



Hochschule Neu-Ulm  
University of Applied Sciences

Bachelorarbeit

im Bachelorstudiengang

**Wirtschaftspsychologie**

an der Hochschule für angewandte Wissenschaften Neu-Ulm

**Die Finalisierung eines Chatbot-Systems zur automatisierten  
Beantwortung von Studierendenfragen zu Studieninhalten**

Erstkorrektor: Prof. Dr. Sonja Köppl

Zweitkorrektor: Prof. Dr. Marlon Füller

Verfasser: Christoph Schultes (Matrikel-Nr.: 300911)

Thema erhalten: 17.04.2023

Arbeit abgeliefert: 24.07.2023

# Inhaltsverzeichnis

Abbildungsverzeichnis .....	III
Anhangsverzeichnis.....	IV
Abkürzungsverzeichnis .....	V
1 Einleitung .....	1
1.1 Motivation.....	1
1.2 Zielsetzung.....	1
1.3 Aufbau.....	2
2 Aktueller Stand des Chatbots .....	2
2.1 Grundlagen .....	2
2.1.1 Chatbot.....	2
2.1.2 Künstliche Intelligenz, Machine Learning und Künstliche Neuronale Netze.....	4
2.1.3 Natural Language Processing.....	8
2.2 Vorstellung .....	10
2.3 Nächste Schritte .....	11
3 Weiterentwicklung .....	11
3.1 Grafical User Interface.....	12
3.1.1 Modernes Interface.....	13
3.1.2 ChatBubbles.....	18
3.1.3 Aktualisierung der Fenstergröße.....	24
3.2 Retraining-Fragen abspeichern .....	27
3.2.1 Nicht verstandene Fragen.....	27
3.2.2 Falsch beantwortete Fragen .....	28
3.3 Bereitstellung der Applikation .....	29
3.3.1 .exe.....	30
3.3.2 .app .....	31
3.3.3 Ausgabe an die Studierende.....	31
4 Schlussbetrachtung.....	32
4.1 Zusammenfassung.....	32

4.2	Fazit .....	33
4.3	Ausblick.....	34
	Literaturverzeichnis.....	35
	Anhang .....	37
	Ehrenwörtliche Erklärung.....	97

## Abbildungsverzeichnis

Abbildung 1: Eigene Darstellung eines künstlichen Neurons nach (Boersch et al., 2007, S. 282) .....	6
Abbildung 2: Lineare Aktivitätsfunktion .....	7
Abbildung 3: GUI zum Zeitpunkt der Übernahme es HNU_Bots .....	11
Abbildung 4: Modernes HNU_Chatbot-Fenster ("Light"-Modus / "Dark"-Modus).....	14
Abbildung 5: HNU_Chatbot.py, Zeile 23-27, Fenstergröße und Icon.....	14
Abbildung 6: HNU_Chatbot.py, Zeile 58-63, CTKLable .....	15
Abbildung 7: HNU_Chatbot.py, Zeile 69-78, Scrollbar für Leinwand .....	15
Abbildung 8: HNU_Chatbot.py, Zeile 130-142, <code>_on_enter_pressed()</code> -Funktion und <code>send_message()</code> -Funktion.....	17
Abbildung 9: Tooltip für Icon .....	18
Abbildung 10: HNU_Chatbot.py, Zeile 12-16, konstante Variablen mit Farbdarstellung .....	19
Abbildung 11: HNU_Chatbot.py, Vergleich der <code>__init__()</code> -Funktion aus BotBubble-Klasse (links) und UserBubble-Klasse (rechts) .....	20
Abbildung 12: Skizze der Berechnung der Koordinaten für die Polygone der Sprechblasen .....	22
Abbildung 13: Einsatz der Gestaltprinzipien im GUI.....	23
Abbildung 14: HNU_Chatbot.py, Zeile 212-221, Aktualisieren der Positionen der Nutzersprechblasen auf der x-Achse innerhalb der Funktion <code>update_user_bubbles()</code> .....	25
Abbildung 15: HNU_Chatbot.py, Zeile 228-242, Aktualisieren des Abstands zwischen den Sprechblasen .....	26
Abbildung 16: Pop-up beim Schließen der Chatbot-App .....	27
Abbildung 17: Zusammenschnitt chat.py, <code>close_question_xlsx_files()</code> -Funktion und <code>save_unknown_questions()</code> -Funktion .....	28
Abbildung 18: HNU_Chatbot.py, Zeile 160-169, <code>antw_button_event()</code> -Funktion.....	29
Abbildung 19: Icon im Ordner .....	32

## Anhangsverzeichnis

Anhang 1: HNU_Chatbot.py.....	37
Anhang 2: HNU_Chatbot Icon.....	42
Anhang 3: chat.py.....	43
Anhang 4: Beispiel Chatverlauf.....	45
Anhang 5: auto-py-to-exe .....	46
Anhang 6: READ ME.txt .....	47
Anhang 7: install_dependencies.py.....	48
Anhang 8: intents_umlaute.json.....	49

## Abkürzungsverzeichnis

GB	Gigabyte
GUI	Grafical User Interface (Benutzeroberfläche)
KI	Künstliche Intelligenz
KNN	Künstliches Neuronales Netzwerk
MB	Megabyte
ML	Machine Learning
NLP	Natural Language Processing

# 1 Einleitung

## 1.1 Motivation

Vor allem mit der Veröffentlichung von ChatGPT durch OpenAI, hat das Thema Chatbots in der Gesellschaft einen großen Aufschwung bekommen und auch auf immer mehr Internetseiten begrüßt ein Chatbot die Nutzer\*. Sie stellen eine nahezu menschliche Konversation nach und können besonders einfache Fragen gut beantworten. Aus diesem Grund nutzen viele Online-Dienste Chatbots, um ihre Servicemitarbeiter zu entlasten. Die Zeit, die sie dadurch einsparen, können die Mitarbeiter auf schwierigere Probleme verwenden. Kunden mit leichten Fragen erhalten so eine schnelle Antwort durch den Chatbot, ohne lange in Telefonwarteschleifen warten zu müssen. Kunden mit schwierigeren Problemen erreichen wiederum schneller einen Servicemitarbeiter und werden ausführlicher betreut. Aufgrund dieser Vorteile kam die Idee auf, einen Chatbot zu entwickeln, der Fragen von Studierenden zu Vorlesungsinhalten beantworten kann. Die ehemaligen Studenten Kevin Knöpfle und Jonas Schiele haben daher in ihren Bachelorarbeiten bereits einen Prototypen für einen solchen Chatbot entwickelt. Der Prototyp soll in dieser Arbeit weiterentwickelt werden und anschließend den Studierenden zur Verfügung gestellt werden können.

## 1.2 Zielsetzung

Das Ziel in dieser Arbeit ist es, den Chatbot-Prototyp so weiterzuentwickeln, dass er an die Studierenden der Hochschule Neu-Ulm ausgegeben werden kann. Dazu soll zunächst eine neue grafische Benutzeroberfläche programmiert werden, die ein moderneres Aussehen hat und intuitiv in ihrer Handhabung ist. Nachdem der Chatbot an die Studierenden ausgegeben worden ist, soll es eine Möglichkeit geben, Rückmeldung von den Studierenden zur Qualität der Antworten des Bots zu erhalten. Um den Chatbot in Zukunft verbessern zu können, sollen Fragen, die vom Chatbot nicht verstanden wurden oder die er falsch beantwortet hat, gesammelt werden, um daran den Bot nachzutrainieren. Zuletzt ist es das Ziel dieser Arbeit, den Chatbot in eine Anwendungsdatei umzuwandeln, damit die Studierenden nicht immer den Programmcode ausführen müssen, um den Chatbot zu starten.

\* Aufgrund der besseren Lesbarkeit wird in dieser Bachelorarbeit das generische Maskulinum als geschlechtsneutrale Form verwendet. Die verwendeten Personenbezeichnungen beziehen sich immer – sofern nicht anders kenntlich gemacht – auf alle Geschlechter.

### **1.3 Aufbau**

Im folgenden Kapitel wird der aktuelle Stand des Chatbot-Prototypen zur Übernahme des Projekts beschrieben. Dazu werden im Abschnitt 2.1 die grundlegenden Begriffe Chatbot, Künstliche Intelligenz, Machine Learning, Künstliche Neuronale Netze und Natural Language Processing erläutert. Kapitel 2.2 befasst sich mit dem Chatbot-Prototypen selbst und es werden in Abschnitt 2.3 die nächsten Schritte erklärt, die bearbeitet werden müssen, um das Ziel dieser Arbeit zu erreichen. Kapitel 3 behandelt die Weiterentwicklung des Chatbots und ist in drei Teile aufgeteilt. Zunächst wird in Abschnitt 3.1 die Benutzeroberfläche der App modernisiert und mit der Einführung von Sprechblasen für die Chatnachrichten überarbeitet. In Kapitel 3.2 soll eine Möglichkeit geschaffen werden, nicht oder falsch beantwortete Fragen abzuspeichern. Abschließend wird in Kapitel 3.3 der Chatbot in eine Anwendungsdatei umgewandelt und Möglichkeiten zur Ausgabe dieser an die Studierenden diskutiert. Zum Ende der Arbeit wird in der Schlussbetrachtung von Kapitel 4 die Arbeit zusammengefasst und ein Fazit gezogen. Des Weiteren wird ein Ausblick formuliert, was zukünftig an dem Chatbot verbessert werden kann.

## **2 Aktueller Stand des Chatbots**

Zunächst werden wichtige Grundlagen und der aktuelle Stand zum Zeitpunkt der Übernahme des Chatbots beschrieben.

### **2.1 Grundlagen**

Im folgenden Abschnitt werden für die Arbeit und das Verständnis von Chatbots und ihre Funktionsweise wichtige Begriffe erklärt. Dazu wird zunächst auf Chatbots an sich eingegangen. Darauf aufbauend wird beschrieben, wie eine künstliche Intelligenz und maschinelles Lernen funktionieren und wie ein Chatbot durch dieses System Nutzereingaben verstehen und darauf antworten kann.

#### **2.1.1 Chatbot**

Ein Chatbot ist ein Programm, das natürlich gesprochene Sprache verstehen und eine passende Antwort dazu generieren kann. Als Bots, kurz für Roboter, können diese Sprach-Programme als künstliche Personen autonom mit anderen Systemen oder menschlichen Nutzern interagieren (vgl. Hoffmann, 2019, S. 19). Muldowney (vgl. 2017, Kapitel 2) definiert einen Chatbot als ein Computerprogramm, das Gespräche führt. Khan und Das (vgl. 2018, S. 1) weiten diese Definition aus und beschreiben einen Chatbot als ein Programm, das eine Nutzer-Eingabe als natürliche Sprache erhält, verarbeitet und eine passende Antwort dazu ausgibt. Die sprachliche Eingabe kann dabei verbal oder schriftlich getätigt werden, worauf der Bot dann eine Antwort generiert (vgl. Hoffmann, 2019, S. 21; vgl. Muldowney, 2017, Kapitel 1).

Als erster Chatbot gilt der Bot ELIZA, der im Jahr 1966 am Massachusetts Institute of Technology (MIT) entwickelt wurde (vgl. Hoffmann, 2019, S. 19; vgl. Khan & Das, 2018, S. 2; vgl. Muldowney, 2017, Kapitel 2). ELIZA ist programmiert worden, um als Psychotherapeut zu agieren und untersuchte die Eingaben der Nutzer auf Schlüsselwörter, die eine vorgegebene Antwort auslösen (vgl. Khan & Das, 2018, S. 2). ELIZA wurde entwickelt, bevor der erste persönliche Computer (PC) auf dem Markt vorgestellt wurde. Somit stand dieser Chatbot nicht zur freien Verfügung. Dennoch ist die Entwicklung der Technologie und Chatbots mit den Jahren weit vorangeschritten und so finden sich heutzutage verschiedene Chatbot-Systeme in alltäglichen Situationen wieder. In vielen Haushalten finden sich verbal gesteuerte Chatbot-Systeme, wie Siri von Apple, Google Assistant von Google oder Alexa von Amazon, die als unterstützende Software die Nutzer begleiten (vgl. Hoffmann, 2019, S. 22; vgl. Khan & Das, 2018, S. 2). Doch nicht nur im privaten Umfeld tauchen Chatbot-Systeme auf. Auch viele Unternehmen nutzen Chatbots beispielsweise auf ihren Internetseiten oder um Anrufe von Kunden entgegenzunehmen. Hierbei haben Chatbots vielseitige Anwendungsmöglichkeiten. Unter anderem können sie zu Unterhaltungszwecken, zum Marketing oder auch im Kundenservice eingesetzt werden. Besonders geeignet sind Chatbots, um das Kundenerlebnis zu verbessern. Zum Beispiel können Kunden, um Informationen zu einem Produkt zu erhalten, ganz einfach den Chatbot fragen, anstatt selbst die Internetseite nach diesen Informationen zu durchsuchen (vgl. Muldowney, 2017, Kapitel 1). Der Vorteil dabei ist, dass die Suche dabei eher wie das Gespräch mit einem Freund empfunden wird, als wie eine reine Eingabe von Suchbegriffen (vgl. Khan & Das, 2018, S. 7). Zudem steht der Chatbot jederzeit zur Verfügung (vgl. Hoffmann, 2019, S. 11) und kann besonders gut einfache Kundenanfragen beantworten, während komplexe Probleme weiterhin von Kundenservice-Mitarbeitern übernommen werden können (vgl. Muldowney, 2017, Kapitel 5). Ein Chatbot ist hierbei günstiger als menschliche Mitarbeiter, hat geringere Ausfallzeiten, da er nicht krank werden kann und gibt schnellere Antworten, was zu geringeren Wartezeiten beim Kunden in der Warteschleife führt (vgl. Muldowney, 2017, Kapitel 2).

So unterschiedlich wie die Stimmen verschiedener Chatbots in Telefonwarteschleifen sind, so divers ist auch ihr Erscheinen als textbasiertes Programm. Dennoch weist der Großteil der Chatbots Ähnlichkeit zueinander auf. Das Interface, also die Benutzeroberfläche von Chatbots, besteht meist aus Sprechblasen, die rechts- und linksbündig angeordnet sind, um besser zwischen eigenen Eingaben und den Antworten des Chatbots unterscheiden zu können. Das Konzept der Sprechblasen stammt dabei in leicht abgewandelter Form von Jens Alfke aus dem Jahr 1997 (vgl. Hoffmann, 2019, S. 29). Als zentraler Treiber dieser Darstellung von einer Konversation im schriftlichen Kontext diente aber vor allem die Applikation WhatsApp (vgl. Hoffmann, 2019, S. 27; vgl. Khan & Das, 2018, S. 98).

Neben der für den Nutzer sichtbaren Benutzeroberfläche, stecken hinter einem Chatbot Algorithmen, die für den Nutzer nicht erkennbar sind. Die Basis, auf der ein Chatbot mit dem Nutzer interagiert, lässt sich aufgrund dieser Prozesse in drei Kategorien unterteilen.

**Regelbasierte Chatbots:** Für regelbasierte Chatbots werden bei der Programmierung Regeln aufgestellt, nach denen der Chatbot arbeitet. Diese Regeln lassen sich wie eine Art Baumdiagramm verstehen, nach dem der Nutzer aus verschiedenen vorgegebenen Fragen wählen kann, die er dem Chatbot stellt und schließlich die dazu ausformulierte Antwort erhält (vgl. Hundertmark, 2021, S. 159). Besonders gut eignet sich diese Variante eines Chatbots zum Beispiel bei Kundenbefragungen, bei denen immer die gleichen Fragen gestellt werden und der Kunde zwischen vorgegebenen Antwortmöglichkeiten wählen kann. Die Gestaltung der Umfrage mit einem Chatbot erzeugt hierbei ein interaktives, spannenderes Kundenerlebnis im Vergleich zum einfachen Fragebogen (vgl. Hundertmark, 2021, S. 159).

**KI-basierte Chatbots:** Gegenüber den regelbasierten Chatbots steht die Möglichkeit eine künstliche Intelligenz (KI) die freie Eingabe des Nutzers auf ihren Inhalt analysieren zu lassen und eine passende Antwort dazu zu generieren (vgl. Hundertmark, 2021, S. 159). Hierfür benötigt der Chatbot ein System zur Verarbeitung natürlicher Sprache, die die Eingabe des Nutzers analysiert, um die Intention dahinter herauszufinden, damit der Chatbot entsprechend auf die Frage antworten kann (vgl. Hundertmark, 2021, S. 160). Auch die Antwort ist hierbei nicht vom Programmierer fest vorgeschrieben, sondern wird vom Chatbot generiert. Um dem System beizubringen, wie es die Fragen interpretieren und Antworten generieren kann, werden Verfahren des maschinellen Lernens genutzt (vgl. Hundertmark, 2021, S. 160).

**Mischform:** Als dritte Kategorie gibt es eine Mischform aus beiden Chatbot-Varianten. In der Beschreibung nach Hundertmark (vgl. 2021, S. 163) hat der Nutzer des Chatbots die Möglichkeit Freitexte als Frage einzugeben, die dann von der KI interpretiert werden, oder auf vordefinierte Fragen zurückzugreifen, die dem regelbasierten Ansatz des Chatbots folgen. Daraufhin kann der Chatbot mit einer vordefinierten Antwort reagieren, oder selbstständig eine Antwort erstellen.

Das Chatbot-System, das in der vorliegenden Arbeit weiterentwickelt wird, agiert in einer Version der Mischform. Nutzer können ihre Fragen frei formulieren und bekommen keine möglichen Fragen vorgegeben. Der Chatbot interpretiert die Fragen mithilfe einer künstlichen Intelligenz und gibt eine, zur herausgefundenen Intention passende, vordefinierte Antwort wieder.

### **2.1.2 Künstliche Intelligenz, Machine Learning und Künstliche Neuronale Netze**

Unter künstlicher Intelligenz werden in der Informatik allgemein Systeme verstanden, die auf Grundlage von Regeln oder Wissen, nach denen gearbeitet werden soll, mit der Umwelt interagieren (vgl. Muldowney, 2017, Kapitel 4). Schon zu Beginn der Entwicklung und Erforschung künstlicher Intelligenz konnten Computer für Menschen komplizierte Aufgaben auf Grundlage

festgelegter Regeln in kürzester Zeit lösen (vgl. Goodfellow et al., 2016, S. 1). Hingegen waren Aufgaben, die für Menschen intuitiv und leicht lösbar sind, jedoch sich schwer beschreiben und in Regeln zusammenfassen lassen, eine große Herausforderung für die KI (vgl. Goodfellow et al., 2016, S. 1). Beispiele hierfür sind das Erkennen von Gesichtern oder auch die Erkennung von Worten und deren Bedeutung in einem Satz. Damit ein Programm wie ein Chatbot diese Aufgabe bewältigen kann, muss es trainiert werden, damit es lernt, Intentionen aus Texten herauszulesen und entsprechend eine Antwort zu geben. Dieser Zweig der KI nennt sich maschinelles Lernen (in Englisch Machine Learning) und entwickelte sich Anfang der 2000er Jahre (vgl. Buduma & Locascio, 2017, S. 4; vgl. Hoffmann, 2019, S. 116; vgl. Khan & Das, 2018, S. 5). Der Computer bekommt im Machine Learning (ML) Daten vorgelegt, auf die er reagiert. Je nach Art des Lernmodells erhält der Computer eine Rückmeldung auf das gezeigte Verhalten, wodurch der Computer seine Reaktion evaluiert und anpasst (vgl. Boersch et al., 2007, S. 274; vgl. Goodfellow et al., 2016, S. 2). Boersch et al. (2007, S. 274) beschreiben, dass ein Computerprogramm „aus einer Erfahrung  $E$  bezüglich einer Aufgabenklasse  $T$  und einer Bewertung  $P$  [lernt], wenn sich durch  $E$  seine durch  $P$  gemessene Leistung bei Aufgaben aus  $T$  verbessert“. Hierbei kann in drei Lernparadigmen unterschieden werden:

**Überwachtes Lernen (supervised learning):** Beim überwachten Lernen wird ein Algorithmus anhand von Trainingsdaten trainiert. Aufgabe ist es dabei jeden Eingabewert in die richtige Ausgabekategorie einzuordnen. Im Regelfall wird dieser Datensatz in einen Trainingsdatensatz zum Trainieren der KI und einen Testdatensatz, um die Qualität der trainierten KI zu überprüfen, aufgeteilt. Dabei entwickelt das Programm eigene Regeln, wie es die Eingaben zuordnet und lernt so die Zusammenhänge zwischen den Merkmalen der Daten in Bezug auf die Kategorien (vgl. Boersch et al., 2007, S. 275; vgl. Hirschle, 2022, S. 17).

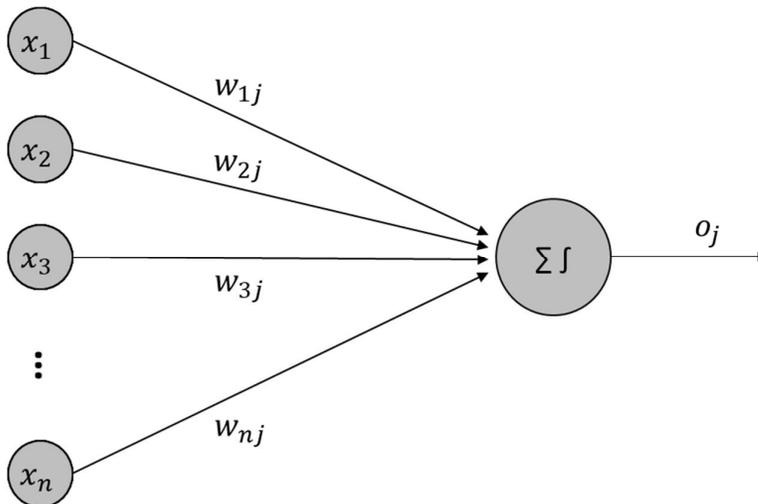
**Bestärkendes Lernen (reinforcement learning):** Anstatt dem System die richtigen Lösungen vorzugeben, erhält das System beim bestärkenden Lernen positive oder negative Rückmeldungen für sein Verhalten (vgl. Boersch et al., 2007, S. 276; vgl. Rey & Wender, 2018, S. 28). Es werden zunächst willkürlich Aktionen ausgeführt, auf die je nachdem, wie zielführend diese waren, das System belohnt oder bestraft wird, häufig in Form eines Punktsystems (vgl. Boersch et al., 2007, S. 277). Aktionen, für die das Programm belohnt wird, werden daraufhin häufiger ausgeführt und erhalten wiederum eine Rückmeldung, bis das gewünschte Verhalten antrainiert wurde. Während des Lernverfahrens kann das Programm zwischen den Möglichkeiten der Exploration, neue Wege zu gehen, um das Ziel zu erreichen und der Exploitation, die gesammelten Erfahrungen zum Lösen der Aufgabe zu nutzen, wählen (vgl. Boersch et al., 2007, S. 277).

**Unüberwachtes Lernen (unsupervised learning):** Anders als beim überwachten Lernen, liegen beim unüberwachten Lernen nicht die Zieldaten vor, sondern es ist die Aufgabe des

Algorithmus, selbstständig, ohne des Feedback eines Lehrers, Muster in den Daten zu erkennen und diese in neue Kategorien zu clustern (vgl. Boersch et al., 2007, S. 278–279; vgl. Hirschle, 2022, S. 17).

Die künstliche Intelligenz hinter dem in dieser Arbeit beschriebenen Chatbot basiert auf einem künstlichen neuronalen Netzwerk (KNN), das durch das überwachte Lernen trainiert wurde. Inspiriert durch die Weiterleitung und Verarbeitung von Reizen durch das Gehirn, sind künstliche neuronale Netzwerke aus mehreren künstlichen Neuronen aufgebaut (vgl. Rey & Wender, 2018, S. 16). Im KNN sind diese künstlichen Neuronen in Schichten angeordnet. Es wird dabei unterschieden in eine Eingabeschicht (Input-Layer), die eine Eingabe von außen aufnehmen und weiterleiten kann, und eine Ausgabeschicht (Output-Layer), die die Antwort des Netzes zurück an die Umwelt gibt (vgl. Boersch et al., 2007, S. 282; vgl. Rey & Wender, 2018, S. 15, vgl. 2018, S. 15–16). Ein KNN besteht mindestens aus diesen beiden Typen von Neuronen. Dazwischen können beliebig viele weitere versteckte Schichten (Hidden-Layer) liegen (vgl. Boersch et al., 2007, S. 282; vgl. Rey & Wender, 2018, S. 17).

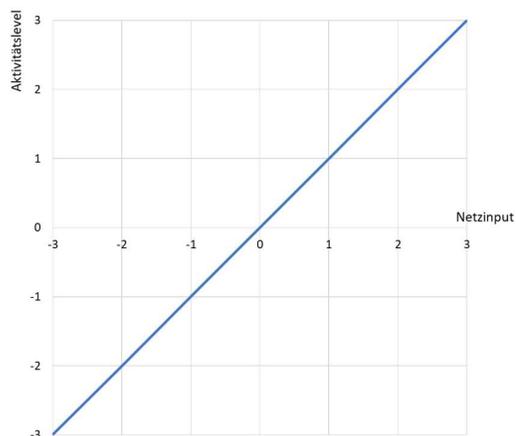
Die Schichten bestehen aus einzelnen künstlichen Neuronen, welche mit der vorherigen Schicht über Kanten (Links) verbunden sind (vgl. Rey & Wender, 2018, S. 17). Diese Verbindungen können unterschiedlich stark sein, was durch ein Gewicht  $w_i$  ausgedrückt wird (vgl. Buduma & Locascio, 2017, S. 11). Je stärker das Gewicht, also je positiver oder negativer der Wert, desto mehr Einfluss hat die Verbindung auf andere, denn durch diese Gewichtung speichert das KNN das erlernte Wissen (vgl. Rey & Wender, 2018, S. 17).



**Abbildung 1: Eigene Darstellung eines künstlichen Neurons nach (Boersch et al., 2007, S. 282)**

Dieses Wissen des KNN wird folgendermaßen auf die Eingabe in das Netzwerk angewandt. Zunächst werden Informationen als Zahlenwerte an die Eingabeschicht gegeben, die diese an die künstlichen Neuronen der nächsten Schicht verteilt, wobei allerdings noch kein Gewicht

angebracht wird (vgl. Rey & Wender, 2018, S. 18). Das Gewicht kommt zwischen den folgenden Schichten zum Tragen. Ein künstliches Neuron erhält wie in Abbildung 1 ersichtlich Eingabewerte  $x_i$  von vorgeschalteten künstlichen Neuronen. Die Eingabewerte werden mit dem Gewicht der jeweiligen Kanten  $w_{ij}$  multipliziert und im Kern des künstlichen Neurons aufaddiert ( $\Sigma$ ) (vgl. Rey & Wender, 2018, S. 19). Überschreitet die Summe einen vorgegebenen Schwellenwert  $\theta$ , wird ein Ausgabewert  $o_j$  in entsprechender Höhe ausgegeben (vgl. Boersch et al., 2007, S. 282–283). Wann ein Schwellenwert überschritten wird, hängt von der verwendeten Aktivitätsfunktion ab (vgl. Boersch et al., 2007, S. 283). Für die Aktivitätsfunktion kann der Programmierer aus mehreren Funktionen mit unterschiedlichen Formen wählen. Dabei wird meist eine Aktivitätsfunktion für alle künstlichen Neuronen aufgestellt (vgl. Rey & Wender, 2018, S. 20). Ein Beispiel für eine Aktivitätsfunktion wäre die Lineare Aktivitätsfunktion (Abb. 2). Hierbei gibt es einen linearen Zusammenhang zwischen der Eingabe in das künstliche Neuron und dem Aktivitätslevel, das an die folgende Schicht weitergegeben wird (vgl. Rey & Wender, 2018, S. 20). Weitere mögliche Aktivitätsfunktionen können Formen einer binären, logistischen oder normalverteilten Funktion abbilden. Mit der Hilfe der Aktivitätsfunktion wird schließlich die Ausgabe des künstlichen Neurons bestimmt und an die nächste Schicht im KNN weitergegeben (vgl. Rey & Wender, 2018, S. 26). Handelt es sich schon um die Ausgabeschicht, wird das Signal als Antwort zurück an die Umwelt gegeben und kann interpretiert werden.



**Abbildung 2: Lineare Aktivitätsfunktion**

Für das Trainieren eines künstlichen neuronalen Netzes wird ein Datensatz genutzt, der einen „bestimmten Ausschnitt der Realität“ (Rey & Wender, 2018, S. 28) abbildet. Dieser Datensatz wird aufgeteilt in einen Trainingsdatensatz und einen Testdatensatz. Die Trainingsdaten werden dem KNN beim supervised learning mit der Aufgabe gegeben, diese den richtigen Kategorien zuzuordnen. Zu Beginn haben dabei die Gewichte der künstlichen Neuronen den gleichen Wert. Aufgrund der Rückmeldung, ob das KNN die Daten richtig zugeordnet hat, passt

der Trainingsalgorithmus die Gewichte der künstlichen Neuronen an, um näher an die richtige Lösung zu kommen. Dieses Verfahren wird fortgesetzt, bis das KNN gelernt hat, die Trainingsdaten den richtigen Kategorien zuzuordnen.

Dies kann verschiedenen Herausforderungen für die Qualität der KI mit sich bringen. Nachdem ein KNN trainiert wurde, kann es im Testzyklus getestet werden, indem der KI Reize zugeführt werden und die Antwort auf Richtigkeit geprüft wird (vgl. Rey & Wender, 2018, S. 29). Zunächst besteht die Möglichkeit des Training Errors. Der Training Error ist der Fehler, den das Programm beim Kategorisieren der Trainingsdaten macht und wird durch den Lernprozess immer weiter vermindert (vgl. Goodfellow et al., 2016, S. 107). Aber die KI soll am Ende nicht nur die Trainingsdaten richtig zuordnen können, sondern auch die Testdaten. Der Fehler, den die KI beim Zuordnen der Testdaten macht, wird Test Error genannt (vgl. Goodfellow et al., 2016, S. 107). Um den Training Error zu minimieren, muss die künstliche Intelligenz lange genug trainiert werden. Ist der Fehlerwert zu hoch, wird dies als Underfitting bezeichnet (vgl. Goodfellow et al., 2016, S. 108). Beim Underfitting hat das Modell die verschiedenen Eigenheiten, die die Kategorien ausmachen, nicht richtig gelernt. Wird das Modell jedoch zu viel trainiert, kommt es zum Overfitting, was daraus resultiert, dass das Modell die individuellen Eigenheiten der Trainingsdaten auswendig gelernt hat, diese jedoch nicht generalisierbar sind und so bei den Testdaten Fehler in der Zuordnung hervorbringen (vgl. Goodfellow et al., 2016, S. 108). Wann das Training beendet werden sollte, damit das Underfitting überwunden wurde, es aber noch nicht zum Overfitting kommt, muss der Programmierer selbst entscheiden.

