die einzelnen Vorlesungsdaten extrahiert und verarbeitet. Es wurden der Name des Moduls, das Startdatum und Enddatum der Veranstaltung und der Raum abgelegt. Nachdem diese Informationen aus den ICal-Dateien entnommen wurden, wurden sie in der Datenbank gespeichert. Hierfür wurden die Module mit den jeweiligen Informationen in der `modules`-Tabelle gespeichert. Die einzelnen Veranstaltungstermine wurden dann in der `lecture_info`-Tabelle, unter Verwendung der Id des zugehörigen Moduls als Foreign Key, abgelegt. Listing 1 zeigt diese Funktionen zur Extrahierung und Verarbeitung der Daten aus den ICal-Dateien.

Dokumente und Informationen der Studiengangsseite:

Um die Dokumente und Informationen der Studiengangsseite bereitzustellen, müssen diese aus dem Internet geladen werden. Da diese Informationen nicht statisch sind und in regelmäßigen Abständen aktualisiert werden können, werden diese Informationen nicht in der Datenbank gespeichert. Sie werden nach jedem Aufruf neu heruntergeladen. Dies geschieht über die Python-Bibliothek `Beautiful Soup`. Hierbei wird die HTML-Datei der Studiengangsseite über einen URL-Request heruntergeladen und dann verarbeitet. Es wird in der Datei nach den passenden Schlagwörtern gesucht und die benötigten Informationen werden extrahiert. Anschließend werden Sie in einem String formatiert und für die Ausgabe vorbereitet (Listing 2).

Schwarzes Brett:

Um die Informationen des Reiters „Aktuelles“ bzw. vom schwarzen Brett zu erhalten. Wird bei jeder Anfrage das RSS-Feed des schwarzen Bretts abgefragt. Die Bibliothek Feedparser wird hierfür genutzt. In Listing 3 ist zusehen, dass das RSS-Feed abgefragt, geparsed und in einem String zur Verfügung gestellt wird. Außerdem wird der Link zum schwarzen Brett für genauere Recherche hinzugefügt.

4.3.3 Verbindung zur Telegram-API

Um den Chatbot über Telegram benutzen zu können, muss dort ein Bot initiiert werden. Dies geschieht über den sogenannten BotFather. Über dieses Tool in Telegram wird es Entwicklern ermöglicht Bots zu erstellen. Diese sind spezielle Konten, welche keine zusätzlichen Telefonnummern benötigen. Sie dienen als Schnittstelle zwischen der Chatplattform und dem Code auf dem Server [46].

Nachdem der Bot bei Telegram initiiert wurde, wird ein Token zur Verfügung gestellt. Dieser Token ist der Schlüssel, um sich mit der HTTP API von Telegram zu verbinden. Um die Verbindung zur API einfach zu halten, wurde die Python Bibliothek `python-telegram-bot` verwendet. Diese Bibliothek ermöglicht die einfache Verbindung zur Telegram API. Außerdem besitzt sie verschiedene vorgefertigte Klassen, was die Entwicklung vereinfacht. Um auf die Nachrichten des Benutzers reagieren zu können, können verschiedene Command- und Message-Handler aufgerufen werden. Command Handler reagieren auf Befehle des Benutzers, welche mit einem „/“ aufgerufen werden. Message Handler reagieren auf sonstige Nachrichten des Benutzers. Die Handler rufen dann bei Eingabe der entsprechenden Befehle bzw. Nachrichten die hinterlegten Funktionen auf. Die Initiierung dieser Handler ist in Listing 5 zusehen. Nachfolgend werden die Funktionen beschrieben, die durch die Handler aufgerufen werden können.

Um den Chat mit dem Bot erstmalig zu starten, wird über den Befehl `/start` die Startfunktion aufgerufen. Diese Funktion wird in Listing 6 gezeigt. Sie gibt eine Begrüßung zurück, in der erste Instruktionen über die Benutzung des Bots genannt werden. Außerdem wird in der Funktion überprüft, ob der Benutzer bereits in der Datenbank registriert ist. Falls nicht wird dieser neu mit seiner Telegram Id und seinem Vor- sowie Nachnamen angelegt.

Mit `/help` kann der Handler angetriggert werden, welcher die Hilfe-Funktion auslöst. In dieser Funktion wird ein String zurückgegeben, der die einzelnen Funktionen des Bots und dessen Bedienung erläutert. Die Funktion ist in Listing 7 zusehen.

Eine Übersicht über die wählbaren Module kann über `/modules` angezeigt werden. Wie in Listing 8 gezeigt wird, werden dabei alle Module aus der Datenbank ausgelesen und zusammen mit der jeweiligen Id ausgegeben. Außerdem werden alle Module, die bereits vom Benutzer gewählt wurden, ausgelesen und angezeigt.

Die Modulwahl und das Entfernen eines Moduls geschehen über die Funktionen, welche in Listing 9 und 10 dargestellt sind. Mit `/add` und `/delete` werden diese entspre-

chenden Funktionen ausgelöst. Den Befehlen muss außerdem die Id des gewünschten Moduls angehängt werden, damit die Funktionen ausgeführt werden können. Wenn ein Benutzer ein neues Fach hinzufügen will, wird zuerst überprüft, ob dieser dieses Modul bereits belegt hat. Falls dies nicht der Fall ist, wird anschließend die Benutzer Id mit der Id des Moduls verknüpft. Beim Entfernen des Moduls läuft dieser Prozess genau entgegen ab. Zuerst wird überprüft, ob das Modul hinterlegt ist. Anschließend wird die Beziehung von Benutzer und Modul aufgehoben.

Alle Nachrichten, die nicht über einen Befehl aufgerufen werden, werden über die Echo-Funktion behandelt. Diese Funktion (Listing 11) nimmt die Benutzereingabe entgegen und übergibt sie an eine Support Vector Machine. Diese SVM entscheidet dann auf Basis eines gelernten Trainingssets, welche Information der Benutzer erlangen möchte. Anschließend werden die benötigten Informationen aufgerufen und an den Benutzer zurückgegeben.

4.3.4 Textvorverarbeitung

Damit die richtigen Antworten ausgegeben werden können, muss die Eingabe des Benutzers vorverarbeitet werden. Dies geschieht mit Hilfe verschiedener NLP Techniken. Um diese in Python zu nutzen, wird die Bibliothek Spacy verwendet. Hierbei wird zuerst der Korpus der deutschen Sprache heruntergeladen und implementiert. Anschließend wird die Benutzereingabe durch die Tokenisierung in einzelne Tokens geteilt. Nach diesem Vorgang werden die einzelnen Tokens auf Stopwörter und Satzzeichen überprüft und gegebenenfalls entfernt. Zum Abschluss werden die einzelnen Tokens durch eine Lemmatisierung in ihre Grundform gebracht. Dies geschieht auf Basis des zuvor implementierten Korpus. Diese Funktion ist in Listing 12 zu sehen. Nachdem der Text verarbeitet wurde, wird er durch die TF-IDF Vektorisierung zu einem Vektor transformiert. Dies geschieht mit Hilfe des TF-IDF Vectorizer der Scikit-learn Bibliothek. Wenn die Benutzereingabe vorverarbeitet und zu einem Vektor transferiert wurde, wird dieser Vektor anschließend an einen Machine Learning Klassifikator übergeben.

4.3.5 Implementierung der ML-Algorithmen

Um die Eingaben des Benutzers korrekt verarbeiten zu können, muss entschieden werden über welches Thema der Nutzer informiert werden möchte. Um diese Entscheidungen treffen zu können, wurden Machine Learning Algorithmen zur Klassifikation verwendet. Diese wurden über die Bibliothek Scikit-Learn implementiert. Hierbei wurden verschiedene Algorithmen getestet und über verschiedene Metriken miteinander verglichen. Getestet wurden eine Support Vector Machine, der kNN-Algorithmus, der Naive Bayes Algorithmus und ein Decision Tree Algorithmus.