### **2.1.3 Natural Language Processing**

Machine Learning Algorithmen benötigen eine numerische Eingabe, damit ein Modell trainiert werden kann. Damit der Computer die natürliche Sprache verarbeiten kann, muss diese zunächst in Zahlenwerte umgewandelt werden. Das Natural Language Processing (NLP) ist ein stark von der Linguistik geprägter Teil der Informatik, der Texte und Sprache in Zahlen umstrukturiert, sodass ein Computer mit ihnen arbeiten kann (vgl. Hoffmann, 2019, S. 111). Die Herausforderung dabei ist, dass die Bedeutung eines Satzes meist nicht in einzelnen Worten zu finden ist, sondern aus dem Zusammenhang mehrerer Wörter erschlossen wird (vgl. Hoffmann, 2019, S. 113). Dennoch müssen für die Verarbeitung, Sätze immer auf eine Wörterebene zerlegt werden (vgl. Hirschle, 2022, S. 8). Hierbei hilft das Verfahren der Wort-Level-Tokenisierung, das aus einem Satz eine Liste an Wörtern und Satzzeichen erstellt. Beispielsweise der Satz „Heute ist es warm.“ wird dabei in die Liste [„Heute“, „ist“, „es“, „warm“, „.“] umgewandelt. Den einzelnen Wörtern können anschließend im One-Hot-Encoding-Verfahren die kategorialen Werte 0 und 1 zugeordnet werden. Hierfür wird eine Matrix erstellt, in der jede Spalte ein Exemplar der vorkommenden Wörter des Textes repräsentiert (vgl. Hirschle, 2022, S. 43). Diese Gesamtheit an Wörtern wird als Vokabular bezeichnet und kann sich aus Wörtern aus mehreren Texten zusammensetzen. In der Matrix werden nun die Positionen der Wörter

mit einer 1 markiert, wenn das Wort im untersuchten Satz vorkommt. Für die beiden Beispielsätze „Heute ist es warm“ und „Morgen soll es regnen“, sieht die entsprechende Matrix wie folgt aus:

<b>Satz</b>	<b>es</b>	<b>heute</b>	<b>ist</b>	<b>morgen</b>	<b>regnen</b>	<b>soll</b>	<b>warm</b>
<b>1</b>	1	1	1	0	0	0	1
<b>2</b>	1	0	0	1	1	1	0

Dabei wird das Vokabular alphabetisch geordnet und alle Buchstaben klein geschrieben. Eine Variante dieses Ansatzes ist das Bag-of-Words-Modell, bei dem nicht nur eine 1 eingetragen wird, wenn das Wort vorkommt, sondern, wenn das Wort mehrmals im untersuchten Abschnitt vorkommt, dieses mit der Häufigkeit des Erscheinens in der Matrix eingetragen wird (vgl. Hirschle, 2022, S. 44). Wie vorausszusehen ist, bekommt diese Matrix bei sehr langen Texten viele Spalten. Bei einem zentralen Wortschatz von rund 70.000 Wörtern in der deutschen Sprache (vgl. Güll, 2010), kann das Training eines statistischen Modells mit entsprechend vielen Variablen sehr zeitaufwändig werden. Daher wird sich mehreren Möglichkeiten bedient, wie die Anzahl an unterschiedlichen Wörtern reduziert werden kann. Einerseits hilft hierbei schon das Kleinschreiben sämtlicher Buchstaben. Des Weiteren können für den Inhalt irrelevante Wörter, sogenannte Stopwords, wie beispielsweise „aber“, „die“, „man“, „mit“ oder „sei“, entfernt werden. Als drittes werden Wörter noch auf ihren Wortstamm oder Grundform reduziert (vgl. Hirschle, 2022, S. 44). Hierfür gibt es zwei mögliche Verfahren. Einerseits kann das Stemming genutzt werden, um Affixe vom Wortende zu entfernen. Aus "laufen", "gelaufen" und "läuft" wird der Stamm "lauf" gebildet. Dieses Verfahren ist natürlich nicht perfekt, da es zu Missverständnissen führen kann, wenn Wörter aus unterschiedlichen Kategorien auf den gleichen Stamm reduziert werden, wie es bei „Stift“ und „stiften“ der Fall sein kann, wenn bei beiden der Stamm „stift“ gebildet wird. Daher bietet die Lemmatisierung eine Alternative für das Stemming-Verfahren. Bei der Lemmatisierung wird jedes Wort auf seine Grundform (Lemma) reduziert (vgl. Hirschle, 2022, S. 44). Das ist besonders bei unregelmäßigen Verben hilfreich, die durch das Stemming häufig nicht auf das Ursprungswort zurückzuführen sind. Dieses Verfahren ist jedoch sehr aufwändig, da jedes Wort zunächst in einem Wörterbuch nachgeschlagen werden muss, um die Grundform herauszufinden (vgl. Hirschle, 2022, S. 45). Mithilfe dieser drei Möglichkeiten, die Anzahl an Wörtern und damit die Anzahl der Spalten in der Matrix zu reduzieren, führt das Bag-of-Words-Verfahren oder das One-Hot-Encoding zu einer Matrix, in der nur noch Zahlenwerte enthalten sind, die vom KNN verarbeitet werden können.

## 2.2 Vorstellung

Diese Arbeit baut auf den Bachelorarbeiten von Kevin Knöpfle und Jonas Schiele auf, die den Prototypen eines Chatbots entwickelt haben. Im folgenden Abschnitt soll der Status dieses Prototypens beschrieben werden.

Der „HNU\_Bot“ basiert auf der Programmiersprache Python. Die Applikation selbst wird mit Ausführen der Datei „app.py“ geöffnet und benötigt die Dateien „chat.py“, „nltk\_utils.py“, „model.py“, „intents.json“ und „data.pth“, um zu starten. Die Gewichte des trainierten Modells, das hinter dem Chatbot steht, ist dabei in „data.pth“ gespeichert und wird in „chat.py“ zusammen mit dem KNN aus „model.py“ und Funktionen zur Verarbeitung von natürlicher Sprache aus „nltk\_utils.py“ aufgerufen, um die Eingaben des Nutzers zu analysieren, die Intention des Nutzers zu identifizieren und schließlich eine geeignete Antwort zurückzugeben. Dabei sind die verschiedenen Intentionen als Tags mit möglichen Fragen des Nutzers (Pattern) vorgegebenen Antworten (Response) in „intents.json“ gespeichert.

Inhaltlich basiert der Chatbot auf den Kontrollfragen zu den Vorlesungen „Big Data und Data-Mining“, „Programmierung“ und „Informationssicherheit“ und umfasst 180 Tags, die jeweils aus mindestens einem Pattern und einer Response bestehen (vgl. Schiele, 2023, S. 46–47). In der Analyse der Nutzereingabe erfolgt zunächst die Word-Level-Tokenisierung und das Stemming. Danach werden Stopwords entfernt und die übrigbleibenden Wortstämme im Bag-of-Words vermerkt (vgl. Schiele, 2023, S. 47–48). Mit dem Bag-of-Words als Eingabe kann das KNN die Intention des Nutzers schätzen. Das KNN gibt dabei einen Prozentwert aus, wie sicher es sich ist, dass der Nutzer mit seiner Frage ein bestimmtes Tag meinte. Überschreitet dieser Wert die minimale Wahrscheinlichkeit von 60%, gibt der Chatbot eine entsprechende Antwort aus. Mit diesem Verfahren liegt der Chatbot bei der Identifizierung der Intention in den von Schiele durchgeführten Tests zu 90-100% richtig (vgl. 2023, S. 49).

Für die Benutzeroberfläche, bzw. Grafical User Interface (GUI), wurde eine dunkle Darstellungsweise gewählt. Im unteren Teil des Fensters ist ein Eingabefeld, in dem der Nutzer seine Frage tippen kann, mit einem Button daneben, um die Nachricht abzusenden (siehe Abb. 3). Darüber werden die Fragen und Antworten in einem Textfeld mit dem jeweiligen Absender abgebildet. Bei dieser Art der Darstellung werden jedoch Wörter, die über den Fensterrand hinausgehen würden, mit einem Zeilenumbruch in die nächste Reihe geschoben, wie in der ersten Zeile der zweiten Antwort des Chatbots bei dem Wort „Ungleich“ zu sehen. Abschnitte werden in den Antworten durch „|“ getrennt und werden dadurch besonders bei längeren Antworten unübersichtlich. Zudem werden Umlaute noch als „ae“, „oe“ und „ue“ und das „ß“ als „ss“ ausgegeben.

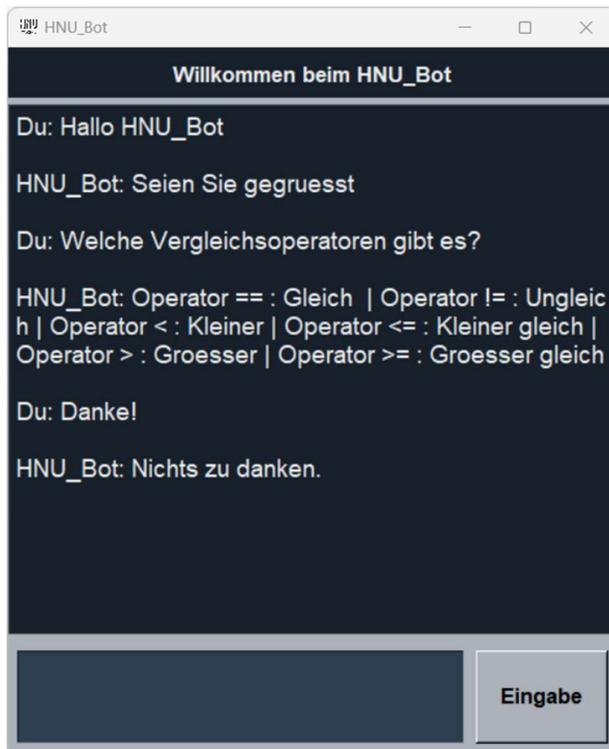


Abbildung 3: GUI zum Zeitpunkt der Übernahme es HNU\_Bots

### 2.3 Nächste Schritte

Der HNU\_Bot kann den Tests zufolge verschiedene Fragen zu Vorlesungsinhalten souverän beantworten. Bevor der Chatbot aber an die Studierenden ausgegeben werden kann, sollen noch einige Punkte vorher bearbeitet werden, um der App den letzten Schliff zu geben. So soll in dieser Arbeit die Benutzeroberfläche des Chatbots in ein modernes GUI im Messenger-App-Stil mit Sprechblasen umgewandelt werden. Zudem soll sich die Möglichkeit offengehalten werden, nach der Übergabe des Chatbots an die Studierenden das KNN nachtrainieren zu können. Hierfür sollen Frage der Studierenden, die der HNU\_Bot nicht versteht, von ihm abgespeichert werden, um in Zukunft als Vorlage für eine weitere Trainingsrunde zu dienen. Aktuell muss jeder, der den Chatbot nutzen möchte, das zugehörige Python-Skript ausführen, um die App zu starten. Hierbei kann es zu Problemen kommen, wenn zum Beispiel die nötigen Bibliotheken nicht installiert sind. Daher soll eine Applikation erstellt werden, die von den Studierenden ausgeführt werden kann, ohne auf Python zurückgreifen zu müssen. Abschließend werden noch Überlegungen angestellt, wie der Chatbot an die Studierenden ausgegeben werden kann.

## 3 Weiterentwicklung

Folgend wird die Fertigstellung des Chatbots beschrieben. Dabei soll zunächst die Entwicklung einer neuen Benutzeroberfläche dargestellt werden. Im nächsten Abschnitt wird vorgestellt,

wie Fragen, die der Chatbot nicht richtig verstanden hat, gesammelt werden können. Abschließend wird sich mit der Erstellung einer App befasst, bei der es nicht mehr nötig ist, den Chatbot über Python zu öffnen.

### 3.1 Grafical User Interface

Welche Regeln hinter Darstellungsmöglichkeiten von Kommunikation im visuellen Raum stecken, untersucht die Gestaltpsychologie. Die Gestaltpsychologie beschäftigt sich mit der Frage, wie Menschen Ordnung in der Welt wahrnehmen und aus visuellen Stimuli Ordnung schaffen (vgl. Nussbaumer Knafllic, 2017, S. 63). In Form von Gestaltprinzipien ist festgelegt, wie wahrgenommene Objekte in einer Szene zueinander in Relation gesetzt werden (vgl. Goldstein, 2015, S. 102). Im Folgenden werden vier Prinzipien vorgestellt, die bei der Gestaltung eines Chatbot-Interfaces beachtet werden sollten.

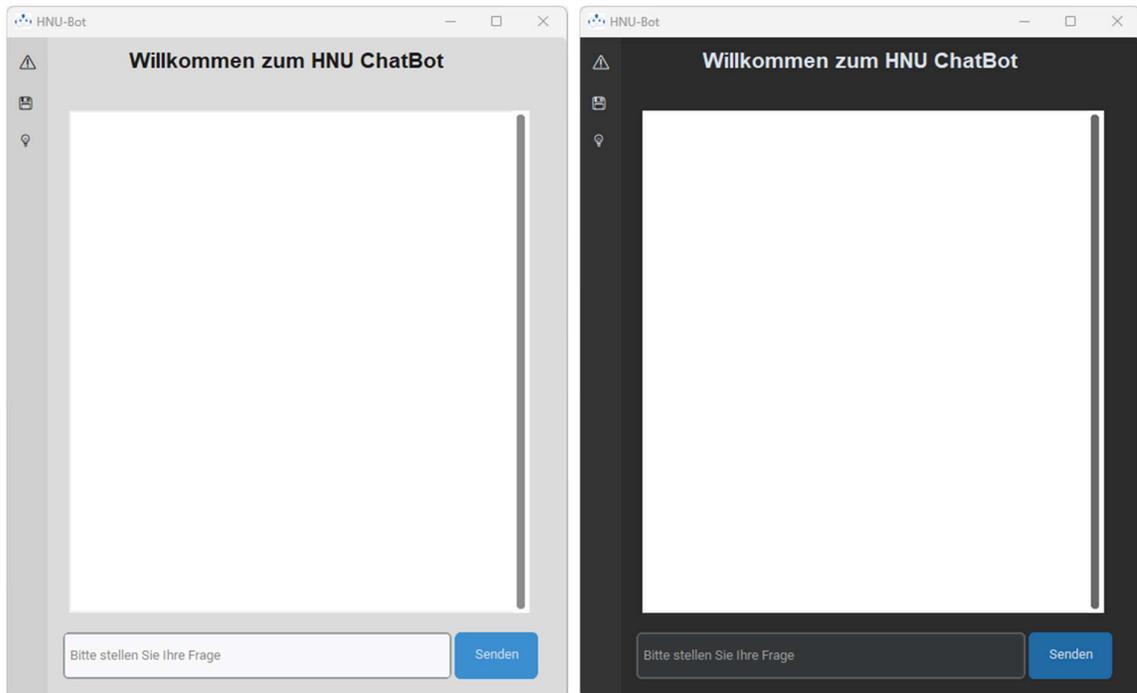
- **Nähe:** Das Prinzip der Nähe besagt, „Dinge, die sich nahe beieinander befinden erscheinen als zusammengehörig“ (Goldstein, 2015, S. 104; vgl. Nussbaumer Knafllic, 2017, S. 63). Im Beispiel einer Messenger-App lässt sich dieses Prinzip beispielsweise an Eingabefeld und Senden-Button beschreiben. Dadurch, dass diese beiden Elemente der App nahe beieinander dargestellt werden, werden diese auch als einander angehörig wahrgenommen. Die Oberfläche wirkt somit intuitiv, wenn es an das Senden von Nachrichten geht.
- **Ähnlichkeit:** Ähneln sich Objekte in ihrer Größe, Form, Farbe oder Ausrichtung, werden diese als eine Gruppe wahrgenommen (vgl. Goldstein, 2015, S. 103; vgl. Nussbaumer Knafllic, 2017, S. 64). So werden in vielen Messenger-Apps beispielsweise die Sprechblasen des Nutzers und die des Kommunikationspartners in unterschiedlichen Farben abgebildet, was die Zuordnung zum Verfasser ermöglicht, ohne einen Namen mit in der Sprechblase zu integrieren.
- **Gemeinsame Region:** Das gleiche Beispiel lässt sich für das Prinzip der gemeinsamen Region anbringen. Dieses besagt, "Elemente, die innerhalb einer gemeinsamen Region liegen, werden zusammengruppiert" (Goldstein, 2015, S. 104). Sprechblasen in Nachrichten-Apps sind somit meist rechts bzw. links am Bildschirmrand angegliedert, je nachdem, ob es sich um Nachrichten vom Nutzer oder vom Kommunikationspartner handelt, um in die jeweilige Kategorie eingeordnet zu werden.
- **Kontinuität:** Rückt eine Nachricht oben aus dem Bildschirm raus, greift das Prinzip der Kontinuität. Denn wenn eine Form teilweise verborgen ist, nehmen Menschen sie dennoch als vollständig wahr (vgl. Nussbaumer Knafllic, 2017, S. 66). Auch wenn dieses Prinzip wohl eine eher schwächere Rolle in der Benutzeroberfläche eines Chatbots einnimmt, ist es trotzdem in der Darstellung präsent.

Damit der Chatbot eine neue Benutzeroberfläche erhält, die den Regeln der Gestaltpsychologie folgt, wurde die neue Syntax-Datei „HNU\_Chatbot.py“ erstellt, die in Anhang 1 zu finden ist. Die Bearbeitung der GUI teilt sich dabei in drei Teile auf. Zunächst wird die generelle Überarbeitung der Benutzeroberfläche hin zu einem moderneren Design beschrieben. Hierzu werden Code-Zeilen, die nicht für die Aktualisierung des Designs oder den Funktionen hinter den Widgets zuständig sind, übersprungen und im späteren Verlauf des Kapitels erläutert. Für den Chatbot soll ein Messenger-App-ähnliches Design genutzt werden, das, anstelle des Textfeldes das der Prototypen nutzt, die Nachrichten durch Sprechblasen abbildet. Wie diese Sprechblasen für Nutzer und Bot abgebildet werden, wird im zweiten Teilkapitel beschrieben. Zuletzt wird auf die Herausforderung eingegangen, wie die richtige Darstellung des GUI gewährleistet werden kann, wenn die Größe des Fensters verändert wird.

### **3.1.1 Modernes Interface**

Die grafische Benutzeroberfläche macht es dem Nutzer möglich, mit dem HNU\_Bot zu interagieren und seine Fragen zu stellen. Daher sollte das Interface übersichtlich und intuitiv sein und eine positive Nutzererfahrungen bereitstellen. Um das zu erreichen, wurde das GUI in dieser Arbeit modernisiert. Das Interface des Chatbot-Prototyps basiert auf der Python-Bibliothek Tkinter. Da Applikationen mit dieser Bibliothek ein veraltetes Design haben, wird zur Modernisierung des Aussehens die Bibliothek CustomTkinter, die von Tom Schimansky entwickelt wurde, verwendet. CustomTkinter baut auf der Tkinter-Bibliothek auf und nutzt für die Bezeichnung der Widgets ähnliche Namen zu denen aus Tkinter (vgl. Schimansky, o. J.). So wird zum Beispiel das Äquivalent zum Widget „Frame“ aus Tkinter in CustomTkinter „CTKFrame“ genannt. Diese Ähnlichkeit vereinfacht die Implementierung einer modernen Benutzeroberfläche und macht die Bibliothek damit zu dem geeigneten Modul für diese Aufgabe.

Für die moderne Benutzeroberfläche wurde eine neue Syntax unter dem Namen „HNU\_Chatbot.py“ erstellt und zunächst die notwendigen Module Tkinter und CustomTkinter importiert. CustomTkinter bietet die Möglichkeit den Erscheinungsmodus der App zu bestimmen. Hier kann zwischen „Light“, „Dark“ und „System“ gewählt werden. Im „Light“-Modus wird das Interface hell dargestellt, während im „Dark“-Modus die App dunkel erscheint. Der „System“-Modus passt die Erscheinung an die Grundeinstellungen des Computers an, auf dem die App läuft. Ist die Grundeinstellung des Systems hell, erscheint die App im „Light“-Modus, ist die Grundeinstellung dunkel, wird in den „Dark“-Modus gewechselt, wie in Abbildung 4 zu sehen ist.



**Abbildung 4: Modernes HNU\_Chatbot-Fenster ("Light"-Modus / "Dark"-Modus)**

Für die Programmierung des GUI wurde ein objektorientierter Programmieransatz gewählt. Damit innerhalb einer Klasse Funktionen auf Variablen aus der Klasse zugreifen können, erhalten alle definierten Funktionen in der Klasse als Eingabe das „self“-Argument. Als Klasse, die die Chatbot-Applikation beschreiben soll, wird die Klasse „ChatApplication“ erstellt, in deren `__init__()`-Funktion ein Tkinter-Fenster unter dem Variablennamen „window“ geöffnet wird. Das Fenster bekommt im Code aus Abbildung 5 eine Breite von 550 Pixeln und eine Höhe von 650 Pixeln zugewiesen, sowie ein neues Icon, das in Anhang 2 zu sehen ist. Das neue Icon erscheint nun in der oberen linken Ecke des Hauptfensters. Da die Zuweisung des Icons mit CustomTkinter nicht funktioniert hat, wurde als Basis der Applikation weiterhin ein Tkinter-Fenster genutzt. Zuletzt wird für das moderne Interface mit der `_setup_main_window()`-Funktion der eigentliche Aufbau des GUI definiert.

```

23     self.window.geometry(f'{550}x{650}')
24
25     # Icon
26     icon = tk.PhotoImage(file='./images/Icon_50x50.png')
27     self.window.tk.call('wm', 'iconphoto', self.window._w, icon)

```

**Abbildung 5: HNU\_Chatbot.py, Zeile 23-27, Fenstergröße und Icon**

Für die Aufstellung der grundlegenden Benutzeroberfläche wird zunächst in der Funktion `_setup_main_window()` der Titel des Hauptfensters neben dem Icon zu „HNU-Bot“ geändert. Um alle Widgets des Interfaces in das Fenster einfügen und positionieren zu können, wird aus dem CustomTkinter-Modul ein CTKFrame erstellt, der dem Hauptfenster untergeordnet wird

und dieses komplett überdeckt. Die Widgets werden anschließend auf diesem CTKFrame verteilt und nicht auf dem Tkinter-Fenster selbst. Das ist nötig, da das Tkinter-Fenster nicht den Erscheinungsmodus verändert. Wenn die Grundeinstellung des Computer beispielsweise im Dunkelmodus ist, nehmen nur die Widgets aus dem CustomTkinter Modul den „Dark“-Modus an, nicht aber das Tkinter-Fenster. Die Überschrift des Chatbot-Prototypen wird durch ein CTKLabel-Widget, mit dem Text „Willkommen zum HNU ChatBot“ ersetzt. Jedes Widget braucht dabei ein Master, dem es untergeordnet wird. Das CTKLabel wird hier dem CTKFrame „frame“ untergeordnet. Die Schrift wird dabei, wie in Abbildung 6 zu sehen, mit der Schriftart Arial, Schriftgröße 21 und fett-gedruckt spezifiziert. Mit der place()-Funktion in Zeile 63 wird das Widget schließlich auf den CTKFrame projiziert. Dabei wird für das CTKLabel, relativ zum Master, eine relative Breite (relwidth) von 1 (100% der Breite des Masters) und eine relative Höhe (relheight) von 0.05 (5% der Höhe des Masters) festgelegt. Positioniert wird das Widget mit dem rely-Argument bei 1% der relativen y-Achse des Masters.

```
58     self.label = ctk.CTkLabel(  
59         self.frame,  
60         text = 'Willkommen zum HNU ChatBot',  
61         font = ctk.CTkFont('Arial', 21, 'bold')  
62     )  
63     self.label.place(relwidth=1, relheight=0.05, rely=0.01)
```

Abbildung 6: HNU\_Chatbot.py, Zeile 58-63, CTKLabel

Unter der Überschrift wird mit den Code-Zeilen 66-67 eine Leinwand (CTkCanvas) erstellt und platziert, auf der später die Nachrichten von Nutzer und Chatbot abgebildet werden sollen. Die CTkCanvas bekommt dabei mit dem Argument bg (background) die konstante Variable CANVAS\_COLOR, hinter der die Farbe Weiß steht, zugewiesen. Alle konstanten Variablen, die in dem Skript verwendet werden, werden in den Zeilen 12-16 bestimmt. Damit auch bei mehreren Nachrichten, die möglicherweise nicht alle gleichzeitig auf die Leinwand passen, der Nutzer zwischen ihnen hin- und herscrollen kann, wird eine CTKScrollbar für die Leinwand erstellt (Abbildung 7).

```
69     # Scrollbar  
70     self.scrollbar = ctk.CTkScrollbar(  
71         self.canvas,  
72         command=self.canvas.yview  
73     )  
74     cw = self.get_canvas_width()  
75     self.scrollbar.place(relheight=1, relx=(cw-17)/cw)  
76     self.canvas.configure(yscrollcommand=self.scrollbar.set)  
77     self.canvas.bind('<Configure>', self.update_scrollbar_position)  
78     self.canvas.bind_all('<MouseWheel>', self._on_mousewheel)
```

Abbildung 7: HNU\_Chatbot.py, Zeile 69-78, Scrollbar für Leinwand

Standardmäßig ist die CTKScrollbar vertikal eingestellt und wird durch das command-Argument in Zeile 72 an die vertikale Achse der Leinwand („canvas“) gebunden. Alle Widgets der

App sind auf eine relative Größe ausgelegt. Das bedeutet, wenn sich die Größe des Hauptfensters verändert, verändert sich auch dementsprechend die Größe der Widgets. Aus diesem Grund darf die Scrollbar nicht an eine feste Stelle auf der Benutzeroberfläche platziert werden, wenn sie immer den gleichen Abstand zum rechten Leinwandrand haben soll, sondern muss relativ zur CTKCanvas-Breite positioniert werden. Um die relative x-Achsenposition auf der CTKCanvas zu erhalten, wird daher in Zeile 74 die aktuelle Breite der Leinwand bestimmt. Die Funktion `get_canvas_width()` wird außerhalb der `_setup_main_window()`-Funktion in den Zeilen 144 und 145 definiert und gibt die aktuelle Breite der Leinwand aus. Mit diesem Wert, der in der Variable „cw“ gespeichert wird, kann in der `place()`-Funktion der Wert für `relx` errechnet werden. Hierzu werden 17 Pixel von der Leinwandbreite abgezogen und das Ergebnis durch die Leinwandbreite geteilt, um die relative Position für die Scrollbar auf der Leinwand zu erhalten. Die Code-Zeile 76 verbindet die Scrollbar zur CTKCanvas. Anschließend wird die Funktion `update_scrollbar_position()` an das Event, dass die Größe der Leinwand verändert wird, gebunden, um die Position der Scrollbar in diesem Fall anzupassen. Diese Funktion ist in den Zeilen 147-149 definiert und enthält den selben Code und die Funktion, wie die Zeilen 74 und 75. Zuletzt wird mit `bind_all()` bestimmt, dass wenn der Mauszeiger irgendwo auf dem Fenster des Chatbots ist und das Mousrad gedreht wird, die Funktion `_on_mousewheel()` ausgeführt wird. Diese wird in den Zeilen 155 und 156 definiert und sorgt dafür, dass die Leinwand hoch- bzw. runtergescrollt wird.

Der Nutzer soll seine Frage in ein Eingabefeld schreiben und an den Chatbot senden können. Hierzu wird ein CTKEntry-Widget erstellt, das als Platzhaltertext „Bitte stellen Sie Ihre Frage“ erhält und am unteren Rand des Fensters platziert wird. Mit der Funktion `focus()` wird der Fokus in dem Fenster auf dieses Eingabefeld gelegt. Das bedeutet, dass der Nutzer das Eingabefeld nicht mehr anklicken muss, sondern nach dem Öffnen der App direkt seine Frage eintippen kann. Zudem wird an das Eingabefeld das Ereignis, dass die Enter-Taste gedrückt wird, gebunden und daraufhin die Funktion `_on_enter_pressed()` ausgeführt. Die Funktion holt sich die geschriebene Nachricht aus dem CTKEntry-Feld und überprüft, ob die Nachricht einen Text enthält oder nur ein leerer String aus dem Eingabefeld gezogen wurde. Hat der Nutzer eine Nachricht eingetippt, wird diese als Argument in Zeile 133 in Abbildung 8 an die `send_message()`-Funktion gegeben. In dieser Funktion werden die Sprechblasen mit der Nachricht des Nutzers und der Antwort des Chatbots auf die Leinwand gebracht. Genaueres über die Funktionsweise hinter diesem Abschnitt folgt im Teilkapitel 3.1.2. Grob beschrieben gibt die `send_message()`-Funktion jedoch die Nachricht des Nutzers in Zeile 137 mit der `get_response()`-Funktion an das KNN. Die Funktion `get_response()` wird in der Datei „chat.py“ definiert. Zurück kommt die Antwort des Chatbots auf die Nachricht des Nutzers, die dann ebenfalls dem Nutzer präsentiert werden kann. Anschließend löscht die Funktion in Zeile 139 die Eingabe des Nutzers aus dem Eingabefeld.

```

130     def _on_enter_pressed(self, event):
131         self.msg = self.msg_entry.get()
132         if self.msg != '':
133             self.send_message(self.msg)
134
135     def send_message(self, msg):
136         user = UserBubble(self.canvas, message=msg)
137         self.response = get_response(msg)
138         bot = BotBubble(self.canvas, message=self.response)
139         self.msg_entry.delete(0, ctk.END)
140         self.user_bubbles.append(user)
141         self.bot_bubbles.append(bot)
142         self.update_scrollregion()

```

**Abbildung 8: HNU\_Chatbot.py, Zeile 130-142, \_on\_enter\_pressed()-Funktion und send\_message()-Funktion**

Im Prototyp wurden die Antworten ohne Umlaute und deren Stelle stattdessen durch deren bekannten Schreibweise „ae“, „oe“ und „ue“ ersetzt, da Python Probleme mit dem Einlesen von Umlauten und einigen Sonderzeichen hat. Mit der Datei „intents\_umlaute.json“ wurde eine Kopie der vom Prototypen genutzten „intents.json“-Datei erstellt, in der die Schreibweise der Wörter wieder mit Umlauten abgespeichert wurde. Zudem wurde in der Datei „chat.py“ die Funktion `replace_umlauts()` definiert (siehe Anhang 3, Zeile 70-84), die einen Text als Eingabe erhält und die falsch eingelesenen Zeichen durch die richtigen Zeichen ersetzt. Beispielsweise wird so „Ä“ durch „ä“ und „Ã“ durch „Ä“ ersetzt. Zudem ersetzt die Funktion das Zeichen „|“, das in den Antworten als Trennlinie von Abschnitten genutzt wurde, durch zwei Zeilenumbrüche, um die Übersichtlichkeit und Lesbarkeit der Nachricht zu verbessern. Die Funktion `replace_umlauts()` wird in die Funktion `get_response()` eingebaut, sodass die `send_message()`-Funktion in Zeile 137 die bereits berichtigte Version der Antwort erhält.

Neben dem Eingabefeld wird der Button zum Senden der Nachricht erstellt, der mit „Senden“ beschriftet ist und, wenn er geklickt wird, wiederum die Funktion `_on_enter_pressed()` ausführt.

Abschließend wird auf der linken Seite der App noch eine Menüleiste als `CTkFrame` erstellt. Auf ihr sind drei Icons untereinander platziert, die jeweils eine eigene Funktion haben, wenn sie angeklickt werden.

Das Icon „⚠“ (Antwort melden) soll die Möglichkeit bieten, Antworten des Chatbots abzuspeichern, die nicht zur gestellten Frage passen, um Schwächen im KNN durch Nachtrainieren ausbessern zu können. Beim Anklicken des Icons wird die Funktion `antw_button_event()` ausgeführt. Auf die Funktionsweise dieser Funktion wird im Kapitel 3.2.2 ausführlich eingegangen.

Als zweites symbolisiert das Icon „📄“ die Funktion den Chatverlauf in einem Word-Dokument abzuspeichern. Hierzu wird beim Anklicken des Icons die Funktion `save_button_event()` ausgeführt. In der Funktion wird zunächst das aktuelle Datum erfasst. Anschließend wird in einer `for`-Schleife durch alle erstellten Chatnachrichten iteriert und mit Hilfe der `get_text()`-Funktion

der Text aus den Nachrichten in der Variable „chat“ gespeichert. Dabei wird vermerkt, ob die Nachricht vom Nutzer selbst oder vom Chatbot geschrieben wurde. Folgend soll der Nutzer einen Speicherort für das Text-Dokument wählen. Nun kann das eigentliche Dokument geschrieben werden. Zunächst wird die Überschrift „Chatverlauf“ mit dem aktuellen Datum eingefügt und anschließend um den Chatverlauf ergänzt und abgespeichert, wie beispielhaft in Anhang 4 abgebildet.

Mit dem dritten Icon „💡“ kann der Nutzer manuell zwischen Hell- und Dunkelmodus der App wechseln. Hierbei wird beim Anklicken des Icons die Funktion `mode_button_event()` ausgeführt, die überprüft, in welchem Modus die App momentan angezeigt wird und dann jeweils den Erscheinungsmodus zu dem anderen Modus umstellt.