Damit die Klassifikationsalgorithmen die Eingaben richtig zuordnen können, müssen sie auf Basis einer Datengrundlage trainiert werden. Diese Datengrundlage wurde in einer CSV-Datei gespeichert. Für jeden Themenbereich wurde ein Tag, also eine Bezeichnung definiert. Diese können als Label der Klassen, in welche die Eingaben ein-

geordnet werden können, angesehen werden. Anschließend wurden verschiedene Formulierungen von Fragen (pattern) diesen Tags zugeordnet. Ein Ausschnitt des Datensatzes ist in Listing 13 zusehen.

Damit die Algorithmen zur Klassifizierung aus den Trainingsdaten lernen können. Müssen die Textdaten zuerst vorbereitet werden. Wie in Listing 14 zu sehen ist, werden diese hierfür zuerst aus der CSV-Datei extrahiert und verarbeitet. Die einzelnen Tags werden in einer Liste abgespeichert. Alle Fragen dazu werden zuerst über die Vorverarbeitungsfunktion bearbeitet und dann in einer separaten Liste abgelegt. Anschließend werden die Fragen in eine Vektorform gebracht. Die Tags werden mithilfe eines Label Encoders encodiert. Anschließend wird die aufbereitete Datengrundlage zufällig in Trainings- und Testmenge unterteilt.

Nachdem die Trainings- und Testmenge vorhanden ist, können die Klassifikationsalgorithmen implementiert und verglichen werden. Um diese miteinander vergleichen zu können, wurden die verschiedenen Algorithmen implementiert und mit der Trainingsmenge trainiert. Anschließend wurden die trainierten Klassifikatoren auf die Testmenge angewendet und die Genauigkeit und der F1-Wert ermittelt. Über diese Werte wurden sie dann miteinander verglichen (Listing 15).

Tabelle 4.1: Vergleich der Machine Learning Klassifikatoren.

Klassifikator	Accuracy	F1-Score
SVM	0,8	0,77
KNN	0,68	0,66
Naive Bayes	0,78	0,76
Decision Tree	0,7	0,68

Wie in Tabelle 4.1 zusehen, schnitt bei diesem Vergleich die Support Vector Machine am besten ab. Mit einer Genauigkeit von 0,8 und einem F1-Wert von 0,77 konnte Sie die Daten am zuverlässigsten klassifizieren. Nur knapp dahinter, reihte sich der Naive Bayes Algorithmus ein. Der KNN und Decision Tree Algorithmus konnten dagegen eher weniger gute Ergebnisse erzielen.

Nachdem die Entscheidung auf Basis der zwei Metriken auf die SVM fiel, wurde eine Funktion entwickelt, die die Benutzereingabe klassifiziert. Diese ist in Listing 16 zusehen. Der Message Handler ruft die Funktion auf und übergibt hierbei die Benutzereingabe. Anschließend wird die Eingabe über die NLP-Funktionen vorverarbeitet und in eine Vektorform gebracht. Zuletzt wird die aufbereitete Eingabe klassifiziert und das vorhergesagte Label bzw. Tag wird zurückgegeben. Auf Basis dieses Labels werden dann die benötigten Informationen gesammelt und an den Benutzer zurückgegeben.

4.3.6 Historie der Benutzerinteraktion

Um die Klassifikation genauer werden zu lassen, muss eine größere Datengrundlage geschaffen werden. Damit ein System immer weiter lernen kann müssen deshalb die

richtig klassifizierten Fragen gespeichert und zum Trainingsset hinzugefügt werden. Dies geschieht bei diesem Chatbot über eine Funktion, die die Historie der Benutzerinteraktion in einer separaten CSV-Datei speichert (Listing 17). In gewissen Zeitintervallen sollte dann durch eine Person geprüft werden, welche Fragen richtig zugeordnet worden sind und welche nicht. Alle richtig klassifizierten Fragen werden dann der Datengrundlage und damit dem Trainings- und Testset hinzugefügt.

4.4 Einordnung des Prototyps

Wie in Kapitel 2.1.3 bereits beschrieben wurde, können Chatbots durch verschiedene Parameter klassifiziert werden. Nachfolgend wird der in dieser Abschlussarbeit entwickelte Chatbot eingeordnet und klassifiziert.

Tabelle 4.2: Einordnung des Chatbots nach Adamopoulou und Moussiades [15].

Parameter	Klassifizierungsmöglichkeiten
knowledge domain	<ul style="list-style-type: none"> • closed domain
service provided	<ul style="list-style-type: none"> • interpersonal
goals	<ul style="list-style-type: none"> • informative
input processing and response generation method	<ul style="list-style-type: none"> • retrieval-based
human-aided	<ul style="list-style-type: none"> • Mensch entscheidet, ob korrekt klassifiziert wurde
build	<ul style="list-style-type: none"> • open source

Die Wissensdomäne (knowledge domain) ist einer der ersten Parameter, über den ein Chatbot klassifiziert werden kann. Der Wissensbereich, des in dieser Abschlussarbeit entwickelten Chatbots, ist geschlossen (closed domain). Der Chatbot kann zwar mehrere, unterschiedliche Informationen zum Studium und zum Studiengang Wirtschaftsinformatik liefern. Ist aber klar auf den Bereich der Informationsbeschaffung zum Studiengang Wirtschaftsinformatik begrenzt.

Ein weiterer Punkt bei der Einordnung von Chatbots, ist die angebotene Dienstleistung (service provided). Dieser Chatbot ist den interpersonellen (interpersonal) Chatbots zuzuordnen. Er beschafft sich die gesuchten Informationen über eine Schnittstelle und gibt diese an den Benutzer weiter.

Das Ziel (goal), welches der entwickelte Chatbot verfolgt ist eindeutig. Die Benutzer sollen mit Informationen zum Studiengang und dem Studienverlauf versorgt werden. Deshalb ist dieser der Gruppe der informativen (informative) Chatbots zuzuordnen.

Wenn die Art der Verarbeitung des Inputs und Erstellung des Outputs (input processing and response generation method) betrachtet wird, muss der Chatbot den abrufbasierenden (retrieval-based) Modellen zugeordnet werden. Der Chatbot generiert die Ant-

worten auf die Fragen der Benutzer aus Abfragen von verschiedenen API's oder Datenbanken.

Der Chatbot besitzt einen gewissen Anteil an menschlicher Hilfe (human-aided). So müssen zumindest Menschen entscheiden, welche Fragen in der Interaktionshistorie richtig klassifiziert wurden und welche nicht. Auf Basis dieser Entscheidungen kann der Bot weiter aus seinem Chatverlauf lernen.

In Hinsicht auf die Entwicklungsplattformen wurden bei der Entwicklung des Bots ausschließlich frei zugängliche (open-source) Python Bibliotheken benutzt. Es wurde auf geschlossene Systeme von großen Firmen verzichtet. Auch die Datengrundlage wurde selbst erstellt.

5 Evaluation des Prototyps

Um den Chatbot zu evaluieren, wurde ein Usability Test in Kombination mit einem Fragebogen angewendet. Die Teilnehmer mussten in verschiedenen Aufgaben Informationen, sowohl über die Webseite der Hochschule Aalen als auch über den in dieser Abschlussarbeit entwickelten Chatbot beschaffen. Diese Aufgaben sind im Anhang B.1 zusehen. Anschließend wurden den Befragten geschlossene Fragen bzw. Behauptungen vorgelegt. Diesen konnten sie zustimmen, eher zustimmen, weder zustimmen noch ablehnen, eher nicht zustimmen oder nicht zustimmen. Zum Abschluss des Fragebogens konnten die Teilnehmer in einer offenen Frage Feedback und Verbesserungsvorschläge geben. Die Fragen, die im Fragebogen gestellt wurden, sind im Anhang B.1 zusehen.