**Abbildung 9: Tooltip für Icon**

Wird der Mauszeiger über die Icons bewegt, erscheint ein Tipp, der beschreibt, was für eine Funktion hinter dem Icon steht, wenn es angeklickt wird (siehe Abbildung 9). Für diese Funktion wurde die neue Klasse „CreateToolTip“ erstellt. Der Code der Klasse ist dabei von Stevoisiaks Antwort auf der Internetseite [stackoverflow.com](https://stackoverflow.com) zu der Frage, wie Tooltips in Tkinter angezeigt werden können (vgl. Stevoisiak, 2018). Um einen neuen Tooltip zu erstellen, muss dem Objekt der Klasse ein Widget gegeben werden – in diesem Fall die CTKLabels in der Menüleiste, die die Icons tragen – und ein Text, der im Tooltip angezeigt werden soll. An das Widget werden die Events, dass der Mauszeiger auf das Widget geschoben wird, bzw. dieses wieder verlässt, gebunden und daraufhin die entsprechenden Funktionen `enter()` oder `leave()` ausgeführt. In der `enter()`-Funktion wird der Tooltip nach 750 Millisekunden, die der Mauszeiger auf dem Icon verweilen muss, angezeigt. Um den Tooltip anzuzeigen, werden zunächst die Koordinaten des Widgets erfasst und erhöht, sodass sie auf eine Position neben dem Widget verweisen. An dieser Stelle wird ein neues Tkinter-Toplevel-Widget erstellt, in das ein Label mit dem geforderten Text für den Tooltip eingebettet wird. Zuletzt schließt die `leave()`-Funktion den Tooltip wieder.

### **3.1.2 ChatBubbles**

Um einen Messenger-App-Stil zu erzielen, sollen die Nachrichten von Nutzer und Chatbot nicht mehr in einem Textfeld untereinander abgebildet werden, sondern mit unterschiedlich

farbigen Sprechblasen ein ansprechenderes Design erhalten. Um diese Sprechblasen zu kreieren, wird für die Sprechblasen des HNU\_Bots und des Nutzers jeweils eine eigene Klasse mit den Namen „BotBubble“ und „UserBubble“ erstellt. Als Grundlage für diese Klassen dient ein Code-Ausschnitt, den Henry Yik (vgl. 2019) auf [stackoverflow.com](https://stackoverflow.com) zu einer Frage, wie sich Sprechblasen für einen Chatbot mit Tkinter erstellen lassen. Der von Yik präsentierte Code erstellt ein Tkinter-Label in das der Text der Nachricht und ein Zeitstempel, wann die Nachricht gesendet wurde, eingetragen wird. Dazu wird ein Polygon mit drei Ecken an das Label ge-

```

12 CANVAS_COLOR = '#FFFFFF'
13 USER_BUBBLE = '#43CD80'
14 BOT_BUBBLE = '#7A8888'
15 FONT_COLOR = '#FFFFFF'
16 BUBBLE_FONT = ('Arial',12)

```



Abbildung 10: HNU\_Chatbot.py, Zeile 12-16, konstante Variablen mit Farbdarstellung

zeichnet, wie es von der Darstellung von Sprechblasen bekannt ist. Ein Problem bei diesem simplen Code ist, dass lange Texte, nicht in mehrere Zeilen aufgeteilt werden, sondern sich das Label über den Fensterrand hinaus erstreckt und so die Nachricht nicht mehr vollständig lesbar ist. Um den Code auf das Design des HNU\_Bots anzupassen, wurden zunächst die konstanten Variablen in Abbildung 10 erstellt. Für die Leinwandfarbe wurde, wie zuvor beschrieben, mit der Variable „CANVAS\_COLOR“, die Farbe Weiß zugeschrieben. Die Sprechblasen für Nutzer und Bot sollen in unterschiedlichen Farben dargestellt werden. Daher erhält in Abbildung 10 die Variable „USER\_BUBBLE“ für die Nutzersprechblase einen grünen Farbton (Zeile 13) und die Variable „BOT\_BUBBLE“ für die Sprechblase des Chatbots eine grauen Farbe (Zeile 14). Die Schriftfarbe wird mit „FONT\_COLOR“ ebenfalls auf Weiß festgelegt und die Schriftart in „BUBBLE\_FONT“ auf Arial mit der Schriftgröße 12.

Mit den konstanten Variablen können nun die Sprechblasen für Nutzer und Bot unterschiedlich dargestellt werden. Hierzu werden zunächst im Code von Yik alle Widgets durch CustomTkinter-Widgets ausgewechselt, um ein moderneres Design mit abgerundeten Ecken, passend zum Design der restlichen Benutzeroberfläche, zu erhalten. Die erstellten Klassen „BotBubble“ und „UserBubble“ unterscheiden sich in einigen Punkten, was es sinnvoll machte, diese Sprechblasentypen in unterschiedlichen Klassen zu programmieren. Für beide Klassen muss jedoch an die `__init__()`-Funktion ein Master-Widget, dem die Sprechblasen untergeordnet werden sollen und eine Nachricht, die in der Sprechblase stehen soll, übergeben werden. Der Master wird zunächst mit der Zeile 246 bzw. 286 in Abbildung 11 zu einer Variable umgewandelt, auf die durch das `self`-Argument in der gesamten Klasse zugegriffen werden kann. So wird in Zeile 247 bzw. 287 eine Box um alle Widgets, die im Master zu finden sind, gedacht und die Koordinaten auf der Leinwand durch zwei Punkte für die linke obere Ecke  $(x_1, y_1)$  und die rechte untere Ecke  $(x_2, y_2)$  dieser Box in der Variable „bbox“ abgespeichert. Diese Koordinaten werden in der darauffolgenden Zeile genutzt, um den Wert für die y-Achse zu errechnen,

```

244 class BotBubble:
245     def __init__(self, master, message=''):
246         self.master = master
247         bbox = self.master.bbox(tk.ALL)
248         y = (bbox[3] if bbox else 0) + 10
249         self.frame = tk.Frame(master, bg=BOT_BUBBLE)
250         self.i = self.master.create_window(15, y, window=self.frame, anchor='nw')
251         self.bottLabel = ctk.CTkLabel(
252             self.frame,
253             text=message,
254             justify=ctk.LEFT,
255             font=BUBBLE_FONT,
256             text_color=FONT_COLOR,
257             fg_color=(BOT_BUBBLE),
258             bg_color=(CANVAS_COLOR),
259             padx=1,
260             pady=7,
261             corner_radius=8
262         )
263         # Bubble width
264         self.bottLabel.configure(wraplength=(self.get_master_width()-75))
265         self.bottLabel.grid(row=1, column=0)
266
267
268
269
270
271
272
273
274 class UserBubble:
275     def __init__(self, master, message=''):
276         self.master = master
277         bbox = self.master.bbox(tk.ALL)
278         y = (bbox[3] if bbox else 0) + 10
279         self.frame = tk.Frame(master, bg=USER_BUBBLE)
280         w = self.master.winfo_width()
281         self.i = self.master.create_window(w-25, y, window=self.frame, anchor='ne', tag='userBubble')
282         self.userLabel = ctk.CTkLabel(
283             self.frame,
284             text=message,
285             justify=ctk.LEFT,
286             font=BUBBLE_FONT,
287             text_color=FONT_COLOR,
288             fg_color=(USER_BUBBLE),
289             bg_color=(CANVAS_COLOR),
290             padx=1,
291             pady=7,
292             corner_radius=8
293         )
294         # Bubble width
295         self.userLabel.configure(wraplength=(self.get_master_width()-75))
296
297
298
299
300
301
302
303
304
305

```

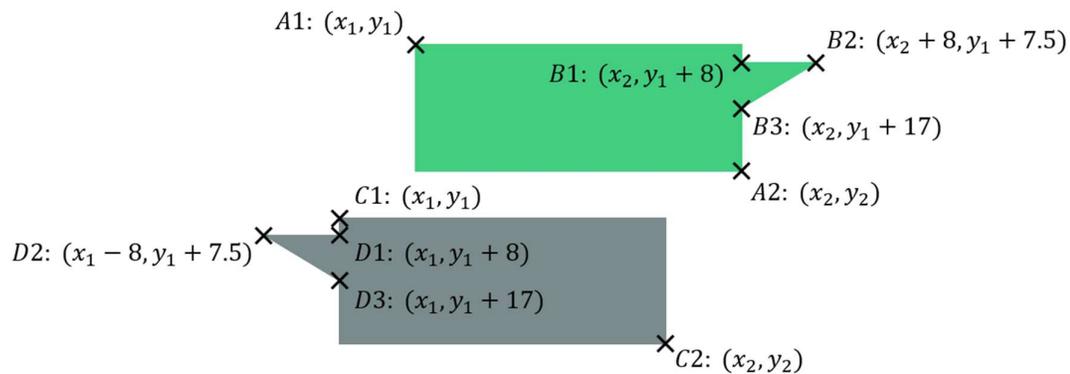
**Abbildung 11: HNU\_Chatbot.py, Vergleich der \_\_init\_\_() -Funktion aus BotBubble-Klasse (links) und UserBubble-Klasse (rechts)**

auf dem die nächste Sprechblase auf dem Master positioniert werden soll. In dem if-Statement in Zeile 248 bzw. 288 wird geprüft, ob es einen Wert für  $y_2$  gibt, welcher den Indexwert 3 in der Variable „bbox“ besitzt. Falls es diesen Wert nicht gibt, bedeutet das, dass es noch kein Widget im Master gibt. In diesem Fall wird 0 als  $y_2$ -Wert verwendet. Anschließend wird zum  $y_2$ -Wert, der 0 oder den y-Wert für den unteren Rand des letzten Widgets auf dem Master annimmt, 10 Pixel addiert, damit in der Darstellung der nächsten Sprechblase ein Abstand zwischen der letzten und der neuen Sprechblase ist. In den folgenden Zeilen wird ein Frame aus dem Tkinter-Modul kreiert. Mit create\_window() kann später das Dreieck für die Sprechblase mit dem Textkörper verbunden werden. Hier wird auch der Abstand zum Rand des Masters, auf dem die Sprechblasen gezeichnet werden sollen, definiert. Da bei Tkinter und CustomTkinter ein Widget immer bei den Koordinaten  $(x_1, y_1)$  mit  $(0, 0)$  anfängt und bei  $(x_2, y_2)$  mit (Breite des Widgets, Höhe des Widgets) aufhört, wird bei der „BotBubble“-Klasse für den x-Wert der neuen Sprechblase 15 als Wert festgelegt. Die Ausrichtung des Widgets wird mit dem anchor-Argument auf „nw“ (north-west) an den linken und unteren Rand ausgerichtet. Der y-Wert für das neue Widget wird von der oben beschriebenen Variable „y“ übernommen. Bei der „UserBubble“-Klasse ist hier der erste Unterschied. Da die Nutzersprechblase rechtsbündig auf der Leinwand angeordnet werden soll, muss dies auch beim x-Wert als Abstand zum rechten Rand beachtet werden. Um diesen zu berechnen, wird zunächst in Zeile 290 die Breite des Master-Widgets in der Variable „w“ gespeichert. Daraufhin kann für den x-Wert in der create\_window()-Funktion die Breite abzüglich einem Abstand von 25 Pixeln, um Platz für die Scrollbar am rechten Bildrand zu lassen, eingesetzt werden. Ebenfalls anders in dieser Funktion ist, dass das anchor-Argument hier mit „ne“ (north-east) bestimmt ist, damit die Sprechblase vom rechten Rand aus gezeichnet wird. Zudem wird der Sprechblase das Tag „userBubble“ vergeben. Dieses Tag wird im weiteren Verlauf des Kapitels 3.1.3 von Bedeutung sein, um die Nutzersprechblasen auf der Leinwand zu identifizieren. Schließlich wird in den Zeilen 251-262 bzw. 292-303 der Textkörper der Sprechblase erstellt. Hierfür wird ein CTKLabel-Widget genutzt, dem als Text die Nachricht, die bei der Erstellung der Klasse mitgegeben wurde, eingefügt wird. Außerdem wird der Text in Zeile 254 bzw. 295 linksbündig ausgerichtet und in den darauffolgenden Zeilen werden die zu Beginn des Kapitels definierten

konstanten Variablen genutzt, um das Aussehen der Schrift, sowie der Sprechblase zu bestimmen. Beim Aussehen der Sprechblase muss hierbei eine Farbe für den Vordergrund des CTKLabels (`fg_color`) und damit die eigentliche Sprechblasenfarbe und eine Farbe für den Hintergrund (`bg_color`) angegeben werden. Die Hintergrundfarbe füllt das rechteckige Feld des Widgets komplett aus, während die Vordergrundfarbe nur den Bereich ausfüllt, der mit den abgerundeten Ecken der Sprechblase definiert ist. Daher muss die Hintergrundfarbe die Farbe der Leinwand sein, damit dies nicht sichtbar wird. Der Rest der `__init__()`-Funktion ist für beide Klassen wieder identisch. Als nächstes wird die Breite der Sprechblase bestimmt und damit auch, an welcher Stelle das CTKLabel Zeilenumbrüche in die Nachricht einbauen soll. Hierbei wird der Vorteil des CTKLabels gegenüber dem im Prototypen verwendeten Text-Widget deutlich. Es werden Texte nicht auf Buchstabenebene geteilt und in die nächste Zeile geschoben, sondern ein Zeilenumbruch wird nur bei einem Leerzeichen eingefügt. Das erleichtert die Lesbarkeit der Nachrichten. Um die Breite der Sprechblase zu bestimmen, wird die `get_master_width()`-Funktion genutzt. Diese gibt die Breite des Masters, also im Fall der Sprechblasen die der Leinwand, aus und wird in den Zeilen 276-277 bzw. 317-318 definiert. Von dem resultierenden Wert werden 75 Pixel subtrahiert, um somit die geforderte Breite der Sprechblase zu erhalten. Zuletzt wird noch das Polygon in Form eines Dreiecks an das CTKLabel gezeichnet. Die Spezifikationen der Koordinaten hierfür stammen aus der Funktion `draw_triangle()`.

Die Funktion `draw_triangle()` wird direkt nach der `__init__()`-Funktion definiert. Sie erhält als Eingabe ein Widget, in diesem Fall das CTKLabel mit der Nachricht und erhält durch die `bbox()`-Funktion, den oberen linken und den unteren rechten Punkt des Widgets. Mit diesen Punkten können nun die Koordinaten für das Dreieck der Sprechblase errechnet werden. Am Beispiel der Nutzersprechblase, die grüne Sprechblase in Abbildung 12, wird folgend die Errechnung der Koordinaten beschrieben. Um das Polygon in der `__init__()`-Funktion zu erstellen, braucht die `create_polygon()`-Funktion die x- und y-Werte für jede Ecke des Vielecks. Als Eingabe für diese Funktion wird daher die Ausgabe der `draw_triangle()`-Funktion genutzt. Diese gibt für die Nutzersprechblase die drei Punkte B1, B2 und B3 zurück, mit denen das Dreieck definiert wird. In der Abbildung ist das Ergebnis der `bbox()`-Funktion durch die Punkte A1 und A2 dargestellt. Für den Punkt B1 entspricht der x-Wert dem x-Wert aus Punkt A2 des CTKLabels. Der y-Wert wird aus dem y-Wert von Punkt A1 errechnet, indem dieser Wert um 8 Pixel erhöht wird. Das verschiebt den Punkt B1 auf der y-Achse nach unten, wie in Abbildung 12 zu erkennen ist. Für Punkt B2 wird der y-Wert nur um 7,5 Pixel verschoben, um eine leicht schräge Kante im Dreieck zu erzeugen. Bei der Errechnung des x-Wertes wird wieder der x-Wert aus A2 genutzt und nun um 8 Pixel erhöht. Der Punkt B2 bildet am Ende die Spitze des Dreiecks, die von der Sprechblase weg zeigt. Der letzte Punkt B3 wird gleichermaßen definiert, wie der Punkt B1, nur dass der y-Wert noch weiter nach unten verschoben wird, indem er um 17 Pixel erhöht wird, anstatt um 8 Pixel. Auf gleiche Weise werden auch die Koordinaten für das Dreieck in

der Sprechblase des Chatbots errechnet, allerdings mit anderen Werten, damit das Dreieck auf der linken Seite der Sprechblase erscheint, wie in Abbildung 12 in der grauen Sprechblase zu erkennen ist.

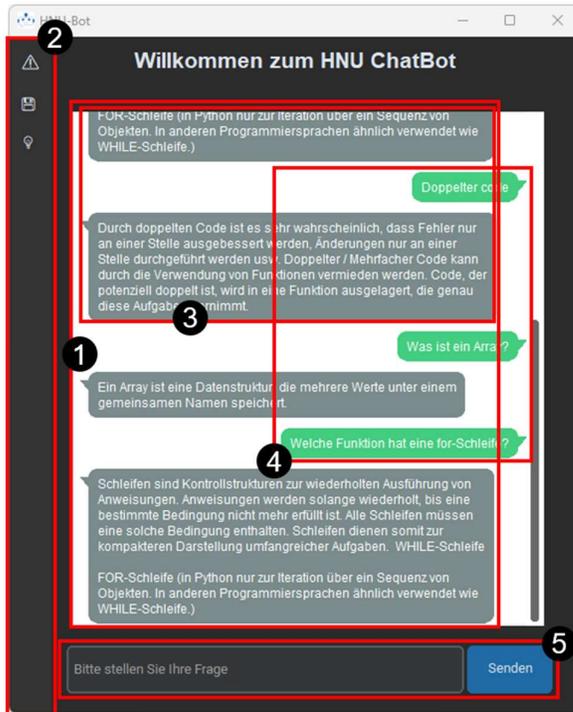


**Abbildung 12: Skizze der Berechnung der Koordinaten für die Polygone der Sprechblasen**

Um nach dem Erstellen der Sprechblasen weiterhin auf den Text der Nachrichten zugreifen zu können, wird in beiden Klassen (Zeile 273-274 bzw. 314-315) die Funktion `get_text()` erstellt, die den Text der Nachrichten zurück gibt.

Als letzte Funktion in den Klassen der Sprechblasen wird die Funktion `config_wraplength()` definiert. Die Funktion konfiguriert die Breite der Sprechblasen neu, indem sie, wie schon vorangegangen beschrieben, mit der Funktion `get_master_width()` die Breite des Masters erhält und von dieser 75 Pixel abzieht. Anschließend wird der Master aktualisiert, um die Veränderung sichtbar zu machen. Diese Funktion macht es möglich die Breite der Sprechblasen anzupassen, wenn das Chatbot-Fenster vergrößert oder verkleinert wird. Daher wird dies im Kapitel 3.1.3 genauer erläutert.

Wie in Kapitel 3.1.1 beschrieben, werden die Sprechblasen in der Klasse „ChatApplication“ innerhalb der `send_message()`-Funktion kreiert. Damit der Chatbot später auf die einzelnen Nachrichten zugreifen kann, werden zu Beginn der Klasse in der `__init__()`-Funktion zwei leere Listen mit den Variablennamen „user\_bubbles“ und „bot\_bubbles“ erstellt. In der Funktion `send_message()` wird zunächst in Zeile 136 ein neues Objekt der „UserBubble“-Klasse mit der Leinwand als Master und der Eingabe des Nutzers als Nachricht erstellt und damit auf die Leinwand projiziert (vgl. Abbildung 8). Als Antwort des Chatbots wird in Zeile 138 ein Objekt der „BotBubble“-Klasse erstellt, das ebenfalls die Leinwand als Master erhält und zusätzlich als Nachricht die Antwort des KNN, die durch die `get_response()`-Funktion gewählt wurde. Beide Sprechblasen-Objekte werden anschließend in den Zeilen 140-142 ans Ende der entsprechenden Liste angefügt und der gezeigte Ausschnitt der Leinwand aktualisiert, damit die Nachricht, die zuletzt gesendet wurde, vollständig zu sehen ist.



**Abbildung 13: Einsatz der Gestaltprinzipien im GUI**

Beim Design der Sprechblasen und der gesamten Benutzeroberfläche wurden die in Kapitel 3.1 beschriebenen Gestaltprinzipien beachtet, um das GUI für den Nutzer intuitiver und ansprechender zu gestalten. So haben die Sprechblasen des Chatbots, zu sehen in Bereich 1 der Abbildung 13, durch die Richtung des Dreiecks eine andere Form und zusätzlich eine andere Farbe, als die Sprechblasen des Nutzers in Bereich 4. Durch das gleiche Auftreten der Bot- bzw. Nutzersprechblasen werden diese vom Nutzer intuitiv der jeweiligen Gruppe zugeordnet, was die Benutzeroberfläche deutlich übersichtlicher erscheinen lässt. Dies ist im Prinzip der Ähnlichkeit beschrieben. Eine gleiche Einteilung der Sprechblasen in die Gruppen Bot- und Nutzersprechblasen unterstützt das Prinzip der gemeinsamen Region. Das Prinzip spiegelt sich im Interface darin wieder, dass die Sprechblasen jeweils links- oder rechtsbündig auf der Leinwand orientiert sind. Auch, dass die Nachrichten nicht die komplette Leinwandbreite einnehmen, sondern nur die Breite abzüglich 75 Pixel, unterstützt den Effekt dieses Prinzips. Noch besser lässt sich das Prinzip der gemeinsamen Region jedoch an der Menüleiste in Bereich 2 verdeutlichen. Dieser Bereich grenzt die Icons auf der linken Seite durch eine farbliche Abhebung des Menüs vom Rest der App ab und deutet so an, dass es sich bei den Icons um zusätzliche Funktionen handelt, die für die Grundfunktion des Chatbots irrelevant sind. Im dritten Bereich der Abbildung 13 kann das Prinzip der Kontinuität beobachtet werden. Obwohl ein Teil der Nachricht über den oberen Rand der Leinwand hinaus geht und somit nicht mehr zu sehen ist, erkennt der Nutzer direkt, dass die Nachricht nach oben hin weiter geht. Die gewohnte Form der Sprechblasen mit abgerundeten Ecken trifft auf diese Nachricht nicht zu, daher wird davon ausgegangen, dass sie über den Rand hinaus gehen muss. Zuletzt kann in

Bereich 5 das letzte beschriebene Prinzip identifiziert werden. Der Senden-Button wurde bei der Platzierung direkt neben dem Eingabefeld positioniert. Nicht nur ist dies die Position, an der der Senden-Button aus Erfahrung mit anderen Messenger-Apps erwartet wird, sondern das Prinzip der Nähe in der Gestaltpsychologie besagt, dass Objekte, die nah zueinander stehen als zusammengehörig betrachtet werden und so die Funktion des Senden-Buttons auf das Eingabefeld bezogen wird.

### 3.1.3 Aktualisierung der Fenstergröße

Mit den Sprechblasen implementiert in das Design des Chatbots sieht der HNU\_Bot vollständig aus. Allerdings sind die Sprechblasen fest an ihre Position und in ihrer Größe auf der Leinwand gebunden. Wird die Größe des Chatbot-Fensters verändert, ändern die Sprechblasen daher weder ihre Größe, noch ihre Position auf der Leinwand. Dies ist besonders auffällig bei den Nachrichten des Nutzers. Wenn die Fensterbreite verändert wird, ändert sich ebenfalls die Breite der Leinwand. Der linke Rand hat weiterhin den x-Wert 0, somit macht dies keinen Unterschied für die Sprechblasen des Chatbots, da die Position dieser Sprechblasen immer einen x-Wert von 15 Pixel hat (siehe Kapitel 3.1.2). Die Position der Nutzersprechblasen errechnet sich jedoch aus der Breite der Leinwand abzüglich 25 Pixel. Somit muss, wenn die Größe des Fensters verändert wird, dieser x-Wert neu berechnet werden. Zudem soll die Breite der Sprechblasen der Breite der Leinwand angepasst werden. So hat der Nutzer die Möglichkeit für besonders lange Antworttexte das Fenster breiter zu ziehen, damit er die gesamte Nachricht sehen kann.

Für diese Aufgaben sind die Funktionen `update_bot_bubbles()` und `update_user_bubbles()` zuständig. Zunächst wird in beiden Funktionen die Breite der Sprechblasen an die Breite der Leinwand angepasst. Hierzu wird jeweils auf jede Sprechblase, die sich in der Liste „user\_bubbles“ bzw. „bot\_bubbles“ befindet, die Funktion `config_wraplength()` angewandt. Wie beschrieben muss der x-Wert für die Position der Sprechblasen des Chatbots nicht aktualisiert werden. Daher ist die Aufgabe der Funktion `update_bot_bubbles()` hiermit erfüllt. Für die Sprechblasen des Nutzers wird der Abstand zum rechten Rand der Leinwand folgend aktualisiert. Dazu wurde schon zu Beginn der Funktion in Zeile 206 die Breite der Leinwand in der Variable „w“ gespeichert. In einer for-Schleife werden auf der Leinwand alle Objekte gesucht, die das Tag „userBubble“ besitzen. Wie im vorangegangenen Kapitel beschrieben, wurde der „UserBubble“-Klasse beim `create_window()`-Befehl dieses Tag mitgegeben. Für jedes Element, mit diesem Tag, wird pro Iteration der for-Schleife, zusehen in der Abbildung 14, die ID abgerufen. Die IDs werden von dem Modul CustomTkinter automatisch beim Erstellen und auf die Leinwand bringen generiert. Das erste Objekt erhält die ID 1, das zweite die 2 und so weiter. Da im Code zuerst das CTKLabel der Nachricht des Nutzers generiert wird, bekommt es die ID 1. Danach wird das Dreieck an das CTKLabel gezeichnet und erhält somit die ID 2. Weiter geht es mit der Sprechblase des Chatbots. Auch hier wird mit dem CTKLabel begonnen,

das die ID 3 erhält und abschließend wird das Dreieck gezeichnet mit der ID 4. Mit der ID des CTKLables der Nutzersprechblase kann erneut mit der bbox()-Funktion auf die Koordinaten des Objekts zugegriffen werden. Da die Breite der Sprechblase schon am Anfang dieser Funktion aktualisiert wurde, sind die Koordinaten, die in Zeile 214 in der Variable „bubble\_coords“ abgespeichert werden, die des veränderten CTKLables. In der folgenden Zeile wird die absolute Breite des CTKLables berechnet, indem von dem zweiten x-Wert der erste x-Wert subtrahiert wird. Mit Hilfe der move()-Funktion kann ein Objekt auf der Leinwand verschoben werden. Wichtig dabei ist, dass die Funktion von der aktuellen  $(x_1, y_1)$ -Position des Objektes ausgeht und nicht vom Punkt (0, 0) auf der Leinwand. So erhält die Funktion, wie in Zeile 216 zu sehen, als erstes Argument die ID des CTKLables das verschoben werden soll und danach die x- und y-Werte, um wie viele Pixel das Objekt verschoben werden soll. Die Sprechblase soll auf der aktuellen Höhe auf der y-Achse der Leinwand bleiben, daher ist der y-Wert 0. Um den x-Wert zu errechnen, um den das Objekt verschoben werden soll, wird von der Breite der Leinwand die x-Koordinate des linken CTKLabel-Rands subtrahiert. Das Resultat ist die neue x-Koordinate, an der die Sprechblase beginnen soll. Da aufgrund des anchor-Arguments „ne“ in der create\_window()-Funktion die Sprechblase ausgehend vom rechten Rand gezeichnet wird, muss hiervon die Breite des neuen Objekts abgezogen werden. Nun würde das CTKLabel sich so verschieben, dass es am rechten Leinwandrand anliegt. Daher muss noch der festgelegte Abstand von 25 Pixeln vom Rand abgezogen werden. Somit wird das CTKLabel an die richtige Position verschoben. Abschließend wird das Dreieck der Nachricht verschoben. Da die ID des Dreiecks eine Nummer höher ist, als das dazugehörige CTKLabel kann so in den Zeilen 219-220 die Koordinaten des Dreiecks herausgefunden werden. Das Dreieck wird wiederum mit der move()-Funktion verschoben. Dabei muss für den x-Wert, um den das Objekt verschoben werden soll, von der Breite der Leinwand nur die x-Koordinate des linken Objektsrands und der Abstand zum rechten Leinwandrand subtrahiert werden.

```
212 # Move position of UserBubble
213 for bubble_id in self.canvas.find_withtag('userBubble'):
214     bubble_coords = self.canvas.bbox(bubble_id)
215     diff = bubble_coords[2] - bubble_coords[0]
216     self.canvas.move(bubble_id, w - bubble_coords[0] - diff - 25, 0)
217
218 # Move position of triangle
219 triangle_id = bubble_id + 1
220 triangle_coords = self.canvas.bbox(triangle_id)
221 self.canvas.move(triangle_id, w - triangle_coords[0] - 27, 0)
```

Abbildung 14: HNU\_Chatbot.py, Zeile 212-221, Aktualisieren der Positionen der Nutzersprechblasen auf der x-Achse innerhalb der Funktion update\_user\_bubbles()

Da sich durch das Strecken und Stauchen der Sprechblasen die Höhe dieser verändert, müssen die Abstände zwischen den Sprechblasen aktualisiert werden. Hierfür ist die Funktion update\_bubble\_gap() zuständig. In einer for-Schleife werden dabei alle Objekte auf der Leinwand

```

228     def update_bubble_gap(self, event=None):
229         for bubble_id in self.canvas.find_all():
230             # Skip triangle
231             if (bubble_id % 2) == 0 or bubble_id == 1:
232                 continue
233             bubble_coords0 = self.canvas.bbox(bubble_id - 2)
234             bubble_coords = self.canvas.bbox(bubble_id)
235
236             if (diff := bubble_coords[1] - bubble_coords0[3]) != 10:
237                 # move position of BotBubble
238                 self.canvas.move(bubble_id, 0, -(diff-10))
239
240                 # Move position of triangle
241                 triangle_id = bubble_id + 1
242                 self.canvas.move(triangle_id, 0, -(diff-10))

```

**Abbildung 15:** HNU\_Chatbot.py, Zeile 228-242, Aktualisieren des Abstands zwischen den Sprechblasen

durchgegangen. Dabei wird zuerst in Zeile 231 der Abbildung 15 geprüft, ob es sich bei dem betrachteten Objekt um die erste Sprechblase mit der ID 1 handelt. Bei dieser Sprechblase bleibt der Abstand zum oberen Rand der Leinwand immer bei 10 Pixeln, da der obere Rand des CTKLabel die Position verändert. Zudem wird geprüft, ob es sich bei dem Objekt um ein Dreieck handelt. Dies ist der Fall, wenn eine ID durch den Modulo-Operator „%“ geteilt wird und dadurch eine Rest von 0 errechnet wird. Da alle Dreiecke eine gerade ID-Nummer haben und alle CTKLabel eine ungerade, lassen sich die beiden Objektarten somit leicht unterscheiden. Trifft eine der beiden Optionen zu, wird der restliche Code in der for-Schleife für diese Wiederholung nicht mehr durchlaufen, sondern die Funktion geht zum nächsten Objekt auf der Leinwand über. Handelt es sich um ein CTKLabel, werden in den Zeilen 233-234 die Koordinaten für die Sprechblase und die Sprechblase darüber eingeholt. Darauffolgend kann in Zeile 236 überprüft werden, ob der x-Wert des unteren Sprechblasenrands 10 Pixel niedriger auf der Leinwand ist, als der obere Rand der nächsten Sprechblase. Dabei wird im gleichen Zuge die Differenz dieser Rechnung durch den Walrus-Operator „:=“ in der Variable „diff“ gespeichert. Ist der Abstand zwischen den beiden Sprechblasen größer oder kleiner als 10 Pixel, wird das CTKLabel in Zeile 238 und das Dreieck in Zeile 242 mit der move()-Funktion auf der y-Achse verschoben. Um den richtigen y-Wert zu errechnen, wird die Formel  $-(diff - 10)$  genutzt, wobei „diff“ die errechnete Differenz aus dem if-Statement ist. Überlappen die beiden Sprechblasen beispielsweise um 5 Pixel nimmt die Differenz einen Wert von -5 an. In die Formel eingesetzt ergibt sich aus  $-(-5 - 10)$  ein Wert von +15. Die Sprechblase wird also um 15 Pixel nach unten verschoben und stellt so den normalen Abstand von wieder her. Umgekehrt verhält es sich, wenn die beiden Sprechblasen 15 statt 10 Pixel auseinander liegen. Aus  $-(15 - 10)$  ergibt sich ein y-Wert von -5, um den die Sprechblase nach oben auf der Leinwand verschoben wird. Auch in diesem Fall wird der reguläre Abstand von 10 Pixeln wiederhergestellt.

Die in diesem Unterkapitel beschriebenen Funktionen werden in der `__init__()`-Funktion in den Zeilen 125-128 an den Master „frame“ gebunden, dem CTKFrame, der als Grundlage über das Tkinter-Fenster gelegt wurde. Wenn das Event „<Configure>“ ausgelöst wird, dies ist immer dann der Fall, wenn die Größe des Fensters verändert wird, werden die Funktionen `update_user_bubbles()`, `update_bot_bubbles()`, `update_bubble_gap()` und `update_scrollregion()` ausgeführt. Das ändert, wie beschrieben die Größen und Positionen der Sprechblasen und sorgt dafür, dass immer automatisch die Leinwand ganz nach unten gescrollt wird. So ist die letzte Nachricht immer zu sehen und wird nicht vom Eingabefeld verdeckt.

### 3.2 Retraining-Fragen abspeichern

Bevor der Chatbot an die Studierenden ausgegeben wird, soll die Grundlage für Verbesserungsmöglichkeiten des KNN gelegt werden. Daher sollen Fragen, bei denen der HNU\_Bot die Intention des Nutzers nicht verstanden hat, oder die er falsch beantwortet hat, abgespeichert werden können. So lassen sich mit den gesammelten Fragen nach einer ersten Testphase bewerten, wie die KI hinter dem Chatbot weiter verbessert werden kann. Die Fragen sollen hierfür in zwei Excel-Dateien abgespeichert werden. Um zu gewährleisten, dass diese Dateien zum Nachtrainieren des KNN genutzt werden können, wird der Nutzer, wenn er die App schließt, durch das Pop-up-Fenster aus Abbildung 16 daran erinnert, dass er die Möglichkeit hat, sie per E-Mail an Frau Prof. Dr. Köppl zu schicken. Ebenfalls ist in dem Pop-up beschrieben, dass die Excel-Dateien in dem Ordner abgespeichert ist, wo die App des Chatbots hinterlegt ist. Damit das Programm den Pfad zu dem Ordner erhält, wird im Skript „chat.py“ in den Zeilen 37-41 der Pfad ermittelt und in der Variable „path“ gespeichert.

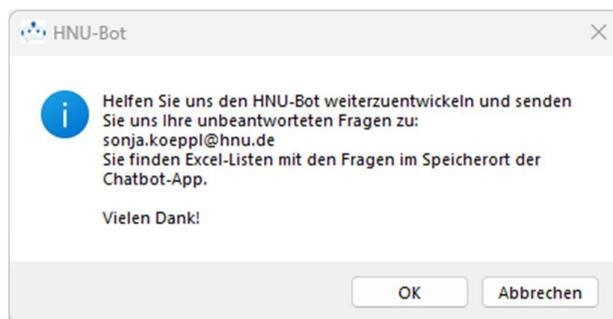


Abbildung 16: Pop-up beim Schließen der Chatbot-App

#### 3.2.1 Nicht verstandene Fragen

Zunächst wird in den Zeilen 57-60, die Excel-Datei „unknown\_questions.xlsx“ im Skript „chat.py“ geöffnet und als Tabelle in Form eines DataFrame-Objekt mit dem Namen „unknown\_questions“ abgespeichert. Zum Einlesen der Datei wird die Python-Bibliothek Pandas benutzt, die zu Beginn des Skripts unter dem Namen „pd“ importiert wurde. Existiert noch keine Datei mit dem Namen „unknown\_questions.xlsx“, gibt pandas einen File-Not-Found-Error zurück. In diesem Fall wird ein leeres DataFrame-Objekt mit einer Spalte und dem Spaltennamen

„Frage“ generiert. Um die Fragen, die der Chatbot nicht verstanden hat, in dem DataFrame-Objekt abzuspeichern, wird in den `get_response()`-Funktion, falls der Schwellenwert von 60% nicht überschritten wird und somit keine passende Antwort zur Frage gefunden wurde, die Funktion `save_unknown_questions()` aufgerufen. Die Funktion wird in den Zeilen 62-68 definiert (siehe Abbildung 17). Diese erhält als Eingabe die Nachricht bzw. Frage des Nutzers, die abgespeichert werden soll und das DataFrame-Objekt „unknown\_questions“. Die Nachricht wird in Zeile 63 unter dem letzten Eintrag in das DataFrame-Objekt eingefügt. Anschließend versucht die Funktion das Objekt abzuspeichern. Gelingt dies nicht, weil die Datei „unknown\_questions.xlsx“ vom Nutzer geöffnet ist, gibt Pandas einen Permission-Error aus. In diesem Fall wird die Excel-Datei geschlossen und danach wird der DataFrame erneut abgespeichert.

Um die Excel-Datei für den Nutzer zu schließen, wird in der `close_question_xlsx_files()`-Funktion in den Zeilen 46-49 eine Liste mit den Namen der beiden Fragenlisten erstellt. Danach prüft die Funktion, ob diese Dateien in der Excel-Desktop-App geöffnet sind. Ist dies der Fall, wird das jeweilige Fenster gespeichert und geschlossen. Andere Excel-Dateien, die der Nutzer möglicherweise geöffnet hat, bleiben geöffnet.

```
45 def close_question_xlsx_files():
46     xlsx_files = [
47         'unknown_questions.xlsx',
48         'wrong_answer.xlsx'
49     ]
50     x = win32com.client.GetActiveObject("Excel.Application")
51     for wb in x.Workbooks:
52         if wb.FullName.split('\\')[-1] in xlsx_files:
53             wb.Close(True)
54         del wb
55     del x
56
62 def save_unknown_questions(msg, unknown_questions):
63     unknown_questions.loc[len(unknown_questions.index)] = msg
64     try:
65         unknown_questions.to_excel(path + '\\unknown_questions.xlsx', index=False)
66     except PermissionError:
67         close_question_xlsx_files()
68     unknown_questions.to_excel(path + '\\unknown_questions.xlsx', index=False)
```

Abbildung 17: Zusammenschnitt chat.py, `close_question_xlsx_files()`-Funktion und `save_unknown_questions()`-Funktion

### 3.2.2 Falsch beantwortete Fragen

Die `get_response()`-Funktion gibt eine Antwort zurück, sobald sich das KNN zu 60% sicher ist, die Intention hinter der Nachricht des Nutzers zu erkennen. Manchmal liegt die KI damit aber falsch. Für diese Fälle wurde das Antwort-melden-Icon in der Menüleiste programmiert. Um die gemeldeten Fragen abzuspeichern, wird zunächst in der Datei „HNU\_Chatbot.py“ auf die

gleiche Weise, wie im vorherigen Teilkapitel, die Datei „wrong\_answer.xlsx“ als ein DataFrame-Objekt geladen. Falls die Datei im vorgegebenen Ordner nicht existiert, wird ein neues DataFrame-Objekt mit den Spalten „Frage“ und „Antwort“ als Variable „wrong\_answer“ erstellt.

```
160     def antw_button_event(self, event=None):
161         # Report wrong answers
162         self.wrong_answer.loc[len(self.wrong_answer.index)] = [self.msg, self.response]
163         self.msg, self.response = '', ''
164         self.wrong_answer = self.wrong_answer[self.wrong_answer['Antwort'].astype(bool)]
165     try:
166         self.wrong_answer.to_excel(self.path + '\\wrong_answer.xlsx', index=False)
167     except PermissionError:
168         close_question_xlsx_files()
169         self.wrong_answer.to_excel(self.path + '\\wrong_answer.xlsx', index=False)
```

**Abbildung 18:** HNU\_Chatbot.py, Zeile 160-169, antw\_button\_event()-Funktion

Das Antwort-melden-Icon führt, wenn angeklickt, die Funktion antw\_button\_event() aus. Zunächst wird in Zeile 162 (siehe Abbildung 18) die letzte Frage des Nutzers und die Antwort des Chatbots darauf, am Ende des DataFrame-Objekts eingefügt. Damit die gleiche Frage und Antwort nicht durch wiederholtes Klicken des Buttons mehrfach in die Tabelle eingetragen wird, werden die Werte für die Nutzernachricht und die Antwort in der folgenden Zeile auf einen leeren String zurückgesetzt. Diese leeren Strings werden durch Drücken des Buttons trotzdem in das DataFrame-Objekt eingetragen. Darum werden in der Zeile 164 nur die Zeilen der Tabelle wieder in der „wrong\_answer“-Variable gespeichert, die in der Spalte „Antwort“ einen Wert und keinen leeren String enthalten. Zum Schluss versucht die Funktion wiederum das DataFrame-Objekt abzuspeichern. Gelingt dies nicht, weil die Excel-Datei bereits geöffnet ist, wird das Excel-Fenster mit der Datei durch die Funktion close\_question\_xlsx\_files() aus der „chat.py“-Datei geschlossen und anschließend die Datei mit den falschen Antworten abgespeichert.

### 3.3 Bereitstellung der Applikation

Mit den letzten Maßnahmen ist die Programmierung des Chatbots abgeschlossen. Um die Nutzung für die Studierenden zu erleichtern, soll das Python-Skript des HNU-Bots im Folgenden in eine ausführbare Applikation umgewandelt werden. Hierbei ist zu beachten, dass verschiedene Betriebssysteme unterschiedliche Dateitypen verlangen. Computer, die mit Windows als Betriebssystem laufen, benötigen den Chatbot in Form einer .exe-Datei, um diesen ausführen zu können. Apple-Geräte, die auf MacOS als Betriebssystem laufen, benötigen eine .app-Datei. Für das Umwandeln des Skripts „HNU\_Chatbot.py“ in eine Anwendung kann die Bibliothek PyInstaller von David Cortesi genutzt werden. Der Nachteil bei dieser Bibliothek ist es jedoch, dass von einem Betriebssystem aus ein Skript immer nur für das eigene Betriebssystem in eine Anwendungsdatei umgewandelt werden kann (vgl. Cortesi, o. J.). Das bedeutet, ein Windows-Rechner kann keine .app-Datei erstellen, sondern nur eine .exe-Datei.

Im folgenden Teil wird der Chatbot nur in das Anwendungsformat für Windows umgewandelt, da es bei der Erstellung einer .app-Datei zu Problemen kam.

### 3.3.1 .exe

Um eine .exe-Datei mit PyInstaller zu erstellen, kann das einfache grafische Interface „auto-py-to-exe“ genutzt werden, um den Befehl für PyInstaller zu generieren. Die Benutzeroberfläche bietet die Möglichkeit - wie in Anhang 5 ersichtlich - verschiedene Optionen für die Erstellung der App auszuwählen. Für die Umwandlung des HNU-Bots muss zunächst unter „Script Location“ der Pfad zum Hauptskript der App angegeben werden. Die nächste Option ist, ob die resultierende Anwendungsdatei eine Datei sein soll oder neben der .exe-Datei ein Ordner voller unterstützender Dateien erstellt wird. Für den Chatbot wird die zweite Option aus dem Grund gewählt, dass die One-File-Option zu einem Problem beim Öffnen der Datei führt. Beim Öffnen des Chatbots soll sich kein Konsolenfenster öffnen. Geschieht dies, muss das Konsolenfenster die gesamte Zeit, in der der Chatbot läuft, geöffnet bleiben, da ansonsten die App nicht funktioniert. Damit die Studierenden von dem Fenster nicht irritiert werden und es schließen, wird es mit der Einstellung komplett weggelassen. Als Icon für die App wird eine ICO-Datei-Version des Icons in Anhang 2 genutzt. Zusätzlich zu den genannten Einstellungsmöglichkeiten werden weitere Dateien, die die App benötigt, angefügt. Damit das Skript des Chatbots ausgeführt werden kann, ist es nötig die Dateien „Icon\_klein.png“, „data.pth“ und „intents\_umlaut.json“ anzufügen, da diese während der Ausführung des Chatbots aufgerufen werden. Desweiteren müssen die Ordner „customtkinter“ und „openpyxl“ mit in den Ordner, der erstellt wird, aufgenommen werden, da das Programm ansonsten Probleme hat, auf diese Dateien zuzugreifen, wenn der Chatbot geöffnet werden soll. In den letzten beiden Abschnitten der Konfiguration ist es im Fall des HNU-Bots nicht nötig weitere Optionen auszufüllen.

Bevor der CONVERT-Button gedrückt werden kann, musste auf dem Gerät, auf dem die Chatbot-App mit auto-py-to-exe erstellt wurde, der Ordner „pathlib-1.0.1.dist-info“ aus dem Ordner „C:\Users\\anaconda3\Lib\site-packages“ entfernt werden, da diese Bibliothek zu einem Fehler bei der Erstellung der App führt.

Auto-py-to-exe erstellt auf Grundlage der getätigten Einstellungen den folgenden Code und führt diesen mit Klicken des CONVERT-Buttons aus.

```
pyinstaller --noconfirm --onedir --windowed --icon "C:/Users/chris/Documents/Uni/HNU Wirtschaftsprsychologie/17_Bachelorarbeit/HNU Bot/Taskleisten_Icon.ico" --add-data "C:/Users/chris/Documents/Uni/HNU Wirtschaftsprsychologie/17_Bachelorarbeit/HNU Bot/data.pth;." --add-data "C:/Users/chris/Documents/Uni/HNU Wirtschaftsprsychologie/17_Bachelorarbeit/HNU Bot/Icon_klein.png;." --add-data "C:/Users/chris/Documents/Uni/HNU Wirtschaftsprsychologie/17_Bachelorarbeit/HNU Bot/intents_umlaut.json;." --add-data "C:/Users/chris/Documents/Uni/HNU Wirtschaftsprsychologie/17_Bachelorarbeit/HNU Bot/customt-
```

```
kinter;customtkinter/" --add-data "C:/Users/chris/Documents/Uni/HNU Wirtschaftspsychologie/17_Bachelorarbeit/HNU Bot/openpyxl;openpyxl/" "C:/Users/chris/Documents/Uni/HNU Wirtschaftspsychologie/17_Bachelorarbeit/HNU Bot/HNU_Chatbot.py"
```

Abschließend ist die fertige .exe-Datei im Ordner „C:\Users\<>name>\output“ zu finden.

### 3.3.2 .app

Bevor die Chatbot-App in eine .app-Datei umgewandelt werden kann, muss der Code für den Chatbot noch an einigen Stellen auf die Apple-Software angepasst werden. So müssen beispielsweise bei den Funktionen zum Einlesen und Abspeichern der falsch oder gar nicht beantworteten Fragen die Schrägstriche „\“ vor dem Dateinamen entfernt werden, da diese sonst mit im Namen der Datei erscheinen. Darüber hinaus ist die Funktion `close_question_xlsx_files()` bei Apple redundant. Diese nutzt in der Windows-Version die Bibliothek `win32com`, die dabei hilft, Operationen an einem Windows-Gerät durchzuführen, um geöffnete Excel-Dateien zu schließen, bevor das Programm versucht, sie zu überschreiben. Bei Apple funktioniert diese Bibliothek allerdings nicht und wird in diesem Fall auch nicht benötigt, da eine Excel-Datei trotzdem gespeichert wird, auch wenn der Nutzer sie geöffnet hat.

Um eine Applikation einer .app-Datei umzuwandeln, kann ebenfalls die Bibliothek `PyInstaller` genutzt werden. Es gibt dafür allerdings kein Interface wie `auto-py-to-exe`, das das Erstellen des Befehls erleichtert. Daher muss dies selbst vorgenommen werden. Alternativ kann auch die Bibliothek `py2app` verwendet werden, um eine Anwendungsdatei zu erstellen.

Da das Umwandeln für Apple-Geräte nicht funktioniert hat, wurde das Skript „`install_dependencies.py`“ erstellt, in dem sich alle Befehle zum Installieren der für den Chatbot nötigen Bibliotheken befinden. Zudem müssen für die Sprachverarbeitung noch weitere Daten für die Bibliothek „`nltk`“ heruntergeladen werden. Auch den Code hierfür enthält das Skript (siehe Anhang 7). Dies ermöglicht es den Studierenden, die einen Apple-PC besitzen, den Chatbot dennoch zu nutzen.

### 3.3.3 Ausgabe an die Studierende

Zuletzt stellt sich die Frage, wie der Chatbot an die Studierenden ausgegeben werden kann. Da der Ordner mit der Anwendungsdatei eine Größe von 1,75 GB hat, wird der Ordner zu einer .zip-Datei komprimiert. Diese hat schon eine reduzierte Größe von nur noch 554 MB. Normalerweise lassen sich per E-Mail Dateien bis zu einer Größe von 20 MB versenden, daher fällt diese Möglichkeit, den Chatbot den Teilnehmern der Kurse zukommen zu lassen, weg. Eine bessere Alternative ist es eher, den komprimierten Ordner für Windows-Geräte auf der Moodle-Kursseite hochzuladen. Von dort aus können sich die Studierenden die Dateien herunterladen und auf ihrem Computer entpacken.

Nach dem Entpacken finden die Studierenden in dem Ordner einen Unterordner und zwei Dateien. In dem Unterordner befinden sich alle Dateien, die der Chatbot braucht, um zu starten. Hier werden auch die Excel-Dateien mit den unbeantworteten Fragen abgespeichert. Außerdem liegt die HNU\_Chatbot.exe in diesem Ordner ab, mit der der Chatbot gestartet wird. Damit die Studierenden nicht den Ordner nach der Anwendungsdatei durchsuchen müssen, wurde eine Verknüpfung zu dieser erstellt und außerhalb des Ordners unter den Namen „HNU\_Chatbot“ abgelegt. Mit dieser lässt sich der Chatbot ebenfalls starten. Zuletzt befindet sich noch eine „READ ME.txt“ Datei in dem Hauptordner (siehe Anhang 6). In dieser können die Studierenden eine Erklärung zum Chatbot finden, wie sie den Chatbot starten können und welche zusätzlichen Funktionen sich innerhalb der Menüleiste befinden. Ebenfalls wird nochmal darauf hingewiesen, dass die Studierenden die Excel-Dateien in dem Unterordner finden und diese an Frau Prof. Dr. Köppl schicken können.

Die Chatbot-App hat zusätzlich ein individuelles Icon erhalten, das sowohl bei der App zu sehen ist (siehe Abbildung 19), als auch in der Taskleiste, wenn die App geöffnet ist.



**Abbildung 19: Icon im Ordner**

Für Apple-Nutzer wurde ein eigener Ordner mit dem Programmcode und den nötigen Dateien erstellt, den sie herunterladen können, um den Chatbot über Python selbst zu starten. Eine umgeschriebene Version der READ ME-Datei erklärt dabei das genaue Vorgehen.

## **4 Schlussbetrachtung**

Abschließend soll die Weiterentwicklung des Chatbots reflektiert werden. Dazu wird zunächst die Arbeit zusammengefasst und ein Fazit gezogen. In einem Ausblick wird erläutert, was an dem Chatbot noch weiter verbessert werden könnte.

### **4.1 Zusammenfassung**

Das Ziel in dieser Arbeit war es, den Chatbot-Prototyp so weiterzuentwickeln, dass eine fertige Anwendung mit modernem, intuitiven Design erstellt wird, die an die Studierenden ausgegeben werden kann. Hierzu wurde zunächst ein Überblick über den aktuellen Stand zu Beginn der Arbeit geschaffen, um Punkte zu identifizieren, die in der Weiterentwicklung des Chatbots professionalisiert werden sollen. Zum Verständnis wurden dabei die grundlegenden Begriffe und Konzepte zu Chatbots, KI, ML und NLP erklärt und anschließend damit begonnen, die

Benutzeroberfläche des Chatbots neu zu gestalten. Mit der Bibliothek CustomTkinter wurde ein modernes Interface erstellt, in dem die Nachrichten des Chatbots und der Nutzer in Form von Sprechblasen dargestellt werden. Es wurden Funktionen definiert, die dafür sorgen, dass das Interface einheitlich und geordnet bleibt, auch wenn die Größe des Fensters verändert wird. Zudem wurden dem Chatbot weitere Funktionen hinzugefügt. Dazu zählt das automatische Abspeichern von Fragen, die der Chatbot nicht verstanden hat und eine Menüleiste mit drei Knöpfen, um falsch beantwortete Fragen abzuspeichern, den Chatverlauf in ein Word-Dokument zu exportieren und den Erscheinungsmodus von hell zu dunkel zu wechseln. Die abgespeicherten Nachrichten, die der Chatbot nicht richtig beantworten konnte, sollen langfristig dazu genutzt werden, um das KNN nachzutrainieren. Zusätzlich wurde eine Pop-up-Nachricht, mit Informationen darüber, wo diese Fragen hingeschickt werden sollen, erstellt. Diese erscheint, wenn der Chatbot geschlossen wird. Abschließend wurde das Python-Skript in eine Anwendungsdatei umgewandelt, die für die Studierenden zur Nutzung in den Moodle-Kursordner hochgeladen werden kann.

#### **4.2 Fazit**

Für die Weiterentwicklung des Chatbots wurden drei Hauptziele formuliert. Zunächst sollte die Benutzeroberfläche des Chatbots neu gestaltet werden. Im Rahmen dieser Arbeit konnte eine moderne Design für das Interface erstellt werden. Zudem werden die Chatnachrichten nun in Form von Sprechblasen angezeigt und nicht mehr in einem Textfeld untereinander. Dabei wurde darauf geachtet, dass wenn die Größe des Fensters verändert wird, die Nachrichten trotzdem weiterhin ordentlich angezeigt werden. Damit ist das erste Ziel dieser Arbeit erreicht. Des Weiteren sollen nicht oder falsch beantwortete Fragen abgespeichert werden, um daran zu einem späteren Zeitpunkt die KI des Chatbots nachzutrainieren zu können. Auch dieses Ziel wurde erreicht, indem der Chatbot nun automatisch Fragen abspeichert, die er nicht verstanden hat oder die Studierenden Fragen, die falsch beantwortet wurden, melden können. Allerdings müssen die Studierenden die Excel-Dateien mit diesen Fragen manuell an Frau Prof. Dr. Köppl senden. Das letzte Ziel war es, eine Anwendungsdatei zu erstellen, mit der es möglich ist, den Chatbot zu starten, ohne selbst den Programmcode ausführen zu müssen. Dies ist für Windows-Geräte gelungen. Studierende, die Windows als Betriebssystem nutzen, können die erstellte .exe-Datei ausführen, ohne Python zu öffnen und manuell den Programmcode auszuführen. Apple-Nutzer müssen sich leider mit dem Programmcode zufrieden geben und die App von Python aus starten lassen. Da der Großteil der Studierenden jedoch Computer mit Windows als Betriebssystem nutzen werden, wurde dieses Ziel ebenfalls in ausreichendem Maße erreicht.

Zusätzlich zu den Zielen, die für die Weiterentwicklung gesetzt wurden, wurde noch die Funktion erstellt, dass die Studierenden ihren Chatverlauf mit dem HNU-Bot als ein Word-Dokument abspeichern können, um diese beispielsweise bei ihren Lernunterlagen abzulegen.

### 4.3 Ausblick

Der nächste Schritt im Projekt des HNU-Chatbots ist es, den Bot an die Studierenden auszugeben und von ihnen testen zu lassen. Mit den gesammelten unbeantworteten Fragen kann anschließend die KI des Chatbots nachtrainiert werden, denn auch wenn der Chatbot in den Tests eine Genauigkeit von 90-100% erreicht hatte, sind während der Arbeit an dem Chatbot einige Probleme beim Beantworten von Fragen, für den Chatbot aufgetaucht. Zum Beispiel versteht der HNU-Bot die Fragen „Was ist eine for-Schleife?“, „Was ist eine while-Schleife?“ oder „Was macht eine Schleife?“ nicht. Andererseits gibt er bei den Fragen „Was ist eine Funktion?“ oder „Ich habe Fragen zur Vorlesung“ aufgrund der Schlüsselwörter „Ich“ und „Funktion“ die Antwort zum Tag „Schleifen“ aus. Beim Nachtrainieren des Chatbots sollte sich dabei eher an den Vorschlag von Hundertmark (vgl. 2021, S. 163), mindestens 15 mögliche Fragen pro Intent zu definieren, gehalten werden. Hierzu können die vom Chatbot ausgespeicherten Fragen eingepflegt werden. Eine Herausforderung für den Chatbot beim Erkennen der Intention des Nutzers stellen Tippfehler da. Daher kann in einer überarbeiteten Version des Chatbots ein System eingebaut werden, um beim Eingeben einer Nachricht Rechtschreibfehler des Nutzers zu identifizieren und hervorzuheben, damit diese vor dem Absenden der Frage verbessert werden können. Weitergedacht kann die Chatbot-App auch für weitere Vorlesungen trainiert werden, um in verschiedenen Modulen Anwendung zu finden. Bevor das Anwendungsfeld jedoch ausgeweitet wird, sollte eine Umfrage dazu gemacht werden, welche Akzeptanz der Chatbot bei den Studierenden auslöst und welche weiteren Funktionen sie sich wünschen würden, damit sie den Bot häufiger nutzen. Zudem kann erneut versucht werden, auch eine Anwendungsdatei für Apple-Geräte zu erstellen, damit Apple-Nutzer ebenfalls den Chatbot als Anwendung starten können und nicht auf den Programmcode zurückgreifen müssen. Einer ersten Testphase mit Studierenden der IT-Vorlesungen steht soweit aber nichts mehr im Weg.

## Literaturverzeichnis

Boersch, I., Heinsohn, J., & Socher, R. (2007). *Wissensverarbeitung: Eine Einführung in die Künstliche Intelligenz für Informatiker und Ingenieure* (2. Aufl.). Spektrum Akademischer Verlag.

Buduma, N., & Locascio, N. (2017). *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. O'Reilly Media.

Cortesi, D. (o. J.). *PyInstaller*. <https://pyinstaller.org/en/stable/>

Goldstein, E. B. (2015). *Wahrnehmungspsychologie: Der Grundkurs* (K. R. Gegenfurtner, Hrsg.; K. Neuser-von Oettingen & G. Plata, Übers.; 9. Aufl.). Springer. <https://doi.org/10.1007/978-3-642-55074-4>

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.

Güll, R. (2010). *Zur Statistik des deutschen Wortschatzes*. <https://www.statistik-bw.de/Service/Veroeff/Monatshefte/20100911#ft-nr-rel1>

Hirschle, J. (2022). *Deep Natural Language Processing: Einstieg in Word Embedding, Sequence-to-Sequence-Modelle und Transformer mit Python*. Hanser.

Hoffmann, A. (2019). *Chatbots: Einführung in die Zukunft von Marketing, PR und CRM*. Franzis.

Hundertmark, S. (2021). *Digitale Freunde: Wie Unternehmen Chatbots erfolgreich einsetzen können*. Wiley-VCH GmbH.

Khan, R., & Das, A. (2018). *Build Better Chatbots*. Springer Science+Business Media.

Muldowney, O. (2017). *Chatbots An Introduction And Easy Guide To Making Your Own*. Curses & Magic.

Nussbaumer Knaflic, C. (2017). *Storytelling mit Daten: Die Grundlagen der effektiven Kommunikation und Visualisierung mit Daten* (M. Kauschke, Übers.). Verlag Franz Vahlen.

Rey, G. D., & Wender, K. F. (2018). *Neuronale Netze: Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung* (3. Aufl.). Hogrefe.

Schiele, J. (2023). *Die Weiterentwicklung eines Chatbot-Prototyps zur automatisierten Beantwortung von Studierendenfragen zu Vorlesungsinhalten*.