Die Evaluation wurde mit insgesamt vierzehn Teilnehmern durchgeführt. Von diesen vierzehn Teilnehmern sind sechs Teilnehmer Studenten bzw. ehemalige Studenten des Studiengangs Wirtschaftsinformatik an der Hochschule Aalen, zwei Teilnehmer sind in der IT-Branche tätig und sechs Teilnehmer haben keinen direkten Bezug zur Informatik. Die Teilnehmer, die keinen Bezug zur Informatik haben, sollen hierbei Studienanfänger ohne Vorkenntnisse im Bereich Informatik simulieren. Die Teilnehmer, welche in der IT-Branche tätig sind, sollen Studienanfänger mit Vorkenntnissen aus dem Bereich Informatik darstellen.

Nachdem Informationen über die Webseite der Hochschule Aalen sowie über den Chatbot durch den Teilnehmer beschafft wurden, wurden geschlossene Fragen über diese Medien gestellt. Alle Fragen inklusive der Ergebnisse sind im Anhang B.2.1 zu finden. Zuerst wurden Fragen zur Informationsbeschaffung mithilfe der Webseite der Hochschule Aalen gestellt. Hierbei gaben 93% der Befragten an, dass sie die Webseite der Hochschule Aalen (beispielhaft Studiengangseite Wirtschaftsinformatik) als nicht intuitiv empfinden. So lehnten fünf Probanden die Behauptung, dass die Webseite intuitiv wäre, ab, acht Teilnehmer lehnten die Behauptung eher ab und nur ein einziger Teilnehmer stimmte der Behauptung eher zu. Bei der Frage, ob die Informationsbeschaffung über die Webseite frustrierend sei, stimmten fünf Teilnehmer zu und weitere sechs Teilnehmer eher zu. Lediglich ein Teilnehmer stimmte weder zu noch lehnte er ab und zwei Teilnehmer stimmten der Aussage eher nicht zu. 79% der Teilnehmer empfanden die Informationsbeschaffung über die Webseite damit eher als frustrierend. Es ist zu erkennen, dass die Mehrheit der befragten Teilnehmer der Informationsbeschaffung über die Webseite eher negativ gegenüberstehen.

Bei der Informationsbeschaffung mit dem Chatbot sieht dies anders aus. Hierbei finden 100% der Befragten den Bot (eher) intuitiv. Neun Teilnehmer stimmten der Behauptung, dass der Bot intuitiv sei, eher zu. Fünf weitere Teilnehmer stimmten dieser Behauptung sogar komplett zu. Außerdem empfand kein Teilnehmer die Kommunikation mit dem Chatbot als frustrierend.

Auch die Bedienung und Kommunikation mit dem Chatbot erhielt positives Feedback. So hat der Chatbot bei allen Teilnehmern die Fragen, die sie ihm stellten, verstanden

und mit kleinen Ausnahmen korrekt beantwortet. Der Chatbot konnte die Fragen der Teilnehmer des Weiteren zügig beantworten. Außerdem gaben alle Teilnehmer an, dass die Kommunikation und Informationsbeschaffung mit dem Chatbot zufriedenstellend war.

100% aller Probanden hatten das Gefühl Zeit bei der Informationsbeschaffung mit dem Chatbot, verglichen mit der Webseite der Hochschule Aalen, zu sparen. Dies lässt die Forschungsfrage, ob der Chatbot die Dauer der Informationsbeschaffung senken kann, mit ja beantworten. Durch den Chatbot wird es ermöglicht, direkte Antworten oder Dokumente auf die gewünschten Fragen zu erhalten. Außerdem gaben alle Benutzer an, dass sie den Chatbot benutzen würden, wenn dieser zur Verfügung stehen würde.



Abbildung 5.1: Frage 9 der Evaluation mit Verhältnis der Antworten.

Das Feedback der Teilnehmer war durchweg positiv. Vor allem freuten sich die Teilnehmer über die Übersichtlichkeit, die schnellen Antworten und die intuitive Bedienung. Doch auch Verbesserungsvorschläge wurden durch die Probanden genannt. So könnte der Chatbot weitere Informationen über z.B. das außerstudentische Leben geben. Auch Schreibfehler können eine falsche Klassifikation auslösen, hier könnte eine Auto-korrektur von Schreibfehlern bei der Benutzereingabe eingebaut werden. Des Weiteren könnten Module bei der Konfigurierung nicht nur über die Id sondern auch über den Modulnamen hinzugefügt werden. Die Auflistung des Feedbacks der Teilnehmer ist im Anhang B.2.2 zu sehen.

Die Evaluation durch den Usability Test und den Fragebogen ergibt somit, dass der Chatbot von den Probanden der Webseite vorgezogen werden würde. Die Webseite wird in der Umfrage eher negativ bewertet, wobei der Chatbot ausschließlich positiv bewertet wird. Auch die Dauer der Informationsbeschaffung kann durch den Chatbot gesenkt werden. Dieser erhöht somit die Effektivität der Studierenden bei der Beschaffung von Informationen.

6 Fazit, Limitationen und Zukunftsausblick

Zum Ende dieser Abschlussarbeit wird in diesem Kapitel ein Fazit gezogen. Es wird hierbei auf die Entwicklung, die Evaluation und die Beantwortung der Forschungsfragen eingegangen. Außerdem werden Limitationen herausgearbeitet, welche die Entwicklung des Chatbots beeinträchtigt haben und Punkte genannt, die am Prototyp verbessert werden können. Zuletzt wird ein Zukunftsausblick gegeben, in dem erläutert wird, wie der Chatbot weiterentwickelt werden könnte und was benötigt wird, um den Chatbot an der Hochschule Aalen zur Verfügung zu stellen.

6.1 Fazit

In dieser Abschlussarbeit wurde ein Chatbot konzipiert und entwickelt. Dieser Chatbot hat den Zweck, Studierende bei der Informationsbeschaffung zu unterstützen. Aufbauend auf einer, durch eine Literaturrecherche gebildete, Wissensbasis (siehe Kapitel 2) und einer Anforderungsanalyse (siehe Kapitel 3), wurde ein Konzept für die Entwicklung eines Chatbots entwickelt (siehe Kapitel 4.1). Anschließend wurde aufbauend auf diesem Konzept ein Prototyp entwickelt (siehe Kapitel 4.3). Durch das Konzept und die darauffolgende Entwicklung wurde die Forschungsfrage, wie ein Chatbot zur Informationsbeschaffung entwickelt werden kann, beantwortet und beschrieben. Dieser Prototyp wurde durch einen Usability Test in Kombination mit einem Fragebogen evaluiert. Aus dieser Evaluation resultierten mehrere spannende Ergebnisse. So fanden die Probanden die Webseite der Hochschule Aalen meist nicht intuitiv und empfanden die Informationsbeschaffung darüber eher als frustrierend. Die Informationsbeschaffung mit dem Chatbot wurde hingegen als intuitiv und nicht frustrierend wahrgenommen. Auch gaben die Befragten an, dass der Chatbot meist alle Fragen verstanden und korrekt und zügig beantwortet hat. Sie empfanden die Informationsbeschaffung mit dem Chatbot als zufriedenstellend. Außerdem hatten die Teilnehmer das Gefühl, Zeit bei der Informationsbeschaffung zu sparen (siehe Abb. 5.1). Somit wurde auch die zweite Forschungsfrage, ob die Dauer der Informationsbeschaffung durch den Chatbot gesenkt werden könne, positiv beantwortet. Nicht nur den Studierenden könnte durch einen Chatbot geholfen werden, auch das Sekretariat könnte durch den Einsatz entlastet werden, da viele Fragen eventuell schon vorher beantwortet werden könnten. Zusammenfassend war die Resonanz der Testpersonen durchweg positiv und alle gaben an, den Chatbot zu nutzen, wenn dieser verfügbar wäre.