Schimansky, T. (o. J.). *CustomTkinter*. <https://customtkinter.tomschimansky.com/>

Stevoisiak. (2018). *Tooltips in Tkinter*. <https://stackoverflow.com/questions/3221956/how-do-i-display-tooltips-in-tkinter>

Yik, H. (2019). *Tkinter Chatbot Speech Bubbles design*. <https://stackoverflow.com/questions/56005668/tkinter-chatbot-speech-bubbles-design>

# Anhang

## Anhang 1: HNU\_Chatbot.py

```
1 import tkinter as tk
2 from tkinter.messagebox import askokcancel, INFO
3 import customtkinter as ctk
4 import pandas as pd
5 from docx import Document
6 from datetime import datetime
7 from chat import get_response, path, close_question_xlsx_files
8
9 ctk.set_appearance_mode('System') # Modes: 'System' (standard), 'Dark', 'Light'
10 ctk.set_default_color_theme('blue') # Themes: 'blue' (standard), 'green', 'dark-blue'
11
12 CANVAS_COLOR = '#FFFFFF'
13 USER_BUBBLE = '#43CD80'
14 BOT_BUBBLE = '#7A8888'
15 FONT_COLOR = '#FFFFFF'
16 BUBBLE_FONT = ('Arial',12)
17
18 class ChatApplication:
19     def __init__(self, path):
20         # Configure Window
21         self.window = tk.Tk()
22         self.window.protocol('WM_DELETE_WINDOW', self.callback)
23         self.window.geometry(f'{550}x{650}')
24
25         # Icon
26         icon = tk.PhotoImage(file='Icon_klein.png')
27         self.window.tk.call('wm', 'iconphoto', self.window._w, icon)
28
29         # Layout
30         self._setup_main_window()
31
32         # Wrong Answers
33         self.path = path
34         try:
35             self.wrong_answer = pd.read_excel(self.path + '\\wrong_answer.xlsx')
36         except FileNotFoundError:
37             self.wrong_answer = pd.DataFrame(columns=['Frage', 'Antwort'])
38
39         # Bubbles
40         self.user_bubbles = []
41         self.bot_bubbles = []
42
43     def run(self):
44         self.window.mainloop()
45
46     def callback(self):
47         if askokcancel('HNU-Bot', 'Helfen Sie uns den HNU-Bot weiterzuentwickeln und senden Sie uns Ihre unbeantworteten
48             self.window.destroy()
49
50     def _setup_main_window(self):
51         self.window.update()
52         self.window.title('HNU-Bot')
53         self.window.configure(width=470, height=650)
54
55         self.frame = ctk.CTkFrame(master=self.window)
56         self.frame.pack(padx=0, pady=0, fill='both', expand=True)
57
58         self.label = ctk.CTkLabel(
59             self.frame,
60             text = 'Willkommen zum HNU ChatBot',
61             font = ctk.CTkFont('Arial', 21, 'bold')
62         )
63         self.label.place(relwidth=1, relheight=0.05, rely=0.01)
64
65         # Textfeld
66         self.canvas = ctk.CTkCanvas(self.window, bg=CANVAS_COLOR)
67         self.canvas.place(relheight=0.76, relwidth=0.825, rely=0.11, relx=0.11)
68
69         # Scrollbar
70         self.scrollbar = ctk.CTkScrollbar(
71             self.canvas,
72             command=self.canvas.yview
73         )
74         cw = self.get_canvas_width()
75         self.scrollbar.place(relheight=1, relx=(cw-17)/cw)
76         self.canvas.configure(yscrollcommand=self.scrollbar.set)
77         self.canvas.bind('<Configure>', self.update_scrollbar_position)
78         self.canvas.bind_all('<MouseWheel>', self.on_mousewheel)
79
```

```

80 # Entry
81 self.msg_entry = ctk.CTkEntry(
82     master=self.frame,
83     placeholder_text='Bitte stellen Sie Ihre Frage'
84 )
85 self.msg_entry.place(relwidth=0.695, relheight=0.07, rely=0.9, relx=0.1)
86 self.msg_entry.focus()
87 self.msg_entry.bind('<Return>', self._on_enter_pressed)
88
89 # Send Button
90 self.send = ctk.CTkButton(
91     master=self.frame,
92     text='Senden',
93     command=lambda: self._on_enter_pressed(None)
94 )
95 self.send.place(relwidth=0.15, relheight=0.07, rely=0.9, relx=0.8)
96
97 # Sidebar
98 # Define the icons to be shown and resize it
99 self.antw = '⚠️'
100 self.save = '💾'
101 self.mode = '💡'
102
103 self.sidebar = ctk.CTkFrame(self.frame, width=40, corner_radius=0)
104 self.sidebar.place(relheight=1, anchor='nw')
105
106 self.antw_b = ctk.CTkLabel(self.sidebar, text=self.antw, font=(0,15))
107 self.save_b = ctk.CTkLabel(self.sidebar, text=self.save, font=(0,15))
108 self.mode_b = ctk.CTkLabel(self.sidebar, text=self.mode, font=(0,15))
109
110 self.antw_b.place(x=10, y=10)
111 self.save_b.place(x=10, y=50)
112 self.mode_b.place(x=10, y=87)
113
114 # Tooltip for Icons
115 CreateToolTip(self.antw_b, 'Antwort melden')
116 CreateToolTip(self.save_b, 'Chatverlauf speichern')
117 CreateToolTip(self.mode_b, 'Hell-/Dunkelmodus')
118
119 # Binding functions to buttons
120 self.antw_b.bind('<Button-1>', self.antw_button_event)
121 self.save_b.bind('<Button-1>', self.save_button_event)
122 self.mode_b.bind('<Button-1>', self.mode_button_event)
123
124 # Chat Bubbles
125 self.frame.bind('<Configure>', self.update_user_bubbles)
126 self.frame.bind('<Configure>', self.update_bot_bubbles)
127 self.frame.bind('<Configure>', self.update_bubble_gap)
128 self.frame.bind('<Configure>', self.update_scrollregion)
129
130 def _on_enter_pressed(self, event):
131     self.msg = self.msg_entry.get()
132     if self.msg != '':
133         self.send_message(self.msg)
134
135 def send_message(self, msg):
136     user = UserBubble(self.canvas, message=msg)
137     self.response = get_response(msg)
138     bot = BotBubble(self.canvas, message=self.response)
139     self.msg_entry.delete(0, ctk.END)
140     self.user_bubbles.append(user)
141     self.bot_bubbles.append(bot)
142     self.update_scrollregion()
143
144 def get_canvas_width(self, event=None):
145     return self.canvas.winfo_width()
146
147 def update_scrollbar_position(self, event=None):
148     cw = self.get_canvas_width()
149     self.scrollbar.place(relheight=1, relx=(cw-17)/cw)
150
151 def update_scrollregion(self, event=None):
152     self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))
153     self.canvas.yview_moveto(1) # scroll to bottom
154
155 def _on_mousewheel(self, event):
156     self.canvas.yview_scroll(int(-1*(event.delta/120)), 'units')
157

```

```

158 # Sidebar functions
159 #=====
160 def antw_button_event(self, event=None):
161     # Report wrong answers
162     try:
163         self.wrong_answer.loc[len(self.wrong_answer.index)] = [self.msg, self.response]
164     except AttributeError:
165         return
166     self.msg, self.response = '', ''
167     self.wrong_answer = self.wrong_answer[self.wrong_answer['Antwort'].astype(bool)]
168     try:
169         self.wrong_answer.to_excel(self.path + '\\wrong_answer.xlsx', index=False)
170     except PermissionError:
171         close_question_xlsx_files()
172         self.wrong_answer.to_excel(self.path + '\\wrong_answer.xlsx', index=False)
173
174 def save_button_event(self, event):
175     # Save chat as .docx-file
176     datum = datetime.today()
177     t = datum.strftime('%d')
178     m = datum.strftime('%m')
179     j = datum.strftime('%Y')
180     chat = ''
181     # Get text from messages
182     for idx, user_bubble in enumerate(self.user_bubbles):
183         chat += '\nIch:\n' + user_bubble.get_text()
184         chat += '\n\nNU Chatbot:\n' + self.bot_bubbles[idx].get_text()
185         chat += '\n\n'
186     # Save chat in .docx
187     files = [('Textdokument', '*.docx')]
188     file = tk.filedialog.asksaveasfile(mode='w', filetypes=files, defaultextension=files)
189     document = Document()
190     document.add_heading(f'Chatverlauf ({t}.{m}.{j})', level=1)
191     document.add_paragraph(chat)
192     if file is not None:
193         document.save(file.name)
194         file.close()
195
196 def mode_button_event(self, event):
197     # Change appearance mode
198     if ctk.get_appearance_mode() == 'Light':
199         ctk.set_appearance_mode('Dark')
200     else:
201         ctk.set_appearance_mode('Light')
202
203 # Update functions
204 #=====
205 def update_user_bubbles(self, event=None):
206     w = self.canvas.winfo_width()
207
208     # Adjust width of UserBubble to width of canvas
209     for bubble in self.user_bubbles:
210         bubble.config_wraplength()
211
212     # Move position of UserBubble
213     for bubble_id in self.canvas.find_withtag('userBubble'):
214         bubble_coords = self.canvas.bbox(bubble_id)
215         diff = bubble_coords[2] - bubble_coords[0]
216         self.canvas.move(bubble_id, w - bubble_coords[0] - diff - 25, 0)
217
218     # Move position of triangle
219     triangle_id = bubble_id + 1
220     triangle_coords = self.canvas.bbox(triangle_id)
221     self.canvas.move(triangle_id, w - triangle_coords[0] - 27, 0)
222
223 def update_bot_bubbles(self, event=None):
224     # Adjust width of BotBubble to width of canvas
225     for bubble in self.bot_bubbles:
226         bubble.config_wraplength()
227
228 def update_bubble_gap(self, event=None):
229     for bubble_id in self.canvas.find_all():
230         # Skip triangle
231         if (bubble_id % 2) == 0 or bubble_id == 1:
232             continue
233         bubble_coords0 = self.canvas.bbox(bubble_id - 2)
234         bubble_coords = self.canvas.bbox(bubble_id)
235
236         if (diff := bubble_coords[1] - bubble_coords0[3]) != 10:
237             # move position of BotBubble
238             self.canvas.move(bubble_id, 0, -(diff-10))
239
240             # Move position of triangle
241             triangle_id = bubble_id + 1
242             self.canvas.move(triangle_id, 0, -(diff-10))
243

```

```

244 class BotBubble:
245     def __init__(self, master, message=''):
246         self.master = master
247         bbox = self.master.bbox(tk.ALL)
248         y = (bbox[3] if bbox else 0) + 10
249         self.frame = tk.Frame(master, bg=BOT_BUBBLE)
250         self.i = self.master.create_window(15, y, window=self.frame, anchor='nw')
251         self.botLabel = ctk.CTkLabel(
252             self.frame,
253             text=message,
254             justify=ctk.LEFT,
255             font=BUBBLE_FONT,
256             text_color=FONT_COLOR,
257             fg_color=(BOT_BUBBLE),
258             bg_color=(CANVAS_COLOR),
259             padx=1,
260             pady=7,
261             corner_radius=8
262         )
263         # Bubble width
264         self.botLabel.configure(wraplength=(self.get_master_width()-75))
265         self.botLabel.grid(row=1, column=0)
266         self.master.update()
267         self.master.create_polygon(self.draw_triangle(self.i), fill=BOT_BUBBLE, outline=BOT_BUBBLE)
268
269     def draw_triangle(self, widget):
270         x1, y1, x2, y2 = self.master.bbox(widget)
271         return x1, y1+17, x1-8, y1+7.5, x1, y1+8
272
273     def get_text(self):
274         return self.botLabel.cget('text')
275
276     def get_master_width(self, event=None):
277         return self.master.winfo_width()
278
279     def config_wraplength(self):
280         self.botLabel.configure(wraplength=(self.get_master_width()-75))
281         self.master.update()
282
283
284 class UserBubble:
285     def __init__(self, master, message=''):
286         self.master = master
287         bbox = self.master.bbox(tk.ALL)
288         y = (bbox[3] if bbox else 0) + 10
289         self.frame = tk.Frame(master, bg=USER_BUBBLE)
290         w = self.master.winfo_width()
291         self.i = self.master.create_window(w-25, y, window=self.frame, anchor='ne', tag='userBubble')
292         self.userLabel = ctk.CTkLabel(
293             self.frame,
294             text=message,
295             justify=ctk.LEFT,
296             font=BUBBLE_FONT,
297             text_color=FONT_COLOR,
298             fg_color=(USER_BUBBLE),
299             bg_color=(CANVAS_COLOR),
300             padx=1,
301             pady=7,
302             corner_radius=8
303         )
304         # Bubble width
305         self.userLabel.configure(wraplength=(self.get_master_width()-75))
306         self.userLabel.grid(row=1, column=0)
307         self.master.update()
308         self.master.create_polygon(self.draw_triangle(self.i), fill=USER_BUBBLE, outline=USER_BUBBLE)
309
310     def draw_triangle(self, widget):
311         x1, y1, x2, y2 = self.master.bbox(widget)
312         return x2, y1+17, x2+8, y1+7.5, x2, y1+8
313
314     def get_text(self):
315         return self.userLabel.cget('text')
316
317     def get_master_width(self, event=None):
318         return self.master.winfo_width()
319
320     def config_wraplength(self):
321         self.userLabel.configure(wraplength=(self.get_master_width()-75))
322         self.master.update()
323

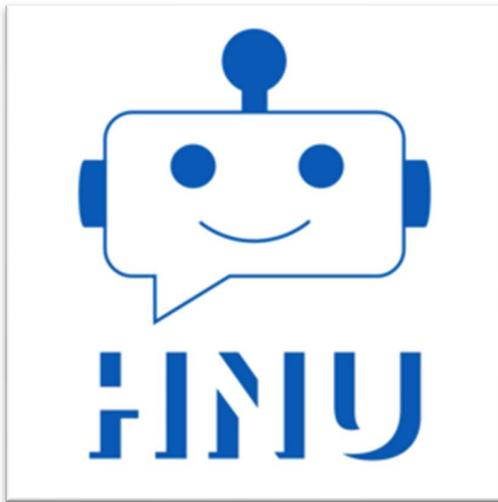
```

```

324 class CreateToolTip(object):
325     def __init__(self, widget, text='widget info'):
326         self.waittime = 750 # Milliseconds
327         self.wraplength = 180 # Pixels
328         self.widget = widget
329         self.text = text
330         self.widget.bind("<Enter>", self.enter)
331         self.widget.bind("<Leave>", self.leave)
332         self.widget.bind("<ButtonPress>", self.leave)
333         self.id = None
334         self.tw = None
335
336     def enter(self, event=None):
337         self.schedule()
338
339     def leave(self, event=None):
340         self.unschedule()
341         self.hidetip()
342
343     def schedule(self):
344         self.unschedule()
345         self.id = self.widget.after(self.waittime, self.showtip)
346
347     def unschedule(self):
348         id = self.id
349         self.id = None
350         if id:
351             self.widget.after_cancel(id)
352
353     def showtip(self, event=None):
354         x, y, cx, cy = self.widget.bbox("insert")
355         x += self.widget.winfo_rootx() + 25
356         y += self.widget.winfo_rooty() + 20
357         # Creates a toplevel window
358         self.tw = tk.Toplevel(self.widget)
359         # Leaves only the label and removes the app window
360         self.tw.wm_overrideredirect(True)
361         self.tw.wm_geometry("%d+%d" % (x, y))
362         label = tk.Label(self.tw, text=self.text, justify='left',
363                         background="#ffffff", relief='solid', borderwidth=1,
364                         wraplength = self.wraplength)
365         label.pack(ipadx=1)
366
367     def hidetip(self):
368         tw = self.tw
369         self.tw = None
370         if tw:
371             tw.destroy()
372
373
374 app = ChatApplication(path)
375 app.run()
376

```

Anhang 2: HNU\_Chatbot Icon



### Anhang 3: chat.py

```
1 import random
2 import json
3 import torch
4 from model import NeuralNet
5 from nltk_utils import *
6 import pandas as pd
7 from inspect import getsourcefile
8 from os.path import abspath
9 import win32com.client
10
11 # Checken ob GPU Support verfügbar
12 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
13
14 # Intents öffnen (r für Readmode)
15 with open("intents_umlaute.json", "r") as json_data:
16     intents = json.load(json_data)
17
18 # Trainiertes Modell öffnen
19 FILE = "data.pth"
20 data = torch.load(FILE)
21
22 input_size = data["input_size"]
23 hidden_size = data["hidden_size"]
24 output_size = data["output_size"]
25 all_words = data["all_words"]
26 tags = data["tags"]
27 model_state = data["model_state"]
28
29 # Modell erstellen und laden
30 model = NeuralNet(input_size, hidden_size, output_size).to(device)
31 model.load_state_dict(model_state) # Gelernte Parameter in das Modell laden
32 model.eval()
33
34 # Chat erstellen
35 bot_name = "HNU_Bot"
36
37 p = abspath(getsourcefile(lambda:0))
38 path = ''
39 for idx, way in enumerate(p.split('\\')):
40     if idx != len(p.split('\\))-1:
41         path = path + way + '\\'
42
43 # Falls eine Excel Datei geöffnet ist, kann pandas sie nicht einlesen
44 # Daher müssen die Dateien zum Speichern der Fragen geschlossen werden
45 def close_question_xlsx_files():
46     xlsx_files = [
47         'unknown_questions.xlsx',
48         'wrong_answer.xlsx'
49     ]
50     x = win32com.client.GetObject("Excel.Application")
51     for wb in x.Workbooks:
52         if wb.FullName.split('\\')[-1] in xlsx_files:
53             wb.Close(True)
54     del wb
55     del x
56
57 try:
58     unknown_questions = pd.read_excel(path + '\\unknown_questions.xlsx')
59 except FileNotFoundError:
60     unknown_questions = pd.DataFrame(columns=['Frage'])
61
62 def save_unknown_questions(msg, unknown_questions):
63     unknown_questions.loc[len(unknown_questions.index)] = msg
64     try:
65         unknown_questions.to_excel(path + '\\unknown_questions.xlsx', index=False)
66     except PermissionError:
67         close_question_xlsx_files()
68         unknown_questions.to_excel(path + '\\unknown_questions.xlsx', index=False)
69
70 def replace_umlauts(txt):
71     umlauts = {
72         'Ä': 'ä',
73         'Å': 'A',
74         'Ö': 'ö',
75         'Ä-': 'Ö',
76         'ÄW': 'u',
77         'Äe': 'U',
78         'ÄV': 'B',
79         'äe': '-',
80         '|': '\n\n'
81     }
82     for k,v in umlauts.items():
83         txt = txt.replace(k,v)
84     return txt
```

```

85
86 def get_response(msg):
87     # Sprachverarbeitung der Eingaben des Anwenders (Tokenize, Stem, Bow)
88     sentence = tokenize(msg)
89     X = bag_of_words(sentence, all_words)
90     X = X.reshape(1, X.shape[0]) # Modell erwartet diese Form (1 Zeile und keine Spalte)
91     X = torch.from_numpy(X).to(device) # Umwandlung Bow von NumPy Array in Torch Tensor
92
93     # Antwort ermitteln
94     output = model(X) # Modell öffnen und auf Bag of Words (X) anwenden
95     _, predicted = torch.max(output, dim=1)
96
97     tag = tags[predicted.item()]
98
99     # Prüfen ob die Wahrscheinlichkeit für das Tag Schwellenwert erreicht
100    probs = torch.softmax(output, dim=1) # Softmax um Wahrscheinlichkeiten zu bekommen
101    prob = probs[0][predicted.item()] # Wahrscheinlichkeit für jeweiliges Tag
102    if prob.item() > 0.60:
103        for intent in intents["intents"]:
104            if tag == intent["tag"]:
105                return replaceumlauts(random.choice(intent["responses"]))
106
107    # Falls der Schwellenwert nicht erreicht wurde oder keine passende Antwort gefunden wurde
108    else:
109        save_unknown_questions(msg, unknown_questions)
110        return "Verzeihung, ich habe das nicht verstanden. Könnten Sie die Frage bitte umformulieren?"
111

```

## Anhang 4: Beispiel Chatverlauf

### Chatverlauf (13.07.2023)

Ich:  
Hallo HNU-Bot!

HNU Chatbot:  
Ich grüße Sie

Ich:  
Welche Funktion hat eine for-Schleife?

HNU Chatbot:  
Schleifen sind Kontrollstrukturen zur wiederholten Ausführung von Anweisungen. Anweisungen werden solange wiederholt, bis eine bestimmte Bedingung nicht mehr erfüllt ist. Alle Schleifen müssen eine solche Bedingung enthalten. Schleifen dienen somit zur kompakteren Darstellung umfangreicher Aufgaben. WHILE-Schleife

FOR-Schleife (in Python nur zur Iteration über ein Sequenz von Objekten. In anderen Programmiersprachen ähnlich verwendet wie WHILE-Schleife.)

Ich:  
Was ist ein Array?

HNU Chatbot:  
Ein Array ist eine Datenstruktur, die mehrere Werte unter einem gemeinsamen Namen speichert.

Ich:  
Vielen Dank!

HNU Chatbot:  
Nichts zu danken.

## Anhang 5: auto-py-to-exe

Auto Py To Exe

GitHub  Help Post Ne

Language:

### Script Location

### Onefile (--onedir / --onefile)

### Console Window (--console / --windowed)

### Icon (--icon)

### Additional Files (--add-data)

<input type="text" value="C:/Users/chris/Documents/Uni/HNU Wirtsch"/>	<input type="text" value="."/>	<input type="button" value="[-]"/>
<input type="text" value="C:/Users/chris/Documents/Uni/HNU Wirtsch"/>	<input type="text" value="."/>	<input type="button" value="[-]"/>
<input type="text" value="C:/Users/chris/Documents/Uni/HNU Wirtsch"/>	<input type="text" value="."/>	<input type="button" value="[-]"/>
<input type="text" value="C:/Users/chris/Documents/Uni/HNU Wirtsch"/>	<input type="text" value="customtkinter/"/>	<input type="button" value="[-]"/>
<input type="text" value="C:/Users/chris/Documents/Uni/HNU Wirtsch"/>	<input type="text" value="openpyxl/"/>	<input type="button" value="[-]"/>

*If you want to put files in the root directory, put a period (.) in the destination.*

### Advanced

### Settings

### Current Command

```
pyinstaller --noconfirm --onedir --windowed --icon  
"C:/Users/chris/Documents/Uni/HNU Wirtschaftspsychologie/17_Bachelorarbeit/HNU
```

## Anhang 6: READ ME.txt

Willkommen beim HNU Chatbot!

Sie finden hier die App zum Chatbot.

Hinweis: Damit die App starten kann, benötigt sie alle Dateien, die sich in diesem Ordner befinden, im gleichen Verzeichnis wie die App selbst. Damit Sie die App an einem beliebigen Ort hinterlegen können, von wo aus Sie auf dem Chatbot zugreifen, haben wir die Verknüpfung "HNU\_Chatbot" erstellt, die Sie an jedem beliebigen Ort abspeichern können.

Funktionen:

Neben der Hauptfunktion des Chatbots, Ihre Nachrichten zu beantworten, gibt es in der linken Menüleiste noch drei weitere Funktionen.

⚠ Sie können Ihre Fragen, die der Chatbot nicht richtig beantworten konnte, melden. Hierzu klicken Sie einfach auf das Icon, nachdem Sie eine unpassende Antwort erhalten haben und die App erstellt eine Excel-Datei, in der diese Fragen abgespeichert werden.

💾 Wenn Sie Ihren Chatverlauf exportieren möchten, wählen Sie die Diskette aus, um einen Speicherort zu wählen. Hier wird anschließend der aktuelle Chatverlauf gespeichert.

💡 Wechseln Sie manuell den Erscheinungsmodus der App zwischen Hell- und Dunkelmodus.

Helfen Sie uns den HNU-Bot weiterzuentwickeln und senden Sie uns Ihre unbeantworteten Fragen zu. Alle Fragen, auf die der Chatbot keine Antwort wusste oder die Sie gemeldet haben, werden in zwei Excel-Listen gespeichert. Sie finden Excel-Listen im Ordner HNU\_Chatbot unter den folgenden Namen:

unknown\_questions.xlsx  
wrong\_answer.xlsx

Senden Sie diese Dateien an [sonja.koepl@hnu.de](mailto:sonja.koepl@hnu.de)  
Vielen Dank!

## Anhang 7: install\_dependencies.py

```
1 # Zuerst die folgenden Bibliotheken installieren
2 # (möglicherweise müssen Sie die 3 von pip3 entfernen)
3 pip3 install pandas
4 pip3 install customtkinter
5 pip3 install nltk
6 pip3 install python-docx
7 pip3 install torch torchvision torchaudio
8
9 # Danach den folgenden Code ausführen:
10
11 import nltk
12 import ssl
13
14 try:
15     _create_unverified_https_context = ssl._create_unverified_context
16 except AttributeError:
17     pass
18 else:
19     ssl._create_default_https_context = _create_unverified_https_context
20
21 nltk.download('stopwords')
22 nltk.download('punkt')
```

## Anhang 8: intents\_umlaute.json

```
{
  "intents": [
    {
      "tag": "Begruessung",
      "patterns": [
        "Hi",
        "Hey",
        "Hallo",
        "Guten Tag",
        "Gute Nacht",
        "Gruess Gott",
        "Ich gruesse Sie",
        "Seien Sie gegruessst"
      ],
      "responses": [
        "Herzlich Willkommen",
        "Ich grüße Sie",
        "Seien Sie begrüßt"
      ]
    },
    {
      "tag": "Hilfe",
      "patterns": [
        "Kannst du mir helfen?",
        "Ich benoetige Hilfe",
        "Ein Thema verstehe ich nicht",
        "Bitte hilf mir."
      ],
      "responses": [
        "Gerne helfe ich Ihnen. Formulieren Sie bitte Fragen die relevante Schlagwörter enthalten."
      ]
    },
    {
      "tag": "Verabschiedung",
      "patterns": [
        "Tschuess",
        "Ciao",
        "Auf Wiedersehen"
      ],
      "responses": [
        "Bis zum nächsten Mal.",
        "Auf Wiedersehen",
        "Danke für deinen Besuch!"
      ]
    },
    {
      "tag": "Danke",
      "patterns": [
        "Danke",
        "Dankeschoen",
        "Vielen Dank",
        "Das war hilfreich, danke!",
        "Wow, danke"
      ],
      "responses": [
        "Nichts zu danken.",
        "Freut mich wenn ich helfen konnte."
      ]
    },
    {
      "tag": "Witz",
      "patterns": [
        "Witz",
        "Joke",

```

```

    "Lustig",
    "Humor"
  ],
  "responses": [
    "In einem Auto fahren ein Ingenieur, ein Physiker und ein Programmierer. Das Auto bleibt plötzlich mit einer Panne liegen. Da sagt der Ingenieur: Das liegt bestimmt an der Einspritzanlage, ich schau mir das gleich mal an. Der Physiker ist folgender Meinung: Ach was, da ist nur ein Kabel lose. Der Programmierer macht folgenden Vorschlag: Wir steigen einfach alle mal aus, dann wieder ein, und dann wird es schon wieder gehen.",
    "Wie viele Software-Entwickler braucht man, um eine Glühbirne zu wechseln? Keinen. Das ist ein Hardware-Problem.",
    "Kommt ein Suchmaschinenoptimierer in eine Bar, Kneipe, Spelunke, Wirtshaus, Gaststätte, Schänke, Bierstube.",
    "Was isst ein Informatiker abends beim Fernsehen? Mikrochips."
  ]
},
{
  "tag": "Blockchain",
  "patterns": [
    "Blockchain"
  ],
  "responses": [
    "Eine bestimmte Anzahl an Transaktionsdaten werden in einem Block zusammengefasst. Ein Block enthält zudem eine ID, einen Zeitstempel, den Hash-Wert des vorherigen Blocks. Dieser Block wird nun an alle Teilnehmer der Blockchain geschickt. Die Teilnehmer überprüfen nun die Richtigkeit der Transaktionsdaten und berechnen den Hash-Wert des aktuellen Blocks. Der Teilnehmer der diese Aufgaben als erstes erledigt hat, meldet die Ergebnisse an die anderen Teilnehmer. Die Teilnehmer speichern den geprüften Block mit samt des berechneten Hash-Wertes in der Blockchain. Die Blockchain entsteht nun aus der definierten Reihenfolge der Blöcke zueinander. Jeder Block verweist auf seinen Vorgänger, in dem der Hash-Wert des Vorgängers in dem darauffolgenden Block gespeichert wird. Wird ein Block in der Blockchain geändert, ändert sich auch der Hash-Wert des Blocks. Alle anderen folgenden Blöcke gelten dann als nicht mehr valide, da jeder Block den Hash-Wert des Vorgängers enthält. Ist ein Block in einer Kette geändert und die gesamte Blockchain neu signiert worden (Neuberechnung der Hash-Werte jedes Blocks), erhält man zwar eine valide Blockchain, die Hashwerte dieser Blockchain unterscheiden sich aber von den Hashwerten der anderen Kopien. Somit kann man durch einen Vergleich zwischen der geänderten Blockchain und den anderen dezentral gespeicherten Kopien die Informationsänderung leicht feststellen. -> Wahr ist ein einer Blockchain was die Mehrheit sagt."
  ]
},
{
  "tag": "dezentral",
  "patterns": [
    "dezentral"
  ],
  "responses": [
    "Eine dezentrale oder distribuierte Speicherung von Daten bedeutet, dass diese eben nicht nur an einer zentralen Stelle abgelegt, sondern an mehrere oder alle Teilnehmer eines Netzwerkes verteilt werden. Jeder Teilnehmer erhält ein identisches Abbild der Daten. | + hoher Schutz gegen Datenverlust ->

```

```

    fällt ein Teilnehmer aus, können Daten von anderen Teilnehmern wiederherge-
    stellt werden | + Keine Machtkonzentration an zentraler Stelle -> zentrale
    Stelle hat nicht mehr alleinige Macht und wird als Gatekeeper eliminiert"
  ]
},
{
  "tag": "zentral",
  "patterns": [
    "zentral"
  ],
  "responses": [
    "Vor- und Nachteil von zentraler Datensicherung: | + Eindeutiger und ein-
    heitlicher für alle sichtbarer Zugang -> ermöglicht schnellen Zugriff -> ef-
    fizient | - Ausfall zentraler Stelle -> Datenverlust (sofern kein sicheres
    Update vorhanden)"
  ]
},
{
  "tag": "Risiko USB-Sticks",
  "patterns": [
    "USB"
  ],
  "responses": [
    "Das Hauptrisiko besteht einerseits bei dem unerlaubten Transfer von Daten
    aus geschützten Bereichen heraus, andererseits bei dem unerlaubten Einschlep-
    pen von Daten, Programmen und besonders Schadcode. -> Gebrauch von USB-Sticks
    sollte durch Unternehmensleitlinie geregelt werden, z. B. Daten nur ver-
    schlüsselt auf USB-Stick speichern, Nutzung privater USB-Sticks verbieten,
    etc."
  ]
},
{
  "tag": "Sicherheit von Bluetooth",
  "patterns": [
    "Bluetooth"
  ],
  "responses": [
    "In der ersten Phase wird zwischen zwei kommunizierenden Geräten ein ge-
    meinsamer statischer Verbindungsschlüssel zur Verknüpfung der beiden Geräte
    vereinbart (Pairing). -> über asymmetrische Verschlüsselung (mit privatem und
    öffentlichen Schlüssel) wird ein statischer Schlüssel übertragen. Dieser sta-
    tische Schlüssel wird im Anschluss für die Datenverschlüsselung mit einer
    symmetrischen Stromchiffre verwendet. Nach diesem Schritt steht zwar dieser
    Schlüssel zur Verfügung, aber wie bei allen Public Key Anwendungen ohne zu-
    sätzliche Massnahmen wissen beide Seiten noch nicht verlässlich, mit wem sie
    eigentlich den Schlüssel ausgetauscht haben. In der zweiten Phase wird dann
    die eigentliche Authentifikation durchgeführt und die Verschlüsselung gestar-
    tet. Möglichkeiten zur Authentifikation sind z. B. just works, numeric compa-
    rison, passkey entry, out of band."
  ]
},
{
  "tag": "WLAN",
  "patterns": [
    "WLAN",
    "W-LAN",
    "drahtlos lokal",
    "Netzwerksicherheit"
  ],
  "responses": [
    "Sicheres WLAN Verschlüsselungsverfahren wählen (WPA2 mit AES) | Sicheres
    Passwort wählen (20-stelliges Passwort, welches aus Klein- und Grossbuchsta-
    ben, Zahlen und Sonderzeichen besteht) | Deaktivieren von WPS (Wi-Fi Protec-
    ted Setup) | Installieren von aktueller Router-Firmware | SSID (Service Set
    Identifier) Name in individuelle Netzwerknamen ändern | Zugriff nur für frei-
    gegebene MAC-Adressen aktivieren | WLAN-Netzwerke mittels Firewall von kabel-
    gebundenen LAN separieren | Einsetzen einer Verschlüsselung oberhalb der

```

```

    durch das WLAN realisierten Netzwerkebene (SSH) | PC-Systeme die an das Internet
    angeschlossen sind, sollten mit Viren-Scanner und Desktop-Firewall ausgestattet
    sein | Ausstrahlung außerhalb des erwünschten Bereiches abschirmen"
  ],
},
{
  "tag": "Advanced Encryption Standards",
  "patterns": [
    "Advanced Encryption Standards",
    "AES"
  ],
  "responses": [
    "Die Varianten der AES Verschlüsselung, AES-128, AES-192 und AES-256 beinhalten
    in ihrer Bezeichnung die Länge des Schlüssels in Bit. Durch die größeren
    Schlüssellängen des AES von 128, 192 oder 256 konnte die Sicherheit, im
    Vergleich zum DES mit einer Schlüssellänge von 56 Bit, stark gesteigert werden.
    Die sicherste AES Variante ist damit AES-256. Möchte man mit dem Computer
    einen Schlüssel herausfinden, dann bräuhete dieser heutzutage (Stand 2019)
    schon bei einer Schlüssellänge von 128 Bit länger, als das Universum alt ist."
  ]
},
{
  "tag": "Verschluesselungsstandard",
  "patterns": [
    "Verschluesselungsstandard",
    "Standards zur Verschluesselung"
  ],
  "responses": [
    "WEP (Wired Equivalent Privacy): Das ist eine uralte Verschlüsselung mit
    vielen Schwachstellen, heutzutage quasi so viel wert, wie gar keine
    Verschlüsselung zu haben. | WPA (Wi-Fi Protected Access) ist die Weiterentwicklung
    von WEP und kam 2003. Bietet zusätzliche Schutzverfahren. Hat aber noch
    nicht zwingend die bessere Verschlüsselung mit AES (Advanced Encryption
    Standard) eingebaut. | WPA2 (Wi-Fi Protected Access Version 2) kam 2006 und
    ist der aktuelle Sicherheitsstandard. WPA2 verwendet den modernen
    Verschlüsselungsalgorithmus AES (Advanced Encryption Standard). WPA2
    arbeitet dabei mit einer Schlüssellänge von 256 Bit, die im Moment als
    sicher bezeichnet werden kann. ! Nachteil: Sicherheit hängt von
    Implementierung und Komplexität des Passwortes ab. | WPA3 (Wi-Fi Protected
    Access Version 3) kam 2018 und bringt wesentliche Verbesserungen bei
    der Authentifizierung und Verschlüsselung mit sich. Für höchste
    Sicherheitsanforderungen, wie bei Behörden, Industriebetrieben,
    Militär oder Regierungen. Bietet höchsten Schutz, auch wenn
    Passwörter verwendet werden, die nicht den Richtlinien für sichere
    Passwörter verwendet werden."
  ]
},
{
  "tag": "Hybride Verschluesselung",
  "patterns": [
    "Hybride Verschluesselung",
    "Hybrid"
  ],
  "responses": [
    "Bei der hybriden Verschlüsselung werden die Daten selbst mit einem
    symmetrischen Verfahren verschlüsselt. Gleichzeitig wird der
    symmetrische Schlüssel mit dem öffentlichen Schlüssel der
    Gegenstelle verschlüsselt bzw. eingehüllt. Damit wird
    vermieden, dass große Datenmengen mit dem langsamen
    asymmetrischen Verfahren verschlüsselt, bzw. entschlüsselt
    werden müssen. Lediglich der Schlüssel des
    symmetrischen Verfahrens muss auf diese Weise verschlüsselt,
    bzw. entschlüsselt werden. Die Gegenstelle kann den
    symmetrischen Schlüssel

```

```

        anschließend mit dem eigenen privaten Schlüssel entschlüsseln und erhält so
        mit den symmetrischen Schlüssel, um die eigentliche Nachricht zu entschlüs-
        seln."
    ]
},
{
    "tag": "Fluechtige Schluessel",
    "patterns": [
        "Fluechtige Schluessel",
        "Kurzzeitschluessel",
        "Einmalschluessel",
        "Sitzungsschluessel"
    ],
    "responses": [
        "Flüchtige Schlüssel werden typischerweise nur einmal (d.h. z. B. für eine
        Sitzung oder für eine Datei) verwendet und sollen nach Gebrauch gelöscht wer-
        den. Sitzungsschlüssel dienen zur Absicherung von Massendaten. -> Zur siche-
        ren Verteilung von Sitzungsschlüsseln werden Master Schlüssel verwendet."
    ]
},
{
    "tag": "Statische Schluessel",
    "patterns": [
        "Statische Schluessel",
        "Langzeitschluessel",
        "Master Schluessel"
    ],
    "responses": [
        "Statische Schlüssel werden für den Zweck der Schlüsselverteilung verwen-
        det. Statische Schlüssel sollen nie zur Absicherung von Massendaten oder
        langfristig als Verschlüsselungsschlüssel normaler Datenbestände eingesetzt
        werden."
    ]
},
{
    "tag": "Authentisierte Verschlüsselung",
    "patterns": [
        "Authentisierte Verschlüsselung",
        "Authentisiert"
    ],
    "responses": [
        "Authentisierte Verschlüsselung ist die Kombination aus Verschlüsselung und
        Daten-Integrität in einem einzigen Verfahren. Verschlüsselungsverfahren hei-
        ßen authentisiert, wenn nicht nur die Vertraulichkeit, sondern auch die In-
        tegrität der zu verschlüsselnden Daten geschützt wird. -> Verschlüsselungs-
        verfahren wird mit MAC kombiniert."
    ]
},
{
    "tag": "Hashfunktion",
    "patterns": [
        "Hashfunktion",
        "Hashwert",
        "Klartext"
    ],
    "responses": [
        "Der wesentliche Unterschied zwischen einer kryptografischen Hashfunktion
        und einer Verschlüsselung besteht darin, dass eine Hashfunktion nicht inver-
        tierbar ist. Das heißt, es ist nicht möglich, den berechneten Hashwert wieder
        in den Ursprungswert zurückzurechnen, im genannten Beispiel in das Passwort.
        Genau das ist aber bei einem Verschlüsselungsalgorithmus erforderlich, damit

```

```

    der legitime Empfänger einer verschlüsselten Nachricht den Originaltext wieder sichtbar machen kann."
  ],
},
{
  "tag": "Secure Hash Algorithmus",
  "patterns": [
    "SHA",
    "Secure Hash Algorithmus"
  ],
  "responses": [
    "Der gebildete Hash-Wert ist immer 256 Bit (32 Byte, 64 hexadezimale Zeichen) lang. Somit gibt es 2256 mögliche Hash-Werte (Dabei gibt es nur 264 Sandkörner auf der Erde. Es gibt also so viele Hashwerte wie Sandkörner auf 2192 Erden). | 1. Jeder String wird auf 512 Bit erweitert und dann in Blöcke zu 32 Bit Wörtern aufgeteilt. | 2. Danach findet die Hashberechnung in 64 Durchläufen über sechs logische Funktionen statt. | 3. Jeder Durchlauf ermittelt acht Hashwerte für die Berechnung der nächsten Iteration. | 4. Zuletzt werden die Werte verkettet und bilden den Hashwert von SHA-256."
  ]
},
{
  "tag": "Kryptografische Hash-Funktionen",
  "patterns": [
    "Kryptografische Hash-Funktionen",
    "Kryptografische Hash Funktionen"
  ],
  "responses": [
    "Eindeutigkeit: eine identische Zeichenfolge muss zum selben Hashwert folgen | Irreversibilität: der Hashwert darf nicht in die ursprüngliche Zeichenfolge zurückberechnet werden können | Kollisionsresistenz: zwei unterschiedliche Zeichenfolgen dürfen nicht den gleichen Hashwert ergeben"
  ]
},
{
  "tag": "Digitale Signatur",
  "patterns": [
    "Digitale Signatur",
    "digital signatur",
    "signatur",
    "Digitale Unterschrift"
  ],
  "responses": [
    "Bei digitalen Signaturen mit Hilfe eines nur dem Absender bekannten Schlüssels erstellt und mit Hilfe eines öffentlichen Schlüssels überprüft. So ist sichergestellt, dass der Unterschreiber im Besitz des privaten Schlüssels ist, und es sich so mit hoher Wahrscheinlichkeit um den vorgegebenen Autor der Nachricht handelt. Digitale Signaturen arbeiten wie asymmetrische Verschlüsselungsverfahren, nur quasi umgekehrt. Bei digitalen Signaturen ist der Verschlüsselungsschlüssel privat, während der Entschlüsselungsschlüssel öffentlich ist. Signieren -> für Datei wird ein Hash-Wert berechnet -> Hash-Wert wird mit privatem Schlüssel verschlüsselt und an Datei angehängt -> Daten mit verschlüsseltem Hash-Wert übertragen Verifizieren -> Empfänger berechnet für Daten Hash-Wert -> Empfänger entschlüsselt den erhaltenen Hash-

```

```

    Wert mit öffentlichem Schlüssel -> Empfänger überprüft Übereinstimmung zwischen
    berechneten Hash-Wert und entschlüsseltem Hash-Wert -> Übereinstimmung:
    Signatur gültig."
  ]
},
{
  "tag": "Message Authentication Codes",
  "patterns": [
    "Message Authentication Codes",
    "MAC"
  ],
  "responses": [
    "Die Integrität einer zu übertragenden Nachricht lässt sich mittels MACs
    wie folgt prüfen: Zunächst vereinbaren Sender und Empfänger einen geheimen
    Schlüssel. Der Sender berechnet für diesen Schlüssel und seine Nachricht einen
    MAC und sendet dann die Nachricht sowie den MAC an den Empfänger. Dieser
    berechnet den MAC zu der empfangenen Nachricht mit dem Schlüssel und vergleicht
    den berechneten MAC mit dem empfangenen. Die Übereinstimmung beider Werte
    interpretiert der Empfänger als erfolgreichen Integritätstest: Die Nachricht
    wurde von einer Partei abgeschickt, die den geheimen Schlüssel kennt, und sie
    wurde während der Übertragung nicht verändert. MACs unterscheiden sich von
    digitalen Signaturen darin, dass die Überprüfung des MACs Kenntnis desselben
    geheimen Schlüssels erfordert, der zu seiner Berechnung genutzt wurde. Daher
    kann jeder, der einen MAC überprüfen kann, diesen auch berechnen; entsprechend
    ist er nicht in der Lage, gegenüber Dritten zu beweisen, von wem die Nachricht
    stammt."
  ]
},
{
  "tag": "Countermode",
  "patterns": [
    "Countermode"
  ],
  "responses": [
    "Countermode: jeder Block wird mit Hilfe eines eindeutigen zusätzlichen
    Initialisierungsvektors verschlüsselt. Um diesen eindeutigen Initialisierungsvektor
    zu generieren, wird eine initial gewählte Zufallszahl mit einem Zähler verknüpft,
    welcher für jeden weiteren Block um 1 hochgezählt wird. -> Sicherer als ECB,
    sollte hier verwendet werden."
  ]
},
{
  "tag": "Electronic Code Book Mode",
  "patterns": [
    "Electronic Code Book Mode"
  ],
  "responses": [
    "Betriebsmodus Electronic Code Book Mode: jeder Block wird unabhängig von
    den anderen Blöcken verschlüsselt / entschlüsselt. -> Unsicher: da partielle
    Informationen über den Klartext verrät. Zwei identische Chiffretextblöcke
    ergeben zwei identische Klartextblöcke."
  ]
},
{
  "tag": "Cipher Block Chaining Mode",
  "patterns": [
    "Cipher Block Chaining Mode",
    "CBC"
  ],
  "responses": [
    "Bei CBC werden die Klartextblöcke vor der eigentlichen Chiffrieroperation
    mit vorangegangenen Chiffretextblöcken bitweise XOR verknüpft, wobei für den
    ersten Klartextblock ein zufälliger Pseudo-Chiffretextblock verwendet wird,
    der auch als Initialisierungsvektor (IV) bezeichnet wird. Damit wird sichergestellt,
    dass zwei identische Klartextblöcke mit einer hohen Wahrscheinlichkeit nicht
    zu einem identischen Chiffretextblock führen werden. Der Initialisierungsvektor
    wird dem Chiffretext in der Regel offen vorangestellt und

```

```

        übermittelt; der Initialisierungsvektor ist also nicht vertraulich wie der
        verwendete Schlüssel."
    ],
},
{
    "tag": "Blockchiffrierverfahren",
    "patterns": [
        "Blockchiffrierverfahren"
    ],
    "responses": [
        "Im Gegensatz zu einer Stromchiffre kann eine Blockchiffre nur einen Block
        der gegebenen Länge verschlüsseln. Längere Texte werden auf ein Vielfaches
        der Blocklänge aufgefüllt und in Blöcke geteilt, und es wird ein Betriebsmo-
        dus gewählt, der festlegt, wie die Blockchiffre darauf anzuwenden ist. Jeder
        Block wird dann mit derselben Verschlüsselungsfunktion verschlüsselt. Für die
        Entschlüsselung werden die Geheimtextblöcke wiederum separat mit demselben
        Entschlüsselungsschlüssel entschlüsselt."
    ]
},
{
    "tag": "Stromchiffrierverfahren",
    "patterns": [
        "Stromchiffrierverfahren",
        "Stromverschlüsselung",
        "Strom"
    ],
    "responses": [
        "Bei einem Stromchiffrierverfahren wird in Abhängigkeit eines Schlüssels
        ein Schlüsselstrom erzeugt; die Chiffrierung der Eingabe erfolgt durch zeich-
        nerweise Verknüpfung der Eingabe mit einem Schlüsselstrom. Die zeichenweise
        Verknüpfung muss dabei einfach wieder rückgängig gemacht werden können, daher
        kommt typischerweise dafür eine bit- oder byte-weise XOR-Verknüpfung zum Ein-
        satz."
    ]
},
{
    "tag": "Asymmetrische Verschlüsselung",
    "patterns": [
        "Asymmetrische Verschlüsselung",
        "Asymmetrisch"
    ],
    "responses": [
        "Bei einem asymmetrischen Verschlüsselungsverfahren (auch Public-Key-Ver-
        fahren genannt), gibt es im Gegensatz zu einem Symmetrischen, nicht nur ein
        Schlüssel, sondern gleich zwei. Dieses sogenannte Schlüsselpaar setzt sich
        aus einem privaten Schlüssel (Privater Schlüssel -> Private Key) und einem
        öffentlichen Schlüssel (nicht geheim -> Public Key) zusammen. Mit dem priva-
        ten Schlüssel, werden Daten Entschlüsselt. Mit dem öffentlichen Schlüssel
        kann man Daten verschlüsseln. Der große Vorteil dieses Verfahrens im Ver-
        gleich zur symmetrischen Verschlüsselung ist in der einfachen Verteilung des
        öffentlichen Schlüssels begründet. Dieser kann wirklich für jeden Menschen
        frei zugänglich sein, ohne dass dadurch das Verfahren unsicher wird. Asymmet-
        rische Verfahren sind im Vergleich zu symmetrischen Verfahren sehr rechenauf-
        wändig. Um kurze Nachrichten zu verschlüsseln, benötigt der Computer viel
        Zeit. Asymmetrische Verfahren sind im Vergleich zu symmetrischen Verfahren
        langsam und komplex und deshalb sehr anfällig gegenüber Implementierungsfeh-

```

```

    lern. Komplizierte mathematische Verfahren, wie sie bei asymmetrischen Ver-
    fahren angewendet werden, bieten mehr Angriffsfläche und damit mehr Ansätze
    für die Kryptoanalyse."
  ]
},
{
  "tag": "Symmetrische Verschlüsselung",
  "patterns": [
    "Symmetrische Verschlüsselung",
    "Symmetrisch"
  ],
  "responses": [
    "Bei der symmetrischen Verschlüsselung wird ein und derselbe Schlüssel ver-
    wendet, um Botschaften zu verschlüsseln und auch wieder zu entschlüsseln. Der
    Schlüssel muss also sowohl dem Sender als auch dem Empfänger bekannt sein und
    zu diesem Zweck vorher ausgetauscht werden. Das ist gleichzeitig auch der
    Nachteil dieser Verschlüsselungsart, da durch den Austausch das Risiko be-
    steht, dass der Schlüssel beim Übermitteln unbefugten Personen bekannt werden
    könnte. Der wichtigste Nachteil symmetrischer Verschlüsselung gegenüber der
    asymmetrischen Verschlüsselung ist, dass der generierte Schlüssel sowohl für
    die Verschlüsselung als auch für die anschließende Entschlüsselung zuständig
    ist. Tauschen zwei Personen verschlüsselte Nachrichten untereinander aus, so
    müssen beide über denselben Schlüssel verfügen. Beim Austausch der Schlüssel
    muss demnach ein geschützter Kanal vorhanden sein. Da dies meist nicht der
    Fall ist, hat ein Feind die Chance, die geheimen Schlüssel bei der Übertra-
    gung abzufangen."
  ]
},
{
  "tag": "Known Plaintext",
  "patterns": [
    "Known Plaintext",
    "Known"
  ],
  "responses": [
    "Im Vergleich zum Chosen Plaintext Angriff kann der Angreifer beim Known
    Plaintext Angriff die zu verschlüsselten Texte nicht frei wählen."
  ]
},
{
  "tag": "Chosen Plaintext",
  "patterns": [
    "Chosen Plaintext",
    "Chosen",
    "Plaintext"
  ],
  "responses": [
    "Es handelt sich um einen Angriff gegen Verschlüsselungsverfahren. Bei ei-
    nem Chosen-Plaintext-Angriff (CPA) wird dem Angreifer unterstellt, dass er
    die zu verschlüsselnden Klartexte frei wählen kann und die dazu gehörenden
    Schlüsseltexte enthält. Dabei handelt es sich um einen aktiven Angriff. Der
    Angreifer versucht nun einen Algorithmus zu erstellen, mit dem er alle mit
    diesem Schlüssel verschlüsselten Chiffretextnachrichten entschlüsseln kann
    (ohne den geheimen Schlüssel tatsächlich zu kennen)."
  ]
},
{
  "tag": "Verschlüsselung",
  "patterns": [
    "Verschlüsselung",
    "Kryptografie"
  ],
  "responses": [
    "Daten in eine für Unbefugte nicht mehr lesbare Form verwandeln. Auf Daten
    kann erst nach der Entschlüsselung wieder lesbar zugegriffen werden. Allge-
    mein versteht man unter Verschlüsselung ein Verfahren, das eine lesbare In-

```

formation (Klartext) durch einen Schlüssel in einen Geheimtext (Chiffre) umwandelt, so dass der Klartext nur unter Verwendung des Schlüssels wieder lesbar wird. Mit der Verschlüsselung von Daten werden mehrere Ziele verfolgt. Im Wesentlichen sind dies folgende drei Ziele: Vertraulichkeit der Daten | Integrität der Daten | Verbindlichkeit und Authentizität der Quelle"

```
    ]
  },
  {
    "tag": "Anti-Virenprogramm",
    "patterns": [
      "Anti-Virenprogramm",
      "Anti Virenprogramm"
    ],
    "responses": [
      "Firewall ist die erste Verteidigungsmethode eines Computers. Sie ist eine sehr steife Software und schützt den Computer / Netzwerk gegen nicht authentifizierte Zugriffe von außen. Ein Virenprogramm säubert die Festplatte hinter der Firewall. Der Computer wird auf gescannt um infizierte Dateien zu erkennen und zu entfernen."
    ]
  },
  {
    "tag": "Application Level Gateway",
    "patterns": [
      "Application Level Gateway"
    ],
    "responses": [
      "Ein Application Level Gateway stellt die aufwändigste Form der Firewall dar und überprüfen die Kommunikation zwischen verschiedenen Netzen auf der Inhaltsebene. Der Schutz besteht im Großen und Ganzen darin, dass üblicherweise ein äußerer Anwender nicht unmittelbar mit einer Anwendung verbunden wird, sondern mit einem vertrauenswürdigen Proxy. Dieser nimmt die Anforderung des äußeren Anwenders auf, prüft dessen Berechtigung und veranlasst dann die gewünschte Funktionsausführung. Ein dabei entstehendes Ergebnis wird an den äußeren Anwender übergeben."
    ]
  },
  {
    "tag": "Paketfilter",
    "patterns": [
      "Paketfilter"
    ],
    "responses": [
      "Paketfilter treffen basierend auf einem Regelwerk die Entscheidung über zulässige Verbindungen auf Basis von Quell- und Absender IP-Adressen. Vorteile: bewährtes einfaches Konzept, leicht erweiterbar, gute Performance Nachteile: Aufwand steigt mit zunehmender Komplexität der Regelwerke, keine Prüfung des Paketinhaltes Ein statischer Paketfilter ist dadurch gekennzeichnet, dass er die Filterregeln der Reihe nach abarbeitet, bis anhand einer Regel das Paket entweder an das Zielsystem weitergeleitet oder verworfen wird. Es werden keine Statusinformationen über den aktuellen Datenfluss gehalten, stattdessen muss jedes Paket einzeln betrachtet werden, völlig unabhängig davon, welche Pakete bereits vorher verarbeitet wurden. Auch dynamische Paketfilter arbeiten die Filterregeln nacheinander ab und entscheiden dann, was mit den Paketen passiert. Dynamische Paketfilter erlauben aber im Gegensatz zu statischen Paketfiltern eine zustandsorientierte Filterung. Das heißt, es werden Statusinformationen über bestehende Verbindungen gehalten und fließen bei der Entscheidung, ob ein Paket verworfen oder weitergeleitet wird, ein."
    ]
  }
}
```

Dynamische Paketfilter führen zu diesem Zweck eine Statustabelle, die Informationen über aktuelle Verbindungen speichert."

```
    ],
  },
  {
    "tag": "URL-Filter",
    "patterns": [
      "URL-Filter",
      "URL Filter"
    ],
    "responses": [
      "Ein URL-Filter blockiert Verbindungen zu einer Webseite basierend auf der Internetadresse. Seiten die auf einer schwarzen Liste stehen (Blacklist), werden nicht weitergeleitet. Meistens kann auch über Regeln definiert werden, welche Seiten blockiert werden sollen."
    ]
  },
  {
    "tag": "TCP/IP",
    "patterns": [
      "TCP/IP",
      "Netzwerkprotokoll"
    ],
    "responses": [
      "TCP/IP ist die Grundlage des Internets. Der Name TCP/IP setzt sich aus den beiden für die Internetkommunikation entscheidenden Protokollen zusammen: dem Transmission Control Protocol (TCP) und dem Internet Protocol (IP). Die Protokolle des TCP/IP-Modells haben einen großen Vorteil: Sie funktionieren losgelöst von der Hardware und der zugrundeliegenden Software. Mithilfe der IP-Adresse und dem Internetprotokoll wird das Datenpaket an den richtigen Empfänger gesendet. TCP hingegen ist dafür verantwortlich, eine Verbindung zwischen den beiden beteiligten Geräten aufzubauen und für die Übertragung aufrechtzuerhalten. Sollte der Transport der Datenpakete gestört worden sein, sorgt das Protokoll für einen erneuten Übermittlungsversuch."
    ]
  },
  {
    "tag": "Port",
    "patterns": [
      "Port",
      "Port-Nummer",
      "Port-Nr",
      "Port Nr"
    ],
    "responses": [
      "Ein Port ist einen Kommunikationsendpunkt. Mit der IP Adresse wird der Rechner und mit dem Port der Dienst (Anwendung) angegeben. -> Mit Ports können Datenpakete im Internet stets der passenden Anwendung zugeordnet werden. Ihr Computer verfügt insgesamt über 65.536 Ports. Kommt ein Datenpaket, zum Beispiel eine Mail, auf Ihrem Computer an, so wird diese stets über den für Mails reservierten Port geleitet."
    ]
  },
  {
    "tag": "Private IP-Adresse",
    "patterns": [
      "Private IP-Adresse",
      "Private IP Adresse",
      "Interne IP-Adresse",
      "Interne IP Adresse"
    ],
    "responses": [
      "So wie die öffentliche IP-Adresse Ihres Netzwerks von Ihrem Internetdienstanbieter vergeben wird, gibt Ihr Router private (oder interne) IP-Adressen an jedes Netzwerkgerät in Ihrem Netzwerk aus. Dies bietet eine eindeutige Identifikation für Geräte in Ihrem Heimnetzwerk, z. B. Ihren Computer, Ihr Handy usw. Ähnlich wie bei öffentlichen IP-Adressen verfügt jedes Gerät in
```

```

        Ihrem Netzwerk über seine Netzwerkkonfigurationseinstellungen auf DHCP, so-
        dass es die eindeutige private IP-Adresse akzeptieren kann, die Ihr Router
        ausgibt. Diese privaten IP-Adressen verlassen niemals Ihr Netzwerk, so wie
        Ihre öffentliche IP-Adresse niemals in Ihrem Netzwerk verwendet wird. Der
        Router steuert den gesamten Netzwerkverkehr sowohl innerhalb als auch außer-
        halb Ihres Heimnetzwerks zum Internet. Es ist Aufgabe des Routers, sicherzu-
        stellen, dass Daten zu und von den richtigen Stellen fließen."
    ]
},
{
    "tag": "Oeffentliche IP-Adresse",
    "patterns": [
        "Oeffentliche IP-Adresse",
        "Oeffentliche IP Adresse",
        "Externe IP-Adresse",
        "Externe IP Adresse"
    ],
    "responses": [
        "Eine öffentliche (oder externe) IP-Adresse ist diejenige, die Ihr ISP (In-
        ternet Service Provider) bereitstellt, um Ihr Heimnetzwerk nach außen zu
        identifizieren. Es ist eine IP-Adresse, die im gesamten Internet eindeutig
        ist. Abhängig von Ihrem Dienst haben Sie möglicherweise eine IP-Adresse, die
        sich nie ändert (eine feste oder statische IP-Adresse). Die meisten ISPs
        stellen jedoch eine IP-Adresse bereit, die sich von Zeit zu Zeit ändern kann
        (eine dynamische IP-Adresse). Für die überwiegende Mehrheit der Benutzer ist
        eine dynamische IP-Adresse in Ordnung. Wenn Sie beim Einrichten Ihres Routers
        von Ihrem Internetdienstanbieter eine statische IP-Adresse erhalten haben,
        geben Sie diese in die Einstellungen Ihres Routers ein. Für eine dynamische
        IP-Adresse geben Sie DHCP in den Netzwerkeinstellungen Ihres Routers an. DHCP
        ist das Dynamic Host Control Protocol. Es weist Ihren Router an, jede öffent-
        liche IP-Adresse zu akzeptieren, die Ihr ISP ausgibt."
    ]
},
{
    "tag": "IPv6",
    "patterns": [
        "IPv6"
    ],
    "responses": [
        "Die Adressgröße wurde von 32 Bits bei IPv4 auf 128 Bits bei IPv6 erhöht.
        IPv6 hat eine theoretische Obergrenze von  $3,4 \times 10^{38}$  Adressen. Das sind mehr
        als 340 Sextillionen Adressen, was angeblich genug ist, um jedem einzelnen
        Atom auf der Erdoberfläche eine Adresse zuzuweisen."
    ]
},
{
    "tag": "IPv4",
    "patterns": [
        "IPv4"
    ],
    "responses": [
        "IPv4 ist eine Adressierung mit 4 binären Adressblöcken von maximal 28 (=
        256) Adressen. Oder IPv4-Adressen sind 32 Bits (4 Bytes) lang. Daher können
         $2^{32} = 4.294.967.296$  Adressen oder  $256 \times 256 \times 256 \times 256 = 4.294.967.296$  Adressen
        dargestellt werden."
    ]
},
{
    "tag": "IP-Adresse",
    "patterns": [
        "IP-Adresse",
        "IP"
    ],
    "responses": [
        "Jedes Gerät innerhalb eines Datennetzwerkes braucht eine Adresse, damit es
        eindeutig identifiziert werden kann. Das gilt sowohl für das kleine Heimnetz-
        werk als auch für das Internet. Nur so ist gesichert, dass der Datenstrom

```

beim richtigen Gerät ankommen. Die IP-Adresse ist vergleichbar mit einer Postadresse oder Telefonnummer im analogen Leben. IP-Adresse unter Windows herausfinden: Windows-Taste und R-Taste drücken -> cmd eintippen um Eingabeaufforderung zu öffnen -> Befehl ipconfig eingeben. Oder: rechte Maustaste auf Startmenü und Netzwerkverbindungen auswählen. Menüpunkt Status anklicken um Netzwerkeigenschaften anzuzeigen. Für die verschiedenen Anschlüsse wie Ethernet und WLAN finden Sie die jeweilige IP-Adresse."

```

    ],
    {
      "tag": "OSI-Modell",
      "patterns": [
        "OSI-Modell",
        "OSI"
      ],
      "responses": [
        "Das OSI-Modell beschreibt die Voraussetzungen, die gegeben sein müssen, damit verschiedene Netzwerkkomponenten miteinander kommunizieren können. Die sieben Schichten des OSI-Modells: 7. Anwendungsschicht (Application layer), 6. Darstellungsschicht (Presentation layer), 5. Kommunikationsschicht (Session layer), 4. Transportschicht (Transport layer), 3. Vermittlungsschicht (Network layer), 2. Sicherungsschicht (Link layer), 1. Bitübertragungsschicht (Physical layer)"
      ]
    },
    {
      "tag": "Firewall",
      "patterns": [
        "Firewall"
      ],
      "responses": [
        "Eine Firewall ist eine Anordnung von Hard- und Softwarekomponenten, welche zwischen Netzsegmenten mit in der Regel unterschiedlichen Schutzbedarf geschaltet wird, um den Datenfluss zwischen den Segmenten zu kontrollieren. Im Prinzip haben die Firewall und die für den Privatgebrauch abgespeckte Version die Personal Firewall nicht mehr viel gemeinsam. Denn während bei der normalen Firewall viele Rechner durch einzelne ausgewählte Rechner geschützt werden, versucht sich der PC bei der Personal Firewall selbst zu schützen. Wie es der Name schon sagt, läuft die Personal Firewall auf dem PC selbst und schützt nur den Rechner, kein Netzwerk. Funktionalitäten von Personal Firewalls: Paket Filter: Dieser kontrolliert, ob die Daten der an- und ausgehenden Pakete auch dem vom Benutzer festgelegten Regeln entsprechen. - Sandboxing: Dabei werden einzelne Programme in eine eingeschränkte Umgebung gesperrt. In diesem implementierten Schutzbereich werden Programme ausgeführt. Falls es sich dabei um Schadsoftware handeln sollte, kann sie aber keinen Schaden anrichten, da durch die Isolation der Rest des Systems davon nicht beeinflusst wird."
      ]
    },
    {
      "tag": "Trojaner",
      "patterns": [
        "Trojaner"
      ],
      "responses": [
        "Folgendes kann ein Trojaner anrichten: Backdoor: ermöglicht Cyberkriminellen die Remote-Steuerung des infizierten Computers. | Keylogger: zeichnet Tastatureingaben auf und übermittelt sie dem Hacker | Trojan-Downloader: laden neue Malware auf den infizierten PC | Trojan-Ransom: modifiziert Dateien auf dem infizierten PC, so dass man nicht mehr zugreifen kann | Trojan-SMS: stehlen ihr Geld, indem SMS von ihrem Mobiltelefon an kostenpflichtige Telefonnummern gesendet werden; Vor Trojaner schützen: Regelmäßige Diagnose-Scans
      ]
    }
  ],
  {
    "tag": "Trojaner",
    "patterns": [
      "Trojaner"
    ],
    "responses": [
      "Folgendes kann ein Trojaner anrichten: Backdoor: ermöglicht Cyberkriminellen die Remote-Steuerung des infizierten Computers. | Keylogger: zeichnet Tastatureingaben auf und übermittelt sie dem Hacker | Trojan-Downloader: laden neue Malware auf den infizierten PC | Trojan-Ransom: modifiziert Dateien auf dem infizierten PC, so dass man nicht mehr zugreifen kann | Trojan-SMS: stehlen ihr Geld, indem SMS von ihrem Mobiltelefon an kostenpflichtige Telefonnummern gesendet werden; Vor Trojaner schützen: Regelmäßige Diagnose-Scans
    ]
  }
}

```

```

    | Keine Dateianhänge in Mails von unbekanntem Absendern öffnen | Nicht auf
    fragwürdigen Internetseiten surfen"
  ],
},
{
  "tag": "Social Engineering",
  "patterns": [
    "Social Engineering",
    "Social-Engineering"
  ],
  "responses": [
    "Absender Adresse prüfen | Persönlich nachfragen | Rhetorik | Auf Rechtschreibfehler achten | Nicht direkt auf Links klicken | Über den Link hovern"
  ]
},
{
  "tag": "Brute Force Attacke",
  "patterns": [
    "Brute Force Attacke",
    "Brute-Force-Attack"
  ],
  "responses": [
    "Passwörter nie im Klartext hinterlegen, sondern als Hashwerte | Erhöhen der Passwortlänge | Erhöhen der Passwortkomplexität | Begrenzung der Anmeldeversuche | Einfügen eines Captchas | Verwenden von Multi-Faktor-Authentifizierung"
  ]
},
{
  "tag": "Buffer Overflows",
  "patterns": [
    "Buffer",
    "Overflows",
    "Buffer Overflows"
  ],
  "responses": [
    "Eine Buffer-Overflow-Verwundbarkeit liegt vor, wenn durch einen Angreifer ein vermeintlich beschränkter Puffer im Speicher über seine Grenzen hinaus kontrolliert überschrieben werden kann. Ursache ist, dass die maximalen Größen bei der Eingabe zwar zur Laufzeit nicht geprüft wird und somit überschrieben werden kann. In einem Namensfeld wird beispielsweise eine Eingabe von 20 Zeichen erwartet. Es ist aber möglich, z. B. von Hand, per Skript oder per übertragenem Datenpaket, mehr Zeichen einzugeben."
  ]
},
{
  "tag": "SQL Injection",
  "patterns": [
    "SQL",
    "SQL-Injection",
    "SQL-Datenbank"
  ],
  "responses": [
    "Der Angreifer versucht dabei, über die Anwendung, die den Zugriff auf die Datenbank bereitstellt, eigene Datenbankbefehle einzuschleusen. Sein Ziel ist es, Daten auszuspähen, in seinem Sinne zu verändern, die Kontrolle über den Server zu erhalten oder einfach größtmöglichen Schaden anzurichten. In eine Eingabemaske wird statt des Kennwortes ein SQL-Befehl eingetragen. Wird diese