6.2 Limitationen

Nach der Konzeption, Entwicklung und Evaluation des Chatbots haben sich auch Limitationen herausgestellt. Es gibt dabei Limitationen am Prototyp selbst sowie in der Entwicklungsumgebung. Der Chatbot kann zum Beispiel in manchen Fällen Zusammenhänge in Sätzen nicht erkennen. Ein Lösungsansatz wäre hierfür z.B. der Umstieg von TF-IDF auf den Word2Vec-Algorithmus zur Vektorisierung der Benutzereingabe.

Eine weitere Einschränkung ist die Auswahl der Module zum Stundenplan über den Chatbot. Dies ist bisher nur über die Id des Moduls möglich. Auch die Wahl des Messengerdiensts könnte eine Einschränkung sein. Nicht jeder Student besitzt Telegram. Außerdem hat die Telegram-API ein Zeichenlimit bei der Ausgabe von Nachrichten und es werden nicht alle HTML-Tags verarbeitet und richtig dargestellt. Auch die Datengrundlage ist limitiert. Um noch genauere Vorhersagen treffen zu können und die Benutzereingaben noch besser klassifizieren zu können, sollte eine größere Datengrundlage vorliegen. Außerdem sollten genaue Datumswerte vom Chatbot verarbeitet werden können. Bisher werden nur Heute, Morgen, diese Woche oder nächste Woche korrekt verarbeitet. Aber nicht nur der Prototyp an sich besitzt Limitationen, auch in der Entwicklungsumgebung sind einige Einschränkungen vorhanden. So besitzt Starplan zum Beispiel keine öffentlich nutzbare API. Ohne eine solche Schnittstelle ist die Bereitstellung der Informationen über Module und Vorlesungen äußerst mühsam. Außerdem würden kontinuierliche Benennungen in der HTML-Datei der Webseite der Hochschule Aalen, die Informationsbeschaffung über Webscraping erleichtern.

6.3 Zukunftsausblick

Um den Bot weiter zu verbessern, können in Zukunft eine Reihe an Entwicklungen durchgeführt werden. So könnten z.B. das Hinzufügen und Löschen eines Moduls zum Stundenplan auch über den Namen des Moduls und nicht nur über die Id erfolgen. Außerdem könnte der Informationsbereich um Informationen zum außerstudentischen Leben erweitert werden. Es könnte hierbei zum Beispiel über das Nachtleben, das Kultur- und Sportangebot oder über die Infrastruktur in Aalen informiert werden. Eine weitere Weiterentwicklung wäre die Möglichkeit, Termine und Vorlesungen für ein spezielles Datum abfragen zu können. Diese Funktion könnte Studenten zum Beispiel bei der Terminplanung unterstützen. Die größte Weiterentwicklung wäre die Ausbreitung des Chatbots auf alle Studiengänge der Hochschule Aalen.

Um den Chatbot an der Hochschule integrieren und einführen zu können, müssen verschiedene Limitationen zuvor beseitigt werden. So wird zum Beispiel eine API von Starplan oder eine andere Schnittstelle zur Bereitstellung der Informationen über Module und Veranstaltungen benötigt. Außerdem sollten die Benennungen in der HTML-Datei der Webseite der Hochschule Aalen einheitlich sein, nur so kann eine reibungslose Informationsbeschaffung durch den Bot gewährleistet werden.

Trotz dieser Limitationen ist ein Chatbot ein Tool, das die Studierenden in ihrem Alltag unterstützen könnte. Es könnte die Dauer der Informationsbeschaffung senken, die Effektivität erhöhen und das Sekretariat entlasten. Eine Einführung eines solchen Chatbots könnte für alle Personen innerhalb der Hochschule einen Vorteil darstellen.

Anhang A: Auszüge aus dem Sourcecode

Der komplette Quelltext ist unter https://github.com/Kn3ule/WI_HSAA_Bot verfügbar!

A.1 Funktionen der Datenerhebung

A.1.1 Extrahierung der Vorlesungsdaten

```
6  #function to read the modules
7  def get_module_from_ical(cal):
8      modules = []
9      for component in cal.walk():
10         if component.name == 'VEVENT':
11             description = component.get('description')
12             description = description.split('\n')
13
14             for x in description:
15                 parts = x.split('\n')
16
17                 if len(parts) == 6:
18                     del parts[3]
19
20                 name = parts[0].split('(')[0][: -1]
21                 try:
22                     number = parts[0].split('(')[1].replace(')', '')
23                 except:
24                     number= ''
25
26                 lecturer = parts[1]
27                 semester = parts[2]
28
29                 if len(parts) > 4:
30                     canvas = parts[3].split(' ')[1]
31                 else:
32                     canvas = ''
33
34                 entry = {
35                     "name": name,
36                     "number": number,
37                     "lecturer": lecturer,
38                     "semester": semester,
39                     "canvas": canvas,
40                 }
41
42                 if entry not in modules:
43                     modules.append(entry)
44
45     return modules
```

Listing 1: Funktion, um Module aus ICal zu extrahieren.

```

47 | #function to read the lecture dates
48 | def get_lecture_dates_from_ical(cal):
49 |     lecture_data = []
50 |
51 |     for component in cal.walk():
52 |         if component.name == "VEVENT":
53 |             name = component.get('description').replace('\n', '').split('\n')[0].split('(')[0][:-1]
54 |             startdt = component.get('dtstart').dt
55 |             enddt = component.get('dtend').dt
56 |             room = component.get('location').split(',')[0]
57 |
58 |             if room == "Canvas" or room == "--":
59 |                 room = "online"
60 |
61 |             entry = {
62 |                 "name": name,
63 |                 "startdt": startdt,
64 |                 "enddt": enddt,
65 |                 "room": room
66 |             }
67 |             lecture_data.append(entry)
68 |
69 |     return lecture_data

```

Listing 2: Funktion, um Veranstaltungsdaten aus ICal zu extrahieren.

A.1.2 Abrufen der Informationen von der Webseite

```

16 | def get_downloads(input):
17 |     page = requests.get(downloads)
18 |     soup = BeautifulSoup(page.text, 'html.parser')
19 |
20 |     panel = soup.find_all('div', class_='panel')
21 |
22 |     try:
23 |         for p in panel:
24 |             if p.find('div', text=input) != None:
25 |                 head = p.find('div', text=input)
26 |                 data = p.find_all('div', class_='paragraph')
27 |
28 |                 heading = f'<b> {str(head.get_text())} </b> \n'
29 |                 output = [heading]
30 |                 for d in data:
31 |                     text = str(d.get_text())
32 |                     if d.find('a') != None:
33 |                         link = hsaa + str(d.find('a').get('href'))
34 |                         link_text = str(d.find('a').getText())
35 |                         output.append(f'{text} \n <a href="{link}"> {link_text} </a> \n')
36 |                     else:
37 |                         output.append(f'{text} \n')
38 |
39 |     except Exception:
40 |         output = ["Ich konnte zu diesem Thema nichts finden"]
41 |
42 |     return output

```

Listing 3: Funktion, um Informationen von der Webseite des Studiengangs zu erhalten.

A.1.3 Abrufen des RSS-Feeds

```
86 def get_news():
87     if hasattr(ssl, '_create_unverified_context'):
88         ssl._create_default_https_context = ssl._create_unverified_context
89     feed = feedparser.parse(f'{basic_url}/infos.rss')
90
91     #check if
92     if len(feed.entries) == 0:
93         return 'Keine Neuigkeiten im Studiengang Wirtschaftsinformatik'
94
95
96     output = [f'<b> Neues im Studiengang Wirtschaftsinformatik: </b> \n\n']
97     for e in feed.entries:
98         title = f'{e.title}\n'
99
100        output.append(f'{title}\n')
101
102    output.append(f'mehr sehen: {basic_url}/news')
103
104    return output
```

Listing 4: Funktion, um RSS-Feed des Studiengangs auszulesen.