```

```

    Eingabe nicht gefiltert, ist es möglich die Abfragen der Datenbank zu manipu-
    lieren."
  ]
},
{
  "tag": "Schwachstellen",
  "patterns": [
    "Schwachstellen",
    "Sicherheitsluecke"
  ],
  "responses": [
    "Fest programmierte Zugangskennungen und Passwörter | Standardpasswörter,
    welche nicht geändert wurden | Fest programmierte symmetrische Verschlüsse-
    lungsschlüssel | Netzwerkprotokolle ohne Verschlüsselung und Authentifizie-
    rung | Vorhersagbare Zufallsgeneratoren, die z. B. nur zwischen 2 und 32.000
    verschiedene Werte erzeugen | Anfälligkeit gegen Buffer Overflows | Jegliche
    weitere Form fehlender Eingabeüberprüfung als Lücke für z. B. SQL-Injektion-
    Angriffe | Die trügerische Hoffnung, dass ein kompliziertes oder geheimes
    Verfahren von Angreifern schon nicht verstanden werde: Security by Obscurity"
  ]
},
{
  "tag": "Return on Security Investment",
  "patterns": [
    "Return on Security Investment",
    "ROSI",
    "Security Investment"
  ],
  "responses": [
    "Der Kern des ROSI-Berechnungsmodells ist einfach: Jede wirksame Sicher-
    heitsinvestition (SI) reduziert entweder die Eintrittswahrscheinlichkeit (P)
    oder die Höhe eines bestimmten Schadens (L). Damit sinkt der erwartende Scha-
    den, die Annual Loss Expectancy (ALE). Der Gewinn dieser Sicherheitsmaßnahme
    ergibt sich damit aus der Verringerung Erwartungswertes minus den Gesamtkos-
    ten der Maßnahme (TCOSI) Die Kennzahl ROSI ist der Quotient aus Gewinn und
    Kosten. ROSI ist dann positiv, wenn die vermuteten Einsparungen durch das
    Vermeiden von Schäden größer ist als die Investition in die Sicherheitsmaß-
    nahme -> investieren"
  ]
},
{
  "tag": "Total Cost of Ownership",
  "patterns": [
    "Total Cost of Ownership",
    "TCO",
    "Security Investment"
  ],
  "responses": [
    "TCO beschreibt die Gesamtkosten einer Sicherheitsinvestition.  $TCO = (Kon-$ 
    zeptionskosten + Investitionskosten) / Betriebszeitraum + Betriebskosten"
  ]
},
{
  "tag": "Gesamtkosten",
  "patterns": [
    "Gesamtkosten",
    "Einzelkosten"
  ],
  "responses": [
    "Umsatzeinbußen und Werteverlust: unmittelbar durch Ausfallzeiten und den
    Verlust von Kunden oder Aufträgen, z. B.: Image Schaden. Werteverlust und Um-
    satzrückgang sind abhängig von der Dauer des Sicherheitsvorfalls. | Wieder-
    herstellungskosten: setzen sich aus externen Kosten für Unterstützungslei-
    stungen und internen Aufwand zusammen, z. B.: Ersetzung oder Wiederanlauf der
    betroffenen Systeme, Wiedereinspielen von Images oder Datenbackups, Rettung

```

```

    teildeser Datenträger. | Schadensersatzleistungen: z. B.: Vertragsstrafen
    bei Lieferverzögerungen, Kosten für Rückrufe, Haftung für Schäden Dritter"
  ],
},
{
  "tag": "Annual Loss Expectancy",
  "patterns": [
    "Annual Loss Expectancy",
    "ALE",
    "Jaehrliche Verluste"
  ],
  "responses": [
    "ALE beschreibt den jährlichen zu erwarteten Verlust durch Angriffe auf Da-
    ten und Informationssysteme. Der erwartete Gesamtverlust ergibt sich als
    Summe der Erwartungswerte aller betrachteten Einzelrisiken ALE errechnet sich
    aus der finanziellen Höhe (Loss L) und der Eintrittswahrscheinlichkeit (Pro-
    bability P) eines Schadens. Optimierung ALE: Entweder wird durch geeignete
    Gegenmaßnahmen die Eintrittswahrscheinlichkeit des Schadens verringert, z. B.
    durch einen Viren-Scanner auf einem E-Mail-Server oder den Betrieb einer
    Firewall. | Oder es werden Maßnahmen ergriffen, die die Auswirkung des Scha-
    dens begrenzen, wie z. B. die konsequente Umsetzung des need-to-know-Prinzips
    bei der Berechtigungsvergabe oder ausgearbeitete und trainierte Notfallpläne,
    betriebsbereite Ersatzsysteme und ausgefeilte Wiederanlaufprozesse."
  ]
},
{
  "tag": "Wirtschaftlich sinnvoll",
  "patterns": [
    "wirtschaftlich",
    "Aus wirtschaftlicher Sicht sinnvoll"
  ],
  "responses": [
    "Den Kosten für eine Sicherheitsmaßnahme sollte, damit die Investition
    wirtschaftlich ist, eine (nachhaltige) Risikoreduktion gegenüberstehen. Ziel
    dabei ist, eine hinreichende Risikoreduktion mit moderaten Kosten zu errei-
    chen."
  ]
},
{
  "tag": "Technische Massnahmen",
  "patterns": [
    "Technische Massnahmen"
  ],
  "responses": [
    "Flächendeckender Einsatz aktueller Viren-Programme | Verwendung einer
    Firewall | Einschränkung von Datenzugriffsmöglichkeiten auf das erforderliche
    Mindestmaß | Einsatz einer angemessenen Infrastruktur mit Komponenten für Ba-
    ckup und Archivierung | Verschlüsselung von Informationen | Regelmäßige Über-
    prüfung und Installation von Sicherheitsupdates | Testen von Softwareänderun-
    gen"
  ]
},
{
  "tag": "Organisatorische Massnahmen",
  "patterns": [
    "Organisatorische Massnahmen"
  ],
  "responses": [
    "Zuordnung von Rollen und Profilen für Systembenutzer | Einschränkung von
    Administratorrechten auf das erforderliche Mindestmaß | Durchführung von re-
    gelmäßigen Datensicherungen (Backup) | Erstellung und Testen von Notfallplä-
    nen | Bekanntgabe der Notfallpläne an jeden Mitarbeiter | Recherchieren der
    Sicherheitseigenschaften verwendeter Software | Erfassung des gesamten Be-
    standes an Hard- und Software | Besondere Vorsicht beim Umgang mit Web-Brow-

```

```

    sern und E-Mail anhängen | Grundsätzliche Beachtung der Sicherheitsrichtli-
    nien und Sicherheitsanforderungen | Am Arbeitsplatz sollen sensitive Informa-
    tionen nicht frei zugänglich sein"
  ]
},
{
  "tag": "Sicherheitsziele",
  "patterns": [
    "Sicherheitsziele",
    "Schutzziele"
  ],
  "responses": [
    "Vertraulichkeit (Privacy): Daten dürfen nur befugten Personen zugänglich
    gemacht werden, auch bei der Übertragung | Integrität (Integrity): Daten dür-
    fen nicht unerkannt bzw. unbemerkt geändert werden | Verfügbarkeit (Availabi-
    lity): Daten dürfen nicht unbeabsichtigt / unbefugt vernichtet werden und
    müssen ggf. rasch wiederherstellbar sein | Authentizität (Authenticity): In-
    formationen müssen jederzeit zu ihrem Ursprung zugeordnet werden"
  ]
},
{
  "tag": "Informationssicherheit",
  "patterns": [
    "Informationssicherheit",
    "IT-Sicherheit"
  ],
  "responses": [
    "Informationssicherheit dient dem Schutz vor Gefahren bzw. Bedrohungen, der
    Vermeidung von wirtschaftlichen Schäden und der Minimierung von Risiken. Nach
    der betriebswirtschaftlich getriebenen Risikoreduktion gibt es eine zweite
    wichtige Motivation und Zielsetzung bei der Einführung von Maßnahmen der In-
    formationssicherheit in Unternehmen und Behörden: gesetzliche Regelungen und
    weitere rechtliche Anforderungen mit Bezug zur Informationssicherheit. Orga-
    nisationen unterfallen dem geltendem Recht und müssen somit Ressourcen inves-
    tieren, um ihre Abläufe regelkonform zu gestalten. Verstöße gegen gesetzliche
    Bestimmungen sind mit Sanktionen bewehrt (i.d.R. Bußgelder)."
  ]
},
{
  "tag": "Versaeumnisse",
  "patterns": [
    "Versaeumnisse"
  ],
  "responses": [
    "Unzureichende Informationssicherheitsstrategie | Schlechte Konfiguration
    von IT-Geräten | Unsichere Vernetzung und Internet-Anbindung | Nichtbeachtung
    von Sicherheitserfordernissen | Schlechte Wartung von IT-Systemen | Sorgloser
    Umgang mit Passwörtern | Nichtnutzung von vorhandenen Sicherheitsmaßnahmen |
    Mangelnder Schutz vor Einbrechern und Elementarschäden"
  ]
},
{
  "tag": "Assets",
  "patterns": [
    "Assets"
  ],
  "responses": [
    "Informationen: z. B. Datenbankdateien, Ordner, Verträge, Vereinbarungen,
    Systemdokumentationen, Handbücher, Forschungsdaten, Schulungsmaterial, Konti-
    nuitätspläne, Personalakten, Patente | Software: z. B. Anwendungen, System-
    software, Entwicklungsanwendungen | Physische Assets: z. B. Computer, Server,
    Router, Firewalls, Kommunikationsgeräte, Wechseldatenträger | Services: z. B.

```

```

    Klimaanlage, Stromanlagen, Wasseranlagen, Telekommunikationsanlagen | Personal: Qualifikation, Fähigkeiten, Fertigkeiten, Erfahrungen | Immaterielle Assets: Ruf und Image des Unternehmens"
  ]
},
{
  "tag": "Motive Datenangriff",
  "patterns": [
    "Motive Datenangriff",
    "Motive"
  ],
  "responses": [
    "Persönliche Bereicherung (Kriminelle) | Wirtschaftsspionage, staatliche Spionage (Firmen, Geheimdienste) | Wirtschaftssabotage, staatliche Sabotage (Cyber War) | Neugierige: auf Daten oder auf den Angriff selbst (irgendjemand) | Jugendlicher Übermut (Script-Kiddies) | Böswilligkeit (z. B. Ex-Freund, entlassender Mitarbeiter) | Gerechtigkeitssinn (Robin Hood) | Auf Risiken hinweisen (z. B. Chaos Computer-Club)"
  ]
},
{
  "tag": "call by reference",
  "patterns": [
    "call by reference"
  ],
  "responses": [
    "Call by reference gilt für Variablen die von einem nicht primitiven Datentyp sind und einer Funktion übergeben werden. In diesem Fall wird die Referenz (Adresse) einer Variablen und nicht ihr Wert an die Funktion übergeben. Alle Änderungen und Aktualisierungen am Parameter verändern auch die ursprüngliche Referenz-Variable."
  ]
},
{
  "tag": "call by value",
  "patterns": [
    "call by value"
  ],
  "responses": [
    "Call by value gilt für Variablen die von einem primitiven Datentyp sind und einer Funktion übergeben werden. Wenn eine solche Variable als Parameter an eine Funktion übergeben wird, wird nur ihr Wert der Funktion übergeben. Änderungen und Aktualisierungen an dem übergebenen Wert verändern nicht die ursprüngliche Variable."
  ]
},
{
  "tag": "Aufbau Funktionen",
  "patterns": [
    "Aufbau Funktionen"
  ],
  "responses": [
    "Funktionsdeklaration steht vor dem eigentlichen Hauptprogramm | Funktionsdeklaration beginnt mit dem Schlüsselwort def, gefolgt von dem Funktionsnamen und den Übergabeparametern in runden Klammern. Parameter dienen als Ablageort für Werte. Innerhalb des Funktionskörpers verhalten sie sich wie lokale Variablen. | In dem Anweisungsblock wird mit Hilfe der Übergabeparameter ein Rückgabeparameter berechnet. Parameter / Variablen sind nur innerhalb des Anweisungsblocks gültig. Ausserhalb der Funktion kann nicht darauf zugegriffen

```

```

        werden. | Funktion wird mit dem Schlüsselwort return beendet und gibt gleich-
        zeitig den berechneten Wert als Rückgabeparameter zurück. Funktion kann genau
        einen Wert an die aufrufende Instanz zurückgeben."
    ]
},
{
    "tag": "Doppelter Code",
    "patterns": [
        "Doppelter Code"
    ],
    "responses": [
        "Durch doppelten Code ist es sehr wahrscheinlich, dass Fehler nur an einer
        Stelle ausgebessert werden, Änderungen nur an einer Stelle durchgeführt wer-
        den usw. Doppelter / Mehrfacher Code kann durch die Verwendung von Funktionen
        vermieden werden. Code, der potenziell doppelt ist, wird in eine Funktion
        ausgelagert, die genau diese Aufgabe übernimmt."
    ]
},
{
    "tag": "Collection-Datentypen",
    "patterns": [
        "Collection-Datentypen",
        "Collection Datentypen"
    ],
    "responses": [
        "Listen: Sammlung, die geordnet und veränderbar ist. Vom Wert her doppelte
        Elemente erlaubt. | Tuple: Sammlung, die geordnet und unveränderlich ist. Vom
        Wert her doppelte Elemente erlaubt. | Set: Sammlung, die ungeordnet und nicht
        indiziert ist. Vom Wert her keine doppelten Elemente erlaubt. | Dictionary:
        Sammlung, die indiziert und veränderbar ist. Vom Wert her keine doppelten
        Elemente erlaubt."
    ]
},
{
    "tag": "Array",
    "patterns": [
        "Array"
    ],
    "responses": [
        "Ein Array ist eine Datenstruktur, die mehrere Werte unter einem gemeinsa-
        men Namen speichert."
    ]
},
{
    "tag": "Anweisungsblock",
    "patterns": [
        "Anweisungsblock"
    ],
    "responses": [
        "In dem Anweisungsblock einer Schleife muss mindestens eine Anweisung ste-
        hen. Diese Anweisung kann auch die Leeranweisung sein. Python verlässt sich
        auf Einrückungen durch Leerzeichen um die Zugehörigkeit zum Anweisungsblock
        zu realisieren."
    ]
},
{
    "tag": "Verzweigung",
    "patterns": [
        "Verzweigung",
        "Auswahl"
    ],
    "responses": [
        "Bedingte Anweisungen: beim Erfüllen einer Bedingung wird ein Anweisungs-
        block ausgeführt. Bei Nichterfüllen wird nichts getan, oder ein anderer An-
        weisungsblock ausgeführt. Mit if leiten Sie eine Wenn-Dann-Bedingung ein (if
        = wenn). Dahinter folgt, in Klammern stehend, die Formulierung der Bedingung.
        Um solche Bedingungen zu formulieren, braucht man Vergleichsoperatoren und in

```

den meisten Fällen auch Variablen. Die Bedingung wird mit einem Doppelpunkt beendet. Für Fälle, in denen die Bedingung nicht erfüllt ist, können Sie einen andernfalls-Zweig definieren. Dies geschieht durch else (else = sonst). Der Else-Zweig ist nicht zwingend erforderlich. Beachte: Python verlässt sich auf Einrückungen durch Leerzeichen um die Zugehörigkeit zu Anweisungsblöcken zu realisieren."

```

    ]
  },
  {
    "tag": "Anweisung",
    "patterns": [
      "Anweisung"
    ],
    "responses": [
      "Eine Anweisung ist ein zentrales Element der Programmierung. Programme setzen sich hauptsächlich aus einer oder mehreren Anweisungen zusammen. Eine Anweisung ist eine formulierte einzelne Vorschrift, die im Rahmen der Abarbeitung eines Programmes ausgeführt wird. Anweisungen sind üblicherweise Zuweisungen, Kontrollanweisungen (Schleifen, bedingte Anweisungen) und Funktionsaufrufe."
    ]
  },
  {
    "tag": "Kommentar",
    "patterns": [
      "Kommentar"
    ],
    "responses": [
      "Eine wichtige und grundlegende Sache bei der Programmierung sind die Kommentare. Wird ein Programm komplexer, verliert man schnell die Übersicht. Schaut man sein Programm nach ein paar Monaten nochmals an, weiß man auf Anhieb gar nicht mehr, was es eigentlich macht. Kommentare sollen helfen, komplizierte Programmabschnitte kurz und prägnant zu erklären. Kommentare werden vom Interpreter komplett ignoriert, da sie nur für den Menschen eine Funktion haben. In den Kommentaren gelten also keinerlei Schreibrichtlinien."
    ]
  },
  {
    "tag": "Fussgesteuerte Schleife",
    "patterns": [
      "Fussgesteuerte Schleife"
    ],
    "responses": [
      "Bei der fussgesteuerten Schleife steht die Bedingung am Ende der Schleife -> Daher wird die Schleife immer einmal durchlaufen, auch wenn die Bedingung der Schleife nicht erfüllt ist."
    ]
  },
  {
    "tag": "Kopfgesteuerte Schleife",
    "patterns": [
      "Kopfgesteuerte Schleife"
    ],
    "responses": [
      "Bei der kopfgesteuerten Schleife steht die Bedingung am Anfang der Schleife -> Wenn Bedingung nicht erfüllt ist, wird die Schleife gar nicht durchlaufen."
    ]
  },
  {
    "tag": "Schleife",
    "patterns": [
      "Schleife"
    ],
    "responses": [
      "Schleifen sind Kontrollstrukturen zur wiederholten Ausführung von Anweisungen. Anweisungen werden solange wiederholt, bis eine bestimmte Bedingung

```

nicht mehr erfüllt ist. Alle Schleifen müssen eine solche Bedingung enthalten. Schleifen dienen somit zur kompakteren Darstellung umfangreicher Aufgaben. WHILE-Schleife | FOR-Schleife (in Python nur zur Iteration über ein Sequenz von Objekten. In anderen Programmiersprachen ähnlich verwendet wie WHILE-Schleife.)"

```

    ]
  },
  {
    "tag": "Index erste Element Liste",
    "patterns": [
      "Index erste Element Liste",
      "Index"
    ],
    "responses": [
      "Auf das erste Element einer Liste kann mit dem Index 0 zugegriffen werden."
    ]
  },
  {
    "tag": "continue",
    "patterns": [
      "continue"
    ],
    "responses": [
      "Continue: beendet einen Schleifendurchlauf und setzt am Beginn der Schleife mit dem nächsten Durchlauf der Schleife fort."
    ]
  },
  {
    "tag": "break",
    "patterns": [
      "break"
    ],
    "responses": [
      "Break: beendet die Schleife und setzt mit dem Programmcode nach der Schleife fort."
    ]
  },
  {
    "tag": "Vergleichsoperatoren",
    "patterns": [
      "Vergleichsoperatoren"
    ],
    "responses": [
      "Operator == : Gleich | Operator != : Ungleich | Operator < : Kleiner | Operator <= : Kleiner gleich | Operator > : Größer | Operator >= : Größer gleich"
    ]
  },
  {
    "tag": "Zuweisungsoperatoren",
    "patterns": [
      "Zuweisungsoperatoren"
    ],
    "responses": [
      "Operator = : Zuweisung | Operator += : Additionszuweisung | Operator -= : Subtraktionszuweisung | Operator *= : Produktzuweisung | Operator /= : Divisionszuweisung | Operator **= : Exponentielle Zuweisung | Operator %= : Modulo-Zuweisung | Operator //= : Ganzzahlige Divisionszuweisung"
    ]
  },
  {
    "tag": "Logische Operatoren",
    "patterns": [
      "Logische Operatoren"
    ],
    "responses": [

```

```

    "Logikoperatoren dienen zum Vergleich von Wahrheitswerten - es werden logische Verknüpfungen formuliert. | Operator and: Verknüpfen von zwei oder mehreren Bedingungen durch und. Alle Bedingungen müssen erfüllt sein, damit die gesamte Bedingung erfüllt ist. | Operator or: Verknüpfen von zwei oder mehreren Bedingungen inklusiv durch oder. Es genügt wenn eine der Bedingungen erfüllt ist, damit die gesamte Bedingung erfüllt ist. | Operator not: Prüft ob ein Ausdruck unwahr ist."
  ],
},
{
  "tag": "Arithmetische Operatoren",
  "patterns": [
    "Arithmetische Operatoren",
    "Arithmetisch"
  ],
  "responses": [
    "Operator + : Positives Vorzeichen, Summe | Operator - : Negatives Vorzeichen, Differenz | Operator * : Produkt | Operator / : Quotient | Operator % : Divisionsrest (Modulo) | Operator ** : Exponent | Operator // : Ganzzahliges Divisionsergebnis"
  ]
},
{
  "tag": "Operatorengruppen",
  "patterns": [
    "Operatorengruppen"
  ],
  "responses": [
    "Zuweisungsoperator, arithmetische Operatoren, Vergleichsoperatoren, logische Operatoren, bitweise Operatoren"
  ]
},
{
  "tag": "Operatoren",
  "patterns": [
    "Operatoren",
    "Operanden",
    "Operationen"
  ],
  "responses": [
    "Operanden sind Variablen oder Konstanten | Operatoren verknüpfen Operanden | Operation ist die Anwendung eines Operators auf mindestens einen Operanden | jede Operation liefert einen Ergebniswert"
  ]
},
{
  "tag": "Schlüsselwoerter",
  "patterns": [
    "Schlüsselwoerter"
  ],
  "responses": [
    "Spezielle Namen | Sind von Python reserviert und dürfen nicht für eigene Namen verwendet werden | Werden zur Einleitung oder Hervorhebung von Programmteilen genutzt | Werden in Python immer klein geschrieben | z. B. break, continue, if, else, in, return, while"
  ]
},
{
  "tag": "Datentypen",
  "patterns": [
    "Datentypen"
  ],
  "responses": [
    "Datentypen geben an, ob die Daten Zahlen, Zeichenketten, Wahrheitswerte oder zusammengesetzte Objekte sind. Python ist eine dynamisch typisierte Programmiersprache -> Bei der Deklaration einer Variablen wird kein Datentyp angegeben -> Python Interpreter konvertiert bei Bedarf automatisch den Datentyp
  ]
}

```

```

        Python unterscheidend zwischen folgenden Datentypen - Primitive Datentypen:
        Number (Integer, Float, Complex), String, Boolean | Nicht primitive Datentypen: Liste"
    ]
},
{
    "tag": "Variablennamen",
    "patterns": [
        "Variablennamen"
    ],
    "responses": [
        "Für Variablennamen gelten folgende Regeln: Sie dürfen kein Leerzeichen
        enthalten | Sie dürfen nur aus Buchstaben und Ziffern bestehen - das erste
        Zeichen muss ein Buchstabe sein | Groß- und Kleinbuchstaben erlaubt - wird
        auch unterschieden | Sie dürfen keine Umlaute oder scharfes-S (ß) enthalten |
        Sie dürfen ein einziges Sonderzeichen, den Unterstich, enthalten | Sie dürfen
        nicht mit reservierten Wörtern identisch sein"
    ]
},
{
    "tag": "Variable",
    "patterns": [
        "Variablen",
        "Konstante"
    ],
    "responses": [
        "Variablen sind Container für Informationen, zum Beispiel für eine Zahl
        oder für einen Text. Der Inhalt einer Variable kann sich jederzeit ändern,
        ist also variabel. Variablen können Daten speichern. Um dies sinnvoll zu tun,
        benötigt die Variable einen Namen. Über den Variablennamen kann der Programmierer
        im Laufe des Programmes darauf zugreifen und auch verschiedene Variablen
        auseinanderhalten. Um eine Variable zu verwenden, muss der Programmierer
        zunächst die Variable erstellen. Das nennt man deklarieren. Dies geschieht
        mit dem Schlüsselwort var. Eine Konstante ist vom Prinzip wie eine Variable.
        Nur speichert eine Konstante unveränderbare Werte. Bei der Deklaration muss
        der Variablen gleich der unveränderbare Wert zugewiesen werden. Konstanten
        werden von Python nicht unterstützt."
    ]
},
{
    "tag": "Wiederholung",
    "patterns": [
        "Wiederholung",
        "Schleife",
        "Iteration"
    ],
    "responses": [
        "Mit dem Programmkonstrukt Wiederholung, auch Schleife oder Iteration genannt,
        wird ein Verarbeitungsteil solange wiederholt, wie es der Steuerungsteil vorgibt.
        Wird eine Wiederholung in Abhängigkeit von einer Bedingungsprüfung ausgeführt,
        so kann diese vor oder nach dem Verarbeitungsteil stehen."
    ]
},
{
    "tag": "Einfache Alternative",
    "patterns": [
        "Einfache Alternative"
    ],
    "responses": [
        "Eine Auswahl / Verzweigung wird benötigt für alternative Verarbeitungen
        bei logischem Ablauf eines Programms. Eine einfache Alternative besteht aus

```

```

    zwei Verarbeitungsteilen und einem Steuerungsteil mit Bedingung. Die Bedingung bestimmt, welcher der beiden Verarbeitungsteile ausgeführt wird."
  ]
},
{
  "tag": "Grafische Ablaufstrukturen",
  "patterns": [
    "Grafische Ablaufstrukturen"
  ],
  "responses": [
    "Grafische Ablaufstrukturen können vorteilhaft sein, um digitale Steuerungsaufgaben, deren Lösung auf der Anwendung eines Algorithmus basieren, zu beschreiben. Mit den zwei grafischen Darstellungsmethoden Programmablaufplan und Struktogramm kann der sich ergebende Programmablauf dargestellt werden."
  ]
},
{
  "tag": "Entwicklungsumgebung",
  "patterns": [
    "Entwicklungsumgebung"
  ],
  "responses": [
    "Definition: Sammlung der wichtigsten Werkzeuge zur Softwareentwicklung unter einer Oberfläche. Bestandteile: Editor: Schreiben des Quellcodes, Quelltextformatierung, Syntaxhervorhebung | Compiler (Interpreter): Übersetzung von Quellcode (Programm in einer Programmiersprache) in Maschinensprache | Debugger: Suche von Laufzeitfehlern, -> Fehler bei Programmausführung | Ausführbarer Code: Maschinensprache, Für den Prozessor erforderliche Darstellung von Befehlen im binären Zahlenformat."
  ]
},
{
  "tag": "Agile Softwareentwicklung",
  "patterns": [
    "Agile Softwareentwicklung",
    "SCRUM"
  ],
  "responses": [
    "Ein agiler Entwicklungsprozess erhöht die Transparenz und Veränderungsgeschwindigkeit und führt zu einem schnelleren Einsatz der zu entwickelten Software. Das Ziel dabei ist es, den Entwicklungsprozess flexibler und schlanker zu machen als es bei den klassischen, plangetriebenen Vorgehensmodellen der Fall ist. Bei der agilen Entwicklung wird versucht, die Entwurfsphase auf ein Mindestmaß zu reduzieren und im Entwicklungsprozess so früh wie möglich zu ausführbarer Software zu gelangen. Diese wird in regelmäßigen, kurzen Abständen mit dem Kunden abgestimmt. So soll es möglich sein, flexibel auf Kundenwünsche einzugehen, um so die Kundenzufriedenheit insgesamt zu erhöhen. Agile Softwareentwicklung zeichnet sich durch selbstorganisierende Teams sowie eine iterative und inkrementelle Vorgehensweise aus. Rollen: Product Owner, Agile Master und Implementation Team. Meetings: Planning Meeting, Stand-Up Meeting, Sprint Review und Retrospektive."
  ]
},
{
  "tag": "V-Modell",
  "patterns": [
    "V-Modell",
    "V Modell"
  ],
  "responses": [
    "Das V-Modell ist ein Vorgehensmodell zum Planen und Durchführen von Entwicklungsprojekten unter Berücksichtigung von Qualitätssicherungsmaßnahmen über den gesamten Systemlebenszyklus. Beim V-Modell werden alle Phasen nacheinander, also linear, durchlaufen. Aber für jede Stufe werden entsprechende Testaktivitäten definiert. Einerseits ermöglicht das, die Softwarequalität effizienter zu kontrollieren und damit Projektrisiken auf ein Minimum zu reduzieren. Andererseits gehört das V-Modell dadurch zu einem der kosten- und

```

```

    zeitaufwändigsten Modelle. Ebenso gibt es auch hier keine Möglichkeit, Softwareanforderungen während der Entwicklungsphase neu zu bewerten."
  ]
},
{
  "tag": "Wasserfallmodell",
  "patterns": [
    "Wasserfallmodell"
  ],
  "responses": [
    "Bei diesem sequentiellen Vorgehensmodell ist der Entwicklungsprozess in aufeinanderfolgenden Phasen organisiert: Analyse, Entwurf, Codierung, Test, Installation, Wartung. Dabei wird jede einzelne Phase stark dokumentiert. Die Projektleitung, und ggf. der Kunde, können damit den Fortschritt des Prozesses anhand der Ergebnisdokumente verfolgen. Sobald die vorhergehende Stufe beendet wird, beginnt die nächste, die auf konkreten Ergebnissen der vorherigen Phase basiert. Deshalb gibt es keine Möglichkeit, beispielsweise Softwareanforderungen, während der Entwicklungsphase neu zu bewerten."
  ]
},
{
  "tag": "Softwarequalitaet",
  "patterns": [
    "Softwarequalitaet"
  ],
  "responses": [
    "Unter dem Begriff der Softwarequalität versteht man die gesamten Merkmale und Eigenschaften eines Softwareprodukts, die sich auf dessen Eignung beziehen, um spezifische und im Vorfeld definierte Erfordernisse zu erfüllen. Merkmale: Funktionalität, Zuverlässigkeit. Effizienz, Benutzbarkeit, Übertragbarkeit und Änderbarkeit"
  ]
},
{
  "tag": "Arten von Softwaretests",
  "patterns": [
    "Softwaretests"
  ],
  "responses": [
    "Funktionale Tests bzw. Funktionstests überprüfen ein System in Bezug auf funktionale Anforderungsmerkmale wie Korrektheit und Vollständigkeit. | Nicht-funktionale Tests überprüfen die nicht funktionalen Anforderungen, wie z. B. die Sicherheit, die Gebrauchstauglichkeit oder die Zuverlässigkeit eines Systems. Dabei steht nicht die Funktion der Software (Was tut die Software?) im Vordergrund, sondern ihre Funktionsweise (Wie arbeitet die Software?). | Schnittstellentests testen die Funktionalität bei der Zusammenarbeit voneinander unabhängiger Komponenten (für jede der beteiligten Komponente, anhand der Spezifikation, beispielsweise mit Hilfe von Mock-Objekten)
  ]
}

```

```

    | Fehlertests testen, ob die Verarbeitung von Fehlersituationen korrekt, d.h.
    wie definiert erfolgt."
  ]
},
{
  "tag": "Top Down",
  "patterns": [
    "Top Down",
    "Top-Down"
  ],
  "responses": [
    "Vorgehen vom Abstrakten zum Konkreten | Schwerpunkt liegt auf Planung und
    Verständnis | Code kann erst geschrieben werden, wenn System gut genug ge-
    plant ist"
  ]
},
{
  "tag": "Bottom Up",
  "patterns": [
    "Bottom-Up",
    "Bottom Up"
  ],
  "responses": [
    "Vorgehen vom Konkreten zum Abstrakten | Programmbestandteile klar defi-
    niert | Aus diesen Teilen werden Elemente des Programms zusammengefasst |
    Programmieren beginnt schneller"
  ]
},
{
  "tag": "Imperative Programmiersprachen",
  "patterns": [
    "Imperative Programmiersprachen"
  ],
  "responses": [
    "Dieses Programmierparadigma gibt vor, dass ein Programm eine Folge von An-
    weisungen ist. Der Programmcode gibt also bei diesen Paradigmen eine Abfolge
    von Befehlen vor und teilt dem Rechner auf diese Weise mit, in welcher Rei-
    henfolge welche Operationen abzulaufen haben, damit das erwartete Ergebnis

```

```

    erreicht wird. Beim Schreiben von imperativen Programmen werden also Befehle
    aneinandergereiht, wodurch bestimmte Funktionen erreicht werden."
  ],
},
{
  "tag": "Deklarative Programmiersprachen",
  "patterns": [
    "Deklarative Programmiersprachen"
  ],
  "responses": [
    "Deklarative Sprachen beschreiben lediglich das Ziel selbst. Die Realisie-
    rung wird vom so genannten Interpreter der Sprache übernommen. Bei der dekla-
    rativen Programmierung wird der Fokus auf das Was gelegt: Was ist das Problem
    bzw., was soll das Ergebnis sein?"
  ]
},
{
  "tag": "Programmierung",
  "patterns": [
    "Programmierung"
  ],
  "responses": [
    "Entwickelte Modelle / Algorithmen werden in einer Programmiersprache so
    spezifiziert, dass sie auf Rechnern ausgeführt werden können."
  ]
},
{
  "tag": "Determinismus",
  "patterns": [
    "Determinismus"
  ],
  "responses": [
    "Zu jedem Zeitpunkt der Ausführung besteht höchstens eine Möglichkeit der
    Fortsetzung. Der Folgeschritt ist also eindeutig bestimmt. Besteht keine Mög-
    lichkeit zur Fortsetzung der Ausführung, so vereinbart man, dass der Algo-
    rithmus terminiert. Hat ein Algorithmus an mindestens einer Stelle zwei oder
    mehr Möglichkeiten der Fortsetzung, von denen eine nach belieben ausgewählt
    werden kann, so heißt er nicht-deterministisch."
  ]
},
{
  "tag": "Determiniertheit",
  "patterns": [
    "Determiniertheit"
  ],
  "responses": [
    "Der Algorithmus muss bei gleichen Voraussetzungen (Parametern und Startbe-
    dingungen) stets das gleiche Ergebnis liefern."
  ]
},
{
  "tag": "Terminierung",
  "patterns": [
    "Terminierung"
  ],
  "responses": [
    "Nach endlich vielen Schritten muss der Algorithmus enden und ein Ergebnis
    liefern. Das Terminieren einen Algorithmus darf nicht mit seiner Endlichkeit
    verwechselt werden. Es ist durchaus möglich, durch eine endliche Beschreibung