A.2 Verbindung zur Telegram API

A.2.1 Einbindung der Command- und Message-Handler

```
151 def main():
152
153     #create an Updater Object and a dispatcher
154     updater = Updater(token=token, use_context=True)
155     dispatcher = updater.dispatcher
156
157     #create Command and MessageHandler
158     start_handler = CommandHandler('start', start)
159     modules_handler = CommandHandler('modules', modules)
160     addmodule_handler = CommandHandler('add', add_module)
161     deletemodules_handler = CommandHandler('delete', delete_module)
162     help_handler = CommandHandler('help', help)
163     echo_handler = MessageHandler(Filters.text & (~Filters.command), echo)
164
165     #add handler to dispatcher
166     dispatcher.add_handler(start_handler)
167     dispatcher.add_handler(echo_handler)
168     dispatcher.add_handler(modules_handler)
169     dispatcher.add_handler(addmodule_handler)
170     dispatcher.add_handler(deletemodules_handler)
171     dispatcher.add_handler(help_handler)
172     updater.start_polling()
```

Listing 5: Einbindung der Command- und Message-Handler.

A.2.2 Funktionen, die durch Handler aufgerufen werden können

```

18 | #method to start the bot
19 | def start(update: Update, context: CallbackContext):
20 |     #check if student is already registered, if not -> add to database
21 |     try:
22 |         new_student = Student(telegram_id = update.message.from_user.id, first_name= update.message.from_user.first_name, last_name= update.message.from_user.last_name)
23 |         my_session.add(new_student)
24 |         my_session.commit()
25 |     except Exception as err:
26 |         my_session.rollback()
27 |     greeting = (
28 |         f'Hallo {update.message.from_user.first_name} {update.message.from_user.last_name}!\n'
29 |         f'Ich bin der Informationsbot des Studiengangs Wirtschaftsinformatik der Hochschule Aalen\n'
30 |         f'Falls du deine Module konfigurieren und hinterlegen möchtest, um sie später abzufragen, starte den Prozess mit: /modules \n'
31 |         f'Um Andere Informationen zu erhalten Frag mich einfach!\n'
32 |         f'Falls du Hilfe benötigst, öffne das Hilfe-Menü mit /help'
33 |     )
34 |     context.bot.send_message(chat_id=update.effective_chat.id, text=greeting)
35 |

```

Listing 6: Start-Funktion.

```

106 | def help(update: Update, context: CallbackContext):
107 |     return_string = (
108 |         '<b> Hilfe: </b>\n\n'
109 |         'Modulübersicht: /modules \n'
110 |         'Modul hinzufügen: /add {id}\n'
111 |         'Modul entfernen: /delete {id}\n\n'
112 |         'Für andere Informationen Fragen sie mich einfach in natürlicher Sprache :D\n'
113 |         'Ich gebe Informationen über folgende Themen:\n'
114 |         'Vorlesungen\n'
115 |         'Stundenplan\n'
116 |         'Praxissemester\n'
117 |         'Allgemeine Informationen\n'
118 |         'Beurlaubung\n'
119 |         'Leistungsanerkennung\n'
120 |         'Modulbeschreibungen\n'
121 |         'Prüfungen\n'
122 |         'SP0\n'
123 |         'Studium Generale\n'
124 |         'Exmatrikulation\n'
125 |         'Summerschool\n'
126 |         'Bachelorarbeit\n'
127 |         'Auslandssemester\n'
128 |     )
129 |     context.bot.send_message(chat_id=update.effective_chat.id, text=return_string, parse_mode = 'HTML')

```

Listing 7: Help-Funktion.

```

40 #method to show all available modules
41 def modules(update: Update, context: CallbackContext):
42     #all available methods were read
43     all_modules = my_session.query(Module)
44     all_modules_string = '<b> Übersicht aller Module </b> \n'
45     for module in all_modules:
46         id = module.id
47         name = module.module_name
48
49         all_modules_string += f'{id} - {name} \n'
50
51 #modules that the student has already chosen
52 my_modules = my_session.query(enrolment_table).join(Student).join(Module).filter(Student.telegram_id == update.message.from_user.id)
53 my_modules_string = '\n \n <b> Meine Module: </b> \n'
54 for module in my_modules:
55     id = module.modules_id
56     name = my_session.query(Module).filter(Module.id == id).first().module_name
57     my_modules_string += f'{id} - {name} \n'
58
59
60 return_string = (
61     '<b>Hier kannst du deine Module konfigurieren:</b> \n'
62     'Mit /add {id} kannst du neue Module für deinen Studienplan hinzufügen.\n'
63     'Mit /delete {id} kannst du Module aus deinem Stundenplan löschen!\n \n \n'
64 )
65
66 #Return String wird zusammengesetzt und als Nachricht zurückgeben
67 return_string = return_string + all_modules_string + my_modules_string
68
69 context.bot.send_message(chat_id=update.effective_chat.id, text=return_string, parse_mode = 'HTML')

```

Listing 8: Funktion zur Modulübersicht.

```

74 #method to connect students and modules(Many-to-Many)
75 def add_module(update: Update, context: CallbackContext):
76     student = my_session.query(Student).filter(Student.telegram_id == update.message.from_user.id).first()
77     module = my_session.query(Module).filter(Module.id == update.message.text.split(' ')[1]).first()
78     relation_exist = my_session.query(enrolment_table).join(Student).join(Module).filter((Student.telegram_id == update.message.from_user.id) & (Module.id == update.message.text.split(' ')[1])).first()
79     try:
80         if relation_exist != None:
81             context.bot.send_message(chat_id=update.effective_chat.id, text="Modul ist schon hinterlegt!", parse_mode = 'HTML')
82         else:
83             student.modules.append(
84                 module
85             )
86
87             my_session.commit()
88             return_string = f'Das Modul: {module.module_name} wurde hinzugefügt!'
89
90             context.bot.send_message(chat_id=update.effective_chat.id, text= return_string, parse_mode = 'HTML')
91     except:
92         context.bot.send_message(chat_id=update.effective_chat.id, text= "Modul nicht verfügbar. Bitte überprüfe deine Eingabe!", parse_mode = 'HTML')

```

Listing 9: Funktion, um Module zum Stundenplan hinzuzufügen

```

95 #method to delete the connection between a student and the chosen module
96 def delete_module(update: Update, context: CallbackContext):
97     student = my_session.query(Student).filter(Student.telegram_id == update.message.from_user.id).first()
98     module = my_session.query(Module).filter(Module.id == update.message.text.split(' ')[1]).first()
99
100     try:
101         student.modules.remove(module)
102         context.bot.send_message(chat_id=update.effective_chat.id, text= f"Modul: {module.module_name} wurde entfernt!", parse_mode = 'HTML')
103     except:
104         context.bot.send_message(chat_id=update.effective_chat.id, text= "Modul konnte nicht entfernt werden!", parse_mode = 'HTML')

```

Listing 10: Funktion, um Module aus dem Stundenplan zu entfernen.

```

134 def echo(update: Update, context: CallbackContext):
135     #give user message to svm and get the decision
136     tag = svm(update.message.text)
137     #share decision tag to get the output for the user
138     output = get_output(tag, update.message.text, update.message.from_user.id)
139     #store user input and decision tag, to learn the model
140     add_user_interaction(tag, update.message.text)
141     You, vor 3 Monaten • Update tg_connection.py
142     #check if output is too long for a single message
143     for o in output:
144         if len(o) > 4096:
145             for x in range(0, len(o), 4096):
146                 context.bot.send_message(chat_id=update.effective_chat.id, text=o[x:x+4096], parse_mode='HTML')
147         else:
148             context.bot.send_message(chat_id=update.effective_chat.id, text=o, parse_mode='HTML')

```

Listing 11: Echo-Funktion, um auf Nachrichten ohne Befehl zu antworten.

A.3 NLP-Funktionen

A.3.1 Vorverarbeitung der Textdaten

```

1  import spacy
2  from spacy.lang.de.stop_words import STOP_WORDS
3
4
5
6  nlp = spacy.load('de_core_news_md')
7
8  def preprocess(sentence):
9      doc = nlp(sentence)
10
11      lemma_tokens = []
12      for token in doc:
13          if token.is_stop == False and token.is_punct == False:
14              lemma_tokens.append(token.lemma_)
15      #am besten keine Kleinschreibung
16      return ' '.join(lemma_tokens)

```

Listing 12: Funktion zur Vorverarbeitung von Textdaten.

A.4 ML-Funktionen

A.4.1 Datengrundlage

```

1  tags;patterns
2  AKTUELLES;Was gibt es für Neuigkeiten?
3  AKTUELLES;Was steht auf dem schwarzen Brett?
4  AKTUELLES;Gibt es Neuigkeiten auf dem schwarzen Brett?
5  AKTUELLES;Gibt es aktuelle Neuigkeiten?
6  AKTUELLES;Was gibt es aktuelles?
7  AKTUELLES;Was steht aktuell auf dem schwarzen Brett?
8  INFORMATIONEN_STUDIUM;Welche allgemeinen Informationen zum Studium gibt es?
9  INFORMATIONEN_STUDIUM;Wie funktioniert die Hochschul IT?
10 INFORMATIONEN_STUDIUM;Gibt es allgemeine Informationen zum Studium?
11 INFORMATIONEN_STUDIUM;Wie kann ich mich mit dem WLAN der Hochschule verbinden?
12 INFORMATIONEN_STUDIUM;Gibt es einen VPN zur Hochschule?
13 INFORMATIONEN_STUDIUM;Wie kann ich mich mit dem VPN der Hochschule verbinden?
14 BEURLAUBUNG;Wie kann ich mich Beurlauben lassen?
15 BEURLAUBUNG;Gibt es ein Formular zum Antrag auf Beurlaubung?
16 BEURLAUBUNG;Gibt es ein Formular zum Antrag auf ein Urlaubssemester?
17 BEURLAUBUNG;Wo finde ich den Antrag auf Beurlaubung?
18 LEISTUNGSANERKENNUNG;Wie kann ich mir Leistungen anerkennen lassen?
19 LEISTUNGSANERKENNUNG;Kann ich mir Leistungen anerkennen lassen?
20 LEISTUNGSANERKENNUNG;Wie kann ich mir Prüfungen anerkennen lassen?
21 LEISTUNGSANERKENNUNG;Wie kann ich mir Prüfungen anrechnen lassen?
22 LEISTUNGSANERKENNUNG;Kann ich mir bereits erbrachte Prüfungen anrechnen lassen?

```

Listing 13: Auszug aus der Trainingsdaten CSV-Datei.

A.4.2 Vorbereitung der Datengrundlage

```
--
16 #initialize Labelencoder and Tf-idf vectorizer
17 le = LabelEncoder()
18 vectorizer = TfidfVectorizer()
19
20 #import training data
21 training_data = pd.read_csv("training_data.csv", sep=';')
22 patterns = training_data['patterns'].values
23
24 tags = []
25 for i in training_data["tags"]:
26     if i not in tags:
27         tags.append(i)
28
29 X = []
30 for pattern in patterns:
31     X.append(preprocess(pattern))
32
33 vectorizer.fit(X)
34 le.fit(training_data['tags'])
35
36
37 X = vectorizer.transform(X)
38 y = le.transform(training_data['tags'])
39
40 trainx, testx, trainy, testy = tts(X, y, test_size=.25, random_state=42)
```

Listing 14: Vorbereitung der Datengrundlage.

A.4.3 Training und Vergleich der Klassifikationsalgorithmen

```
42 #define svm and get accuracy and f1-score
43 svm_model = SVC(kernel='linear').fit(trainx, trainy)
44 print("SVM:\naccuracy:", svm_model.score(testx, testy))
45 svm_prediction = svm_model.predict(testx)
46 print("f1-score:", f1_score(testy, svm_prediction, average='weighted'))
47 print('\n\n')
48
49
50 #define kNN
51 knn_model = KNeighborsClassifier(n_neighbors=2).fit(trainx, trainy)
52 knn_prediction = knn_model.predict(testx)
53 print("KNN: \naccuracy:", knn_model.score(testx, testy))
54 print("f1-score:", f1_score(testy, knn_prediction, average='weighted'))
55 print('\n\n')
56
57 #define NB
58 NB_model = GaussianNB().fit(trainx.toarray(), trainy)
59 NB_prediction = NB_model.predict(testx.toarray())
60 print("NB: \naccuracy:", NB_model.score(testx.toarray(), testy))
61 print("f1-score:", f1_score(testy, NB_prediction, average='weighted'))
62 print('\n\n')
63
64 #define decision tree algo
65 dt_model = DecisionTreeClassifier().fit(trainx, trainy)
66 dt_prediction = dt_model.predict(testx)
67 print("Decision Tree: \naccuracy:", dt_model.score(testx.toarray(), testy))
68 print("f1-score:", f1_score(testy, dt_prediction, average='weighted'))
69 print('\n\n')
```

Listing 15: Einbindung, Training und Vergleich der ML-Algorithmen.

A.4.4 Einbindung der SVM

```
73 def svm(user_input):  
74     transform_input = vectorizer.transform([preprocess(user_input)])  
75     tag = le.inverse_transform(svm_model.predict(transform_input))[0]  
76  
77     return tag
```

Listing 16: Funktion, um Eingabe des Benutzers zu klassifizieren.

A.4.5 Abspeicherung der Klassifikationshistorie

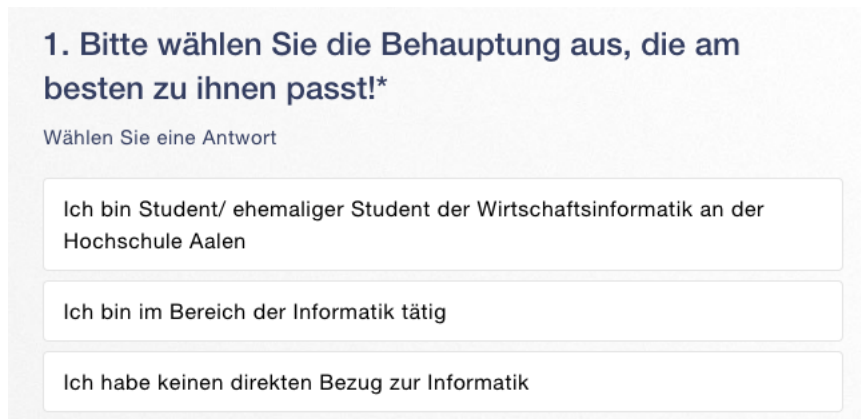
```
3 def add_user_interaction(tag, question):  
4     df = pd.DataFrame([  
5         [tag, question]  
6     ],  
7     columns=['predicted_tag', 'question_asked']  
8     )  
9  
10    df.to_csv(r'user_interaction.csv', mode = 'a', header = False, index = False, sep=';')
```

Listing 17: Funktion zum Abspeichern der Klassifikationshistorie.

Anhang B: Evaluation und Evaluationsergebnisse

B.1 Evaluation durch Usability Test und Fragebogen

Begonnen wurde die Evaluation indem erfragt wurde, ob der befragte Student oder ehemaliger Student des Studiengangs Wirtschaftsinformatik an der Hochschule Aalen ist, im Bereich Informatik tätig ist oder er keinen Bezug zur Informatik hat.



1. Bitte wählen Sie die Behauptung aus, die am besten zu ihnen passt!*

Wählen Sie eine Antwort

- ☐ Ich bin Student/ ehemaliger Student der Wirtschaftsinformatik an der Hochschule Aalen
- ☐ Ich bin im Bereich der Informatik tätig
- ☐ Ich habe keinen direkten Bezug zur Informatik

Abbildung B.1: Frage zur Einordnung der Befragten.

Folgende Aufgaben wurden dem Befragten im Zuge eines Usability Tests gestellt.

Nachfolgend werden Ihnen verschiedene Aufgaben gestellt, die Sie bitte in vorgegebener Reihenfolge bearbeiten sollen.

1. Öffnen Sie die Seite des Studiengangs Wirtschaftsinformatik. Sie finden diese unter: <https://www.hs-aalen.de/de/courses/39>
2. Navigieren Sie zum Schwarzen Brett um aktuelle Informationen zu erhalten.
3. Suchen Sie nach Informationen zum Praxissemester.
4. Suchen Sie nach dem Formular zur Anmeldung der Bachelorarbeit
5. Finden Sie heraus, wie der Vorlesungsplan für das Semester 4(WI4) im Studiengang Wirtschaftsinformatik (WI) stattfindet.
6. Starten Sie den Chatbot in Telegram. Dieser ist unter WI_HSAA_Bot zu finden.
7. Fügen Sie IT-Sicherheit, Datenbanken und Wirtschaftsrecht zum Stundenplan hinzu!
8. Fragen Sie die Vorlesungen für den heutigen Tag ab.
9. Fragen Sie ab wann in dieser Woche die Veranstaltung IT-Sicherheit stattfindet
10. Fragen Sie ab, ob morgen Wirtschaftsrecht gelesen wird
11. Fragen Sie ab, welche Vorlesungen nächste Woche gelesen werden
12. Beschaffen Sie sich das nötige Dokument zur Exmatrikulation
13. Informieren Sie sich über das Studium Generale und dessen Ablauf.

Abbildung B.2: Aufgaben zur Durchführung eines Usability Tests.

Nachdem die Befragten die Aufgaben abgearbeitet haben, wurde der Chatbot mit folgenden Behauptungen evaluiert. Die Befragten konnten den Behauptungen zustimmen, eher zustimmen, weder zustimmen noch ablehnen, eher nicht zustimmen oder nicht zustimmen.

Tabelle B.1: Behauptungen zur Evaluation.

Die Interaktion auf der Webseite der Hochschule Aalen war intuitiv.
Die Informationsbeschaffung auf der Webseite der Hochschule Aalen war frustrierend.
Die Interaktion mit dem Chatbot war intuitiv.
Der HSAA Chatbot hat alle meine Fragen verstanden und korrekt beantwortet.
Der HSAA Chatbot konnte mir meine Fragen zügig beantworten.
Die Kommunikation mit dem Chatbot war frustrierend.
Die Informationsbeschaffung mit dem Chatbot war zufriedenstellend.
Ich habe das Gefühl durch den Chatbot Zeit bei der Informationsbeschaffung zu sparen (im Vergleich zur Webseite der Hochschule).
Wenn der Studiengang Wirtschaftsinformatik an der Hochschule Aalen einen Chatbot zur Informationsbeschaffung anbieten würde, würde ich diesen nutzen.

Zum Abschluss der Umfrage, wurden die Teilnehmer nach Feedback und Verbesserungsvorschlägen gefragt.

11. Gibt es Feedback, dass Sie den Entwicklern mitteilen möchten? (Verbesserungsvorschläge, Meinungen etc.)

Schreiben Sie einen kurzen Text...

500

Abbildung B.3: Frage nach Feedback im Fragebogen.

B.2 Auswertung der Evaluation

B.2.1 Fragen und Antworten

Alle Fragen konnten mit stimme ich zu (++), stimme ich eher zu (+), stimme ich weder zu noch lehne ich ab (0), stimme ich eher nicht zu (-) und stimme ich nicht zu (--) beantwortet werden.

Tabelle B.2: Auswertung der Fragen bei 14 Teilnehmern.

Frage	++	+	0	-	--
Die Interaktion auf der Webseite der Hochschule Aalen war intuitiv.	0	1	0	8	5
Die Informationsbeschaffung auf der Webseite der Hochschule Aalen war frustrierend.	5	6	1	2	0
Die Interaktion mit dem Chatbot war intuitiv.	5	9	0	0	0
Der HSAA Chatbot hat alle meine Fragen verstanden und korrekt beantwortet.	10	4	0	0	0
Der HSAA Chatbot konnte mir meine Fragen zügig beantworten.	12	1	1	0	0
Die Kommunikation mit dem Chatbot war frustrierend.	0	0	0	4	10
Die Informationsbeschaffung mit dem Chatbot war zufriedenstellend.	13	1	0	0	0
Ich habe das Gefühl durch den Chatbot Zeit bei der Informationsbeschaffung zu sparen (im Vergleich zur Webseite der Hochschule).	14	0	0	0	0
Wenn der Studiengang Wirtschaftsinformatik an der Hochschule Aalen einen Chatbot zur Informationsbeschaffung anbieten würde, würde ich diesen nutzen.	13	1	0	0	0

B.2.2 Feedback der Teilnehmer

Tabelle B.3: Feedback und Verbesserungsvorschläge der Teilnehmer.

Feedback:
war alles ok
Sehr schöner bot. Hilft weiter..: jeder kennt die HS Aalen Seiten und ihre komplette Unübersichtlichkeit
Sehr intuitiv, schnelle Antworten, präzise beim Verstehen
Keine Verbesserungsvorschläge

Geiles Ding
Eventuell könnte man bei den Modulen noch die ETC angeben. Ansonsten eine sehr schöne Erleichterung beim Suchen der Dokumente.
Es wurden nicht alle Formulierungen der Fragen erkannt.
Die Module könnten bei der Modulkonfiguration auch über den Modulnamen eingegeben werden
Der Bot hat mir alle Fragen sofort und richtig beantwortet :)
Daumen hoch, schnelle und klare Antworten ohne lästiges Suchen. Bin sehr zufrieden mit dem Chatbot :)
Coole Idee zur raschen und einfachen Informationsbeschaffung, kann man noch auf weitere Funktionen erweitern (z.B. außerstudentisches Leben etc.)
auf Schreibfehler wird nicht richtig reagiert evtl. eine Autokorrektur einbauen

Quellenverzeichnis

- [1] A. R. Hevner, S. T. March, J. Park, und S. Ram, „Design science in information systems research“, *MIS quarterly*, S. 75–105, 2004.
- [2] M. Benner-Wickner, R. Kneuper, und I. Schlömer, „Leitfaden für die Nutzung von Design Science Research in Abschlussarbeiten“, IUBH Discussion Papers-IT & Engineering, 2020.
- [3] A. R. Hevner, „A three cycle view of design science research“, *Scandinavian journal of information systems*, Bd. 19, Nr. 2, S. 4, 2007.
- [4] R. Khan und A. Das, *Build Better Chatbots*. Apress, 2018. doi: 10.1007/978-1-4842-3111-1.
- [5] A. M. Turing, „I.—COMPUTING MACHINERY AND INTELLIGENCE“, *Mind*, Bd. LIX, Nr. 236, S. 433–460, Okt. 1950, doi: 10.1093/mind/LIX.236.433.
- [6] J. Weizenbaum, „ELIZA—a computer program for the study of natural language communication between man and machine“, *Commun ACM*, Bd. 9, Nr. 1, S. 36–45, 1966.
- [7] V. Sharma, M. Goyal, und D. Malik, „An intelligent behaviour shown by chatbot system“, *International Journal of New Technology and Research*, Bd. 3, Nr. 4, S. 263312, 2017.
- [8] E. Adamopoulou und L. Moussiades, „Chatbots: History, technology, and applications“, *Machine Learning with Applications*, Bd. 2, S. 100006, Dez. 2020, doi: 10.1016/j.mlwa.2020.100006.
- [9] Mgr. T. Zemčík, „A Brief History of Chatbots“, *DEStech Transactions on Computer Science and Engineering*, Nr. aicae, Okt. 2019, doi: 10.12783/dtcse/aicae2019/31439.
- [10] R. S. Wallace, „The Anatomy of A.L.I.C.E.“, in *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, G. and B. G. Epstein Robert and Roberts, Hrsg. Dordrecht: Springer Netherlands, 2009, S. 181–210. doi: 10.1007/978-1-4020-6710-5_13.
- [11] M. das G. Bruno Marietto u. a., „Artificial Intelligence Markup Language: A Brief Tutorial“, *International Journal of Computer Science & Engineering Survey*, Bd. 4, Nr. 3, S. 1–20, Juni 2013, doi: 10.5121/ijcses.2013.4301.
- [12] „Siri“, 2022. <https://www.apple.com/siri/> (zugegriffen Juni 29, 2022).
- [13] „Amazon Alexa“, 2022. <https://developer.amazon.com/de-DE/alexa> (zugegriffen Juni 29, 2022).
- [14] „Google Assistant“, 2022. https://assistant.google.com/intl/de_de/ (zugegriffen Juni 29, 2022).

- [15] E. Adamopoulou und L. Moussiades, „An Overview of Chatbot Technology“, in *IFIP Advances in Information and Communication Technology*, 2020, Bd. 584 IFIP, S. 373–383. doi: 10.1007/978-3-030-49186-4_31.
- [16] U. Kamath, J. Liu, und J. Whitaker, *Deep learning for NLP and speech recognition*, Bd. 84. Springer, 2019. doi: <https://doi.org/10.1007/978-3-030-14596-5>.
- [17] K. R. Chowdhary, „Natural Language Processing“, in *Fundamentals of Artificial Intelligence*, K. R. Chowdhary, Hrsg. New Delhi: Springer India, 2020, S. 603–649. doi: 10.1007/978-81-322-3972-7_19.
- [18] S. Vijayarani, M. J. Ilamathi, und M. Nithya, „Preprocessing techniques for text mining-an overview“, *International Journal of Computer Science & Communication Networks*, Bd. 5, Nr. 1, S. 7–16, 2015.
- [19] S. Kannan u. a., „Preprocessing techniques for text mining“, *International Journal of Computer Science & Communication Networks*, Bd. 5, Nr. 1, S. 7–16, 2014.
- [20] A. K. Singh und M. Shashi, „Vectorization of text documents for identifying unifiable news articles“, *International Journal of Advanced Computer Science and Applications*, Bd. 10, Nr. 7, 2019.
- [21] H. Christian, M. P. Agus, und D. Suhartono, „Single document automatic text summarization using term frequency-inverse document frequency (TF-IDF)“, *ComTech: Computer, Mathematics and Engineering Applications*, Bd. 7, Nr. 4, S. 285–294, 2016.
- [22] V. Brühl, „Big Data, Data Mining, Machine Learning und Predictive Analytics: Ein konzeptioneller Überblick“, CFS Working Paper Series, 2019.
- [23] T. M. Mitchell, *Machine learning*, Bd. 1, Nr. 9. McGraw-hill New York, 1997.
- [24] J. Alzubi, A. Nayyar, und A. Kumar, „Machine learning from theory to algorithms: an overview“, in *Journal of physics: conference series*, 2018, Bd. 1142, Nr. 1, S. 012012.
- [25] Fumo D, „Types of Machine Learning Algorithms You Should Know“, Juni 15, 2017. <https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861> (zugegriffen Juli 07, 2022).
- [26] T. O. Ayodele, „Types of machine learning algorithms“, *New advances in machine learning*, Bd. 3, S. 19–48, 2010.
- [27] Gandhi R, „Support Vector Machine — Introduction to Machine Learning Algorithms“, Juni 07, 2018. <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47> (zugegriffen Juli 07, 2022).
- [28] Patel S, „Chapter 1 : Supervised Learning and Naive Bayes Classification — Part 1 (Theory)“, Apr. 30, 2017. <https://medium.com/machine-learning->

- 101/chapter-1-supervised-learning-and-naive-bayes-classification-part-1-theory-8b9e361897d5 (zugegriffen Juli 07, 2022).
- [29] S. Raschka, „Naive bayes and text classification i-introduction and theory“, *arXiv preprint arXiv:1410.5329*, 2014.
- [30] S. Tan, „An effective refinement strategy for KNN text classifier“, *Expert Systems with Applications*, Bd. 30, Nr. 2, S. 290–298, Feb. 2006, doi: 10.1016/J.ESWA.2005.07.019.
- [31] Asiri S, „Machine Learning Classifiers“, Juni 11, 2018. <https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623> (zugegriffen Juli 07, 2022).
- [32] A. M. Mahmood, N. Satuluri, und M. R. Kuppa, „An Overview of recent and traditional decision tree classifiers in machine learning“, *International Journal of Research and Reviews in Ad Hoc Networks*, Bd. 1, Nr. 1, S. 2011, 2011.
- [33] D. L. Olson und D. Delen, „Performance Evaluation for Predictive Modeling“, in *Advanced Data Mining Techniques*, D. L. Olson und D. Delen, Hrsg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 137–147. doi: 10.1007/978-3-540-76917-0_9.
- [34] M. Grandini, E. Bagli, und G. Visani, „Metrics for multi-class classification: an overview“, *arXiv preprint arXiv:2008.05756*, 2020.
- [35] S. Meshram, N. Naik, M. v R, T. More, und S. Kharche, „Conversational AI: Chatbots“, in *2021 International Conference on Intelligent Technologies (CO-NIT)*, 2021, S. 1–6. doi: 10.1109/CONIT51480.2021.9498508.
- [36] „What is Python? Executive Summary“, 2022. <https://www.python.org/doc/essays/blurbl/> (zugegriffen Juli 14, 2022).
- [37] „What is PostgreSQL?“, 2022. <https://www.postgresql.org/about/> (zugegriffen Juli 14, 2022).
- [38] „The Python SQL Toolkit and Object Relational Mapper“, 2022. <https://www.sqlalchemy.org/> (zugegriffen Juli 14, 2022).
- [39] „icalendar“, 2022. <https://icalendar.readthedocs.io/en/latest/about.html> (zugegriffen Juli 14, 2022).
- [40] „python-telegram-bot“, 2022. <https://github.com/python-telegram-bot/python-telegram-bot> (zugegriffen Juli 14, 2022).
- [41] „About pandas“, 2022. <https://pandas.pydata.org/about/index.html> (zugegriffen Juli 14, 2022).
- [42] „feedparser“, 2022. <https://feedparser.readthedocs.io/en/latest/> (zugegriffen Juli 14, 2022).
- [43] „Beautiful Soup“, 2022. <https://beautiful-soup-4.readthedocs.io/en/latest/> (zugegriffen Juli 14, 2022).

- [44] Stefan Luber und Nico Litzel, „Was ist Scikit-learn?“, Sep. 17, 2018. <https://www.bigdata-insider.de/was-ist-scikit-learn-a-756150/> (zugegriffen Juli 14, 2022).
- [45] „Facts & Figures“, 2022. <https://spacy.io/usage/facts-figures> (zugegriffen Juli 14, 2022).
- [46] „Bot API“, 2022. <https://core.telegram.org/api> (zugegriffen Juli 20, 2022).