```

```

    (finit) einen Prozess (z. B. mit Endlosschleife) zu definieren, der nicht
    nach endlicher Zeit beendet wird, also nicht terminiert."
  ],
},
{
  "tag": "Endlichkeit",
  "patterns": [
    "Endlichkeit"
  ],
  "responses": [
    "Die Beschreibung des Algorithmus muss endlich sein. Die Anzahl der Anwei-
    sungen ist auf eine endliche Anzahl von Daten begrenzt."
  ]
},
{
  "tag": "Ausfuehrbarkeit",
  "patterns": [
    "Ausfuehrbarkeit"
  ],
  "responses": [
    "Jeder Einzelschritt muss ausfuehrbar sein."
  ]
},
{
  "tag": "Eindeutigkeit",
  "patterns": [
    "Eindeutigkeit"
  ],
  "responses": [
    "Ein Algorithmus darf keine widerspruechliche Beschreibung haben. Diese muss
    eindeutig sein."
  ]
},
{
  "tag": "Allgemeinguelteigkeit",
  "patterns": [
    "Allgemeinguelteigkeit"
  ],
  "responses": [
    "Der Algorithmus loest eine Vielzahl von Problemen gleicher Art oder Klasse.
    Die Problembeschreibung erfolgt ueber Parameter oder Eingabedaten."
  ]
},
{
  "tag": "Algorithmus",
  "patterns": [
    "Algorithmus"
  ],
  "responses": [
    "Eindeutige Vorgehensweise (unabhaengig von der Programmiersprache) zur Lo-
    sung eines Problems. Anhand dieses Loesungsplans werden in Einzelschritten
    Eingabedaten in Ausgabedaten umgewandelt."
  ]
},
{
  "tag": "Eigenschaften Software",
  "patterns": [
    "Software"
  ],
  "responses": [
    "Software ist ein immaterielles Produkt, unterliegt keinem Verschleiss, wird
    nicht durch physikalische Gesetze begrenzt und ist im Allgemeinen leicht und
    schnell abaendbar."
  ]
},
{
  "tag": "Big Data",

```

```

"patterns": [
  "Fuenf Eigenschaften Big Data",
  "5 Merkmale Big Data",
  "Big Data"
],
"responses": [
  "Volume - Datenmenge, Variety - Datenvielfalt, Velocity - Geschwindigkeit,
  Value - Werthaltigkeit, Validity - Widerspruchsfreiheit"
]
},
{
  "tag": "Datenflut",
  "patterns": [
    "Wo kommen die Daten her?",
    "Weitere Beispiele Daten?"
  ],
  "responses": [
    "Taglich erzeugt jeder Mensch ca. 1.5 GB",
    "Mobile Endgerate generieren 44 ZB.",
    "50 Mrd. vernetzte Gerate generieren 4 ZB.",
    "YouTube speichert 72 Std. Videomaterial pro Min.",
    "Facebook verarbeitet taglich 4 PB.",
    "0.5 TB Daten erzeugt eine Boeing 787 pro Flug.",
    "Das Gesundheitswesen fuhrte bereits zu 150 EB weltweit. Taglich werden es
    3 TB mehr.",
    "Der Einzelhandelskonzern Wal-Mart generiert 2.5 PB pro Jahr.",
    "Autonomes fahren verursacht 4 TB pro Tag und Auto.",
    "Intelligente Fabriken erzeugen 1 PB Daten pro Fabrik und Tag.",
    "90 % der vorhandenen Daten entstanden in den letzten 2 Jahre."
  ]
},
{
  "tag": "Netzeffekt",
  "patterns": [
    "Netzeffekt"
  ],
  "responses": [
    "Auswirkungen der Teilnahme an einem Netz oder Netzwerk auf die Akteure 1.
    Der Nutzen eines Teilnehmers an einem Netzwerk (oder Standard) steigt mit der
    Zahl der Nutzer (marginal effect) | 2. Nutzenzuwachs gilt nicht nur fur den
    einzelnen Teilnehmer, sondern auch fur alle anderen Akteure des Netzwerks.
    Auswirkungen von Netzeffekten: 1. Netzeffekte legen fur Akteure der Plattfor-
    mokonomie Wachstumsstrategien nahe, um moglichst schnell einen Standard zu
    schaffen. | 2. Standardisierung fuhrt zu Marktsicherheit: Bei Fehlen von
    Standards sehen sich Nachfrager mit dem Risiko konfrontiert, auf einen nicht
    zukunftsfahigen Standard bzw. auf einen sich im Vergleich schlechter entwi-
    ckelndes Netzwerk zu setzen (= Marktunsicherheit). | 3. Folge: auf Netzwerk-
    markten werden Standardisierungen bis hin zur Monopolisierung vorangetrieben
  ]
}

```

```

    (nicht nur der Monopolist hat Interesse an der Ausschaltung des Wettbewerbs
    hat, sondern auch die Nachfrager)."
```

],  
 },  
 {  
 "tag": "Netz",  
 "patterns": [  
 "Netz"  
 ],  
 "responses": [  
 "Ein Netz wird durch Akteure gebildet, die denselben Standard verwenden."  
 ]  
 },  
 {  
 "tag": "Plattform",  
 "patterns": [  
 "Plattform",  
 "Platform"  
 ],  
 "responses": [  
 "Plattformen sind digitale Marktplätze und bilden damit das Geschäftsmodell  
 der aktuellen und zukünftigen Wirtschaft ab. Hier stehen vor allem Partner  
 und Netzwerke im Fokus. Ein Wesensmerkmal und Erfolgsfaktor von Plattformen  
 ist die Tatsache, dass sie Mehrwerte für alle Beteiligten schaffen: Der An-  
 bieter profitiert von geringeren Transaktionskosten, weil Interaktionen auto-  
 matisiert über die gemeinsame technische Plattform ablaufen. Die Interessen-  
 ten können durch die Preistransparenz und die Bewertungen anderer Kunden  
 leichter das beste Angebot für ihren Bedarf finden. Je mehr Anbieter auf der  
 Plattform sind, desto interessanter wird sie dabei für die Kunden und eine  
 hohe Kundenbasis lockt wiederum zusätzliche Anbieter an."  
 ]  
 },  
 {  
 "tag": "Digitale Disruption",  
 "patterns": [  
 "Disruption"  
 ],  
 "responses": [  
 "Mit der Disruption wird eine Art Störung bezeichnet, welche bestehende  
 konventionelle Geschäftsmodelle oder Angebote durch eine digitale Innovation  
 ablöst oder versucht sie vom Markt zu drängen."  
 ]  
 },  
 {  
 "tag": "Exploration-Exploitation-Dilemma",  
 "patterns": [  
 "Exploration Exploitation Dilemma",  
 "Exploration",  
 "Exploitation",  
 "Dilemma",  
 "Exploration-Exploitation-Dilemma"  
 ],  
 "responses": [  
 "Der Gama Parameter hat einen Wert zwischen 0 und 1 ( $0 \leq \text{Gamma} < 1$ ). Je  
 kleiner Gamma (je näher an Null), desto eher tendiert der Agent dazu die un-  
 mittelbaren Belohnungen zu berücksichtigen. -> Agent wird nicht die Umgebung  
 erkunden (keine Exploration) Je größer Gamma (je näher an Eins), desto eher  
 werden zukünftige Belohnungen mit größerer Gewichtung berücksichtigt und der  
 Agent ist bereit auf die Belohnung zu warten. -> Agent wird alle möglichen  
 Aktionen ausprobieren (Exploration) um zum Ziel zu gelangen. Am Anfang muss  
 der Gama-Parameter am größten sein. Am Anfang kennt der Agent die Umgebung  
 noch nicht (kein Wissen in der Q-Matrix) -> Exploration notwendig. Im Laufe  
 des Trainings kann der Gama-Parameter reduziert werden. Dann wird das Wissen

```

    aus der Q-Matrix verwendet um sich für den nächsten Schritt zu entscheiden. -
    > Exploitation"
  ]
},
{
  "tag": "Q-Learning",
  "patterns": [
    "Q-Learning",
    "Q Learning",
    "Q-Matrix",
    "Q Matrix"
  ],
  "responses": [
    "1. Definieren des Gama-Parameters, und das Erstellen der Belohnungsmatrix R | 2. Initialisieren der Q-Matrix mit der Null-Matrix | 3. Wiederhole für jede Episode, bis Training beendet ist: Wählen eines beliebigen Startzustandes | 4. Der initiale Zustand wird zum aktuellen Status | 5. Wählen einer möglichen Aktion aus allen Aktionen die aus dem aktuellen Zustand heraus möglich sind | 6. Anwenden der gewählten Aktion, um zum nächsten Zustand zu gelangen | 7. Ermitteln des maximalen Q-Wertes für diesen Zustand, basierend auf allen möglichen Aktionen | 8. Dafür wird diese Formel verwendet:  $Q(\text{Zustand}, \text{Aktion}) = R(\text{Zustand}, \text{Aktion}) + \text{Gamma} * \text{Max}[Q(\text{nächster Zustand}, \text{alle Aktionen})]$  | 9. Wiederholen der Schritte bis der aktuelle Status dem finalen Status entspricht | Der obige Algorithmus wird vom Agenten verwendet, um aus Erfahrungen zu lernen. Jede Episode entspricht einer Trainingseinheit. In jeder Trainingseinheit erkundet der Agent die Umgebung (dargestellt durch Matrix R) und erhält die Belohnung (falls vorhanden), bis er den Zielzustand erreicht. Der Zweck des Trainings besteht darin, das Gehirn unseres Agenten zu verbessern, dass durch die Matrix Q dargestellt wird. Mehr Training führt zu einer optimierten Matrix Q."
  ]
},
{
  "tag": "Elemente bestaerkendes Lernen",
  "patterns": [
    "Bestaerkendes Lernen",
    "Bestaerktes Lernen"
  ],
  "responses": [
    "Agent: der RL-Algorithmus welcher durch Versuch und Irrtum lernt | Umwelt: die Umgebung in der sich der Agent bewegt | Aktionen: alle möglichen Schritte, die der Agent in der Umwelt ausführen kann | Zustände: aktueller Status des Agenten, wird von der Umgebung zurückgemeldet | Belohnung: sofortige Rückmeldung von der Umgebung, um den letzten Zustand zu bewerten | Policy: Strategie mit welcher der Agent die nächste Aktion basierend auf dem aktuellen Status ermittelt"
  ]
},
{
  "tag": "Unterschied ueberwachtes und bestaerkendes Lernen",
  "patterns": [
    "Ueberwachtes und bestaerkendes Lernen",
    "Ueberwachtes Lernen und bestaerkendes Lernen"
  ],
  "responses": [
    "Definition: Bei überwachtem Lernen: Der Algorithmus lernt von gelabelten Daten. Und arbeitet mit nicht gelabelten Daten um Label vorherzusagen. | Bei bestärkendem Lernen: Ein Agent interagiert in einer Umgebung indem Aktionen

```

```

ausgeführt werden und dabei wird von Belohnung und Bestrafung gelernt. Danach
kann sich der Agent selbständig in der Umgebung interagieren.",
  "Entscheidungsstil: Bei überwachtem Lernen: Reinforcement Learning hilft
die Entscheidungen sequentiell zu treffen. | Bei bestärkendem Lernen: Rein-
forcement Learning hilft die Entscheidungen sequentiell zu treffen.",
  "Training: Bei überwachtem Lernen: Wird in einer Umgebung trainiert und
muss mit der Umgebung interagieren. | Bei bestärkendem Lernen: Wird auf Bei-
spielaten trainiert.",
  "Abhängigkeiten: Bei überwachtem Lernen: In RL-Methoden ist die Lernent-
scheidungen untereinander abhängig. Daher sollten alle abhängigen Entschei-
dungen mit entsprechender Belohnung versehen sein. | Bei bestärkendem Lernen:
Im Supervised learning sind die Entscheidungen unabhängig voneinander, daher
werden die Labels für jede Entscheidung gegeben.",
  "Beispiel: Bei überwachtem Lernen: Schachspiel | Bei bestärkendem Lernen:
Objekterkennung"
]
},
{
  "tag": "LIME",
  "patterns": [
    "LIME",
    "Local Interpretable Model Agnostic Explanations"
  ],
  "responses": [
    "Zu Beginn wird das Modell wie gewohnt trainiert. Das erlernte Verhalten
ist meist leider nicht greifbar. Der Algorithmus wird als Black Box angese-
hen. Danach wird ein neuer Datensatz anhand der gegebenen Merkmalswerte durch
das Modell bewertet. Danach kann für eine Erklärbarkeit für die erhaltene Be-
wertung gesucht werden. Dafür werden die Merkmalswerte des zu klassifizieren-
den Datensatzes permutiert. Die Merkmalswerte werden dafür multipliziert und
leicht verändert. Anschließend werden die permutierten Datensätze ebenfalls
durch das Modell bewertet. Nun muss entschieden werden, wie viele Merkmale in
die Erklärung mit einfließen sollen (in der Regel zwischen 5 und 10). Die An-
zahl muss ein Gleichgewicht zwischen der Komplexität und der Erklärbarkeit
des Modells sicherstellen. Mithilfe der erhaltenen Bewertungen der permutier-
ten Daten (Dimensionalität reduziert auf die Anzahl der Variablen welche in
die Erklärung einfließen sollen) kann das lokale Verhalten des Modells ange-
näbert werden. Dazu wird ein einfaches lineares Modell trainiert welches sich
dem originalen Modell in der Nähe des zu bewertenden Datensatzes gut annä-
hert. Die Erklärbarkeit ist in diesem Fall global nicht zutreffend, sondern
nur lokal um den zu klassifizierenden Datensatz herum zutreffend."
  ]
},
{
  "tag": "Datenschutzgrundverordnung",
  "patterns": [
    "Datenschutzgrundverordnung",
    "Datenschutzverordnung"
  ],
  "responses": [
    "1. regelt dem Umgang personenbezogener Daten, 2. soll das Datenschutzrecht
in das Zeitalter der Digitalisierung führen, 3. betrifft jedes Unternehmen
(das im Internet tätig ist oder personenbezogene Daten nutzen) | Speziell für
ML-Modelle wichtig: Werden personenbezogene Daten erhoben, so teilt der Ver-

```

```

    antwortliche folgendes mit: das Bestehen einer automatisierten Entscheidungs-
    findung und aussagekräftige Informationen über die involvierte Logik. Art.
    13-15 & 22 Regulation (EU) 2016/679"
  ],
},
{
  "tag": "Verstaendnis ML",
  "patterns": [
    "Verstaendnis ML",
    "Verstaendnis Machine Learning",
    "Verstaendnis maschinelles Lernen",
    "Entscheidung von ML",
    "Entscheidung von Machine Learning",
    "Entscheidung maschinelles Lernen"
  ],
  "responses": [
    "Die Fähigkeit, anderen Menschen die Gründe für eine Entscheidung mitzutei-
    len, ist ein wichtiger Aspekt der menschlichen Intelligenz. Dies ist beson-
    ders wichtig in der sozialen Interaktion und im Bildungskontext, wo Schüler
    versuchen die Argumentation des Lehrers zu verstehen.",
    "Wenn wir wissen, auf welchen Features und Grundlagen unser Modell Ent-
    scheidungen und Vorhersagen trifft, können wir besser einschätzen, ob diese
    gelernten Regeln Sinn machen und auch für generelle Vorhersagen taugen.",
    "Mit dieser Art Sanity Check können wir vermeiden, dass unser Modell fal-
    sche Schlüsse zieht.",
    "Nur wenn wir die Entscheidungen eines Modells nachvollziehen können, wer-
    den wir ihm wirklich vertrauen.",
    "Lernen vom System",
    "Einhalten von Gesetzen"
  ]
},
{
  "tag": "Unterschiede Machine Learning und Deep Learning",
  "patterns": [
    "ML und DL",
    "Machine Learning und Deep Learning"
  ],
  "responses": [
    "Datenstruktur: Bei Machine Learning: Strukturierte Daten | Bei Deep Learn-
    ing: Unstrukturierte und strukturierte Daten",
    "Datensatzgröße: Bei Machine Learning: Klein - Groß | Bei Deep Learning:
    Groß - extrem groß",
    "Hardware: Bei Machine Learning: Funktioniert mit einfacher Hardware | Bei
    Deep Learning: Braucht leistungsstarke Computer (mit GPUs). Neuronale Netze
    multiplizieren Matrizen, die sehr viel Rechenzeit brauchen - GPUs beschleuni-
    gen den Vorgang. -> Berechnungen auf Cluster in Cloud durchführen",
    "Feature Extraktion: Bei Machine Learning: Sie müssen die Merkmale in der
    Regel manuell extrahieren und verstehen. | Bei Deep Learning: Sie müssen die
    Merkmale nicht manuell extrahieren und verstehen.",
    "Dauer des Trainings: Bei Machine Learning: Ein paar Minuten bis Stunden. |
    Bei Deep Learning: Bis zu Wochen und Monaten. Neuronale Netze müssen enorm
    viele Gewichte berechnen.",
    "Interpretierbarkeit: Bei Machine Learning: Einige Algorithmen sind leicht
    zu interpretieren (logische Regression, einfache Entscheidungsbäume), andere
    sind fast unmöglich (SVM). | Bei Deep Learning: Schwer zu interpretieren und
    oft sogar unmöglich, da tiefe neuronale Netze verwendet werden."
  ]
},
{
  "tag": "Flaches und tiefes neuronales Netz",
  "patterns": [
    "Flaches und tiefes neuronales Netz",
    "Flaches tiefes neuronales Netz"
  ],
  "responses": [
    "Die Anzahl der Neuronen und Neuronenschichten sowie der Verbindungsmög-
    lichkeiten der Neuronen verschiedener Schichten bestimmt die Komplexität (die

```

```

Tiefe) des Neuronalen Netzes und dessen Fähigkeit, Problemstellungen zu lösen. Flache neuronale Netze haben keine oder nur eine Hidden-Schicht. Tiefe neuronale Netze dagegen zählen mehr als zwei Hidden-Schichten als tief. Heute haben tiefe neuronale Netze mehrere hundert Schichten."
]
},
{
  "tag": "Loesung Support Vector Machine",
  "patterns": [
    "Support Vector Machine",
    "SVM"
  ],
  "responses": [
    "Nicht linear trennbare Daten in einen Raum mit höherer Dimension überführen, oder zusätzliche Merkmale berechnen, so dass die Daten linear trennbar werden. Bei der Rücktransformation in den ursprünglichen Raum wird die Hyper-ebene zu einer nichtlinearen Trennfläche im ursprünglichen Merkmalsraum."
  ]
},
{
  "tag": "Besonderheit Ergebnis Klassifikation",
  "patterns": [
    "Klassifikation"
  ],
  "responses": [
    "Das Ergebnis ist kein einzelner Wert, sondern die Wahrscheinlichkeitswerte für die Zugehörigkeit zu jeder Klasse. -> Zuordnung zur Klasse mit der höchsten Wahrscheinlichkeit."
  ]
},
{
  "tag": "Einfache lineare und multiple lineare Regression",
  "patterns": [
    "Lineare und multiple lineare Regression"
  ],
  "responses": [
    "Im Unterschied zur einfachen linearen Regression, bei der nur eine unabhängige Variable (x) untersucht wird, modelliert die multiple lineare Regression die Einflüsse mehrerer unabhängiger Variablen (x1, x2, x3, x4, etc.) auf eine abhängige Variable (y). Allerdings wird auch bei dieser Methode angenommen, dass die Zusammenhänge zwischen den unabhängigen Variablen und den abhängigen Variablen linearer Natur sind."
  ]
},
{
  "tag": "Random Forest",
  "patterns": [
    "zufaelliger Wald",
    "Zufallswald",
    "Random Forest"
  ],
  "responses": [
    "Random Forest ist ein Verfahren, das häufig im Machine Learning verwendet wird. Es zählt zu den überwachten Lernverfahren (Supervised Learning) und nutzt die Ergebnisse einer Vielzahl verschiedener Entscheidungsbäume, um bestmögliche Entscheidungen oder Vorhersagen zu treffen. Die Entscheidungsbäume wurden nach einem Zufallsprinzip unkorreliert erstellt. Jeder Baum

```

```

    trifft für sich einzelne Entscheidungen. Aus der Menge der Einzelentscheidungen liefert der Algorithmus eine endgültige Entscheidung."
  ]
},
{
  "tag": "Vorteil Entscheidungsbaum",
  "patterns": [
    "Entscheidungsbaum"
  ],
  "responses": [
    "Vorteil vom Entscheidungsbaum: anschauliche, leicht nachvollziehbare Darstellung der Entscheidungsfindung",
    "Vorteil vom Entscheidungsbaum: ermöglicht die Visualisierung der Klassifizierungsregeln -> White Box Modell",
    "Vorteil vom Entscheidungsbaum: mathematisch leicht abbildbar",
    "Vorteil vom Entscheidungsbaum: Keine so aufwändige Datenvorbereitung, da kategoriale Merkmalswerte nicht in Zahlen codiert werden müssen",
    "Vorteil vom Entscheidungsbaum: Es können numerische und kategoriale Merkmalswerte gleichzeitig verwendet werden"
  ]
},
{
  "tag": "Nachteil kNN",
  "patterns": [
    "Nachteil kNN",
    "Contra kNN",
    "Nachteil des k-Nearest Neighbour Klassifikators"
  ],
  "responses": [
    "Einer der größten Nachteile von kNN-Klassifikatoren ist, dass der Klassifikator zur Laufzeit immer alle verfügbaren Trainingsdaten benötigt. Es werden im Voraus keine Zusammenhänge gelernt. Die gesamte Berechnung findet zur Laufzeit bei der Klassifikation eines Datensatzes statt. Dies führt zu erheblichen Rechenaufwand, besonders bei vielen Trainingsdaten mit einer hohen Dimensionalität."
  ]
},
{
  "tag": "k ungerade",
  "patterns": [
    "k ungerade",
    "ungerades k",
    "Wahl von k"
  ],
  "responses": [
    "Um die Klassenzugehörigkeit des zu klassifizierenden Datenpunktes zu bestimmen, wird die Mehrheit der Klassenzugehörigkeiten der k-nächsten Nachbarn

```

```

    ermittelt. Um sicher eine Mehrheitsentscheidung herbeiführen zu können, wird
    eine ungerade Anzahl an Nachbarn in diese Entscheidung mit einbezogen."
  ],
},
{
  "tag": "k zu gross",
  "patterns": [
    "k zu gross",
    "grosses k"
  ],
  "responses": [
    "zu großes k: viele Objekte aus anderen Klassen in der Entscheidungsmenge"
  ]
},
{
  "tag": "k zu klein",
  "patterns": [
    "k zu klein",
    "kleines k"
  ],
  "responses": [
    "zu kleines k: hohe Sensitivität übergeben Ausreißern"
  ]
},
{
  "tag": "Klassifikationsverfahren",
  "patterns": [
    "Klassifikationsverfahren"
  ],
  "responses": [
    "K-nächste Nachbar Klassifikator: einfachster Klassifikator der Daten anhand
    der Klasse der nächsten Nachbarn klassifiziert.",
    "Lernende Vektorquantisierung: aus den Trainingsdaten wird eine begrenzte
    Anzahl an Prototypvektoren abgeleitet, welche die Trainingsdaten repräsentieren.
    Anschließend werden die Daten anhand der Klasse des nächsten Prototypvektors
    klassifiziert.",
    "Polynomklassifikator: Die Beziehung zwischen den Labels und den Merkmalen
    wird in den Entscheidungsfunktionen nachgebildet. Für jede Klasse wird eine
    Entscheidungsfunktion approximiert. Die Entscheidungsfunktionen liefern
    Wahrscheinlichkeitswerte für die Klassenzugehörigkeit.",
    "Entscheidungsbaum (decision tree): Vorhersage wird aufgrund von mehreren
    Zweigen von Wenn-Dann-Entscheidungen getroffen. In jedem Zweig werden
    Merkmalschwellwerte gefunden, die die verbleibende Trainingsdatenmenge am
    besten aufteilt hatte.",
    "Random forest: Entscheidungen aus einzelnen Entscheidungsbäumen werden
    kombiniert. Modell wird robuster und weniger anfällig für Overfitting.",
    "Support Vektor Maschine: Menge von Objekten wird durch eine Hyperebene in
    zwei Klassen unterteilt.",
    "Künstliches Neuronales Netz: bis zu einem gewissen Grad dem Aufbau des
    biologischen Gehirns nachempfunden. Es besteht aus einem abstrahierten
    Modell miteinander verbundener Neuronen, durch deren spezielle Anordnung
    und Verknüpfung sich Anwendungsprobleme lösen lassen."
  ]
},
{
  "tag": "Model Drift",
  "patterns": [
    "Model Drift",
    "Model-Drift",
    "Modell Drift"
  ],
  "responses": [
    "Model Drift bezieht sich auf die sich im Laufe der Zeit verschlechternde
    Vorhersageleistung eines Modells aufgrund einer Änderung der Umgebung, die
    gegen die Annahmen des Modells verstößt. Die Modelldrift ist eine
    Fehlbezeichnung, da sich nicht das Modell ändert, sondern die Umgebung,
    in der das Modell betrieben wird. Modelldrift nachzuweisen ist oft
    schwierig. Auch gibt

```

```

    es keinen einheitlichen Ansatz. Unterschiedliche Modellierungsprobleme erfordern unterschiedliche Lösungen. Möglichkeiten um Modelldrift nachzuweisen sind: Untersuchen der Modelldrift, trainiert auf vergangenen Daten | Untersuchen der Verteilung von Trainings- und Live-Daten | Untersuchen der Korrelationen zwischen Merkmalen | Untersuchen der Zielverteilungen. Model Drift entgegenwirken: Das Modell-Deployment (Modellbereitstellung) muss als kontinuierlicher Prozess behandelt werden. Anstatt ein Modell einmal bereitzustellen und zu einem anderen Projekt überzugehen, müssen Data Scientists ihre Modelle neu trainieren, wenn sie feststellen, dass die Datenverteilungen erheblich von denen des ursprünglichen Trainingssatzes abweichen. - Re-Training der Modelle muss automatisiert werden."
  ],
},
{
  "tag": "Aktives Lernen",
  "patterns": [
    "aktives Lernen",
    "active learning"
  ],
  "responses": [
    "Aktives Lernen: iteratives Verfahren, bei dem der Algorithmus die Daten selbständig auswählt, von denen er lernen möchte. Der Klassifikator wird mit einer kleinen Menge gelabelter Daten trainiert. Anschließend wird der Klassifikator verwendet um den größeren Teil der nicht gelabelten Trainingsdaten zu labeln. Aus diesen Daten wählt der Algorithmus diejenigen Daten aus, bei denen mit einer hohen Wahrscheinlichkeit das falsche Label vorhergesagt wird. Diese Daten werden dann von einem Orakel (meist ein Mensch) gelabelt und dem Trainingsdatenset mit den gelabelten Trainingsdaten hinzugefügt. Danach wird der Algorithmus mit dem vergrößerten gelabelten Trainingssatz trainiert und der Vorgang wiederholt. Dies geschieht normalerweise bis Konvergenz eintritt."
  ]
},
{
  "tag": "Teilüberwachtes Lernen",
  "patterns": [
    "Teilüberwachtes Lernen",
    "semi supervised learning",
    "semi-supervised-learning"
  ],
  "responses": [
    "Teilüberwachtes Lernen: iteratives Verfahren, bei dem der Lernprozess seine eigenen Vorhersagen verwendet, um sich selbst zu trainieren. Der Klassifikator wird mit einer kleinen Menge gelabelter Daten trainiert. Anschließend wird der Klassifikator verwendet um den größeren Teil der nicht gelabelten Trainingsdaten zu labeln. Eine Teilmenge mit den sichersten Vorhersagen wird aus dem Trainingsset mit den gelabelten Trainingsdaten hinzugefügt. Danach wird der Algorithmus mit dem vergrößerten gelabelten Trainingssatz trainiert und der Vorgang wiederholt. Dies geschieht normalerweise bis Konvergenz eintritt."
  ]
},
{
  "tag": "Wahrheitsmatrix",
  "patterns": [
    "Wahrheitsmatrix",
    "Konfusionmatrix",
    "Performance Metriken"
  ],
  "responses": [
    "Für binaere Klassifikatoren gilt, dass jede Beobachtung auf einen von zwei Zuständen abgebildet wird (z. B. JA und NEIN, oder 0 und 1, oder gesund und krank oder Hund und Katze). Die binaere Antwort des Klassifikators kann nun in Bezug auf die (unbekannte) Realität entweder korrekt oder falsch sein. Die Ergebnisse der Klassifikation werden in einer Klassifikationstabelle oder Wahrheitsmatrix (engl. confusion matrix) zusammengefasst; diese enthält die

```

```

    Anzahl aller Beobachtungen in den vier möglichen Kombinationen aus Klassifi-
    zierungsergebnis und Wirklichkeit. Falls das Klassifikationsergebnis richtig
    ist, spricht man von richtig positiv und richtig negativ, abhängig davon zu
    welcher Klasse die Beobachtung tatsächlich gehört. Falls der Klassifikator
    eine falsche Antwort liefert, spricht man von einer falsch positiven bzw.
    falsch negativen Entscheidung."
  ]
},
{
  "tag": "Kreuzvalidierung",
  "patterns": [
    "Kreuzvalidierung"
  ],
  "responses": [
    "Bei der Kreuzvalidierung wird der zur Verfügung stehende Beispieldatensatz
    in k möglichst gleich große Teilmengen unterteilt. Die Teilmengen weisen im
    Idealfall die gleiche Wahrscheinlichkeitsverteilung wie der gesamte Beispiel-
    datensatz auf. Nun werden k Durchläufe gestartet, bei denen jeweils die i-te
    Teilmenge als Testdatensatz und die verbleibenden k-1 Teilmengen als Train-
    ingsdaten verwendet werden. Nach jedem Durchlauf wird die Fehlerquote auf
    den Testfehlern bestimmt. Am Ende errechnet sich die Gesamtfehlerquote als
    Durchschnitt aus den Einzelfehlerquoten der k-Einzeldurchläufe."
  ]
},
{
  "tag": "Bias-Varianz-Dilemma",
  "patterns": [
    "Optimaler Punkt",
    "Bias-Varianz-Dilemma",
    "Bias Varianz Dilemma"
  ],
  "responses": [
    "Um Underfitting bzw. ein hohes Bias zu überwinden kann dem Modell grund-
    sätzlich neue Parameter hinzugefügt werden -> Modellkomplexität nimmt zu,
    Bias nimmt ab -> zusätzliche Parameter führen aber auch zu einer Zunahme der
    Varianz -> Bias-Varianz-Dilemma. Es gilt einen optimalen Punkt für unser Mo-
    dell zu finden, indem die Abnahme des Bias und die Varianzzunahme gleich
    sind."
  ]
},
{
  "tag": "Underfitting",
  "patterns": [
    "Underfitting",
    "Unter Anpassung"
  ],
  "responses": [
    "Underfitting -> das Modell entspricht weder den Trainingsdaten noch wurde
    es auf den Testdaten verallgemeinert -> hohes Bias, geringe Varianz"
  ]
},
{
  "tag": "Overfitting",
  "patterns": [
    "Overfitting",
    "Über Anpassung"
  ],
  "responses": [
    "Overfitting -> Datenpunkte können in zwei Komponenten verlegt werden, die
    Information selbst und stochastisches Rauschen -> das Modell lernt zusätzlich
    zu den Informationen noch das stochastische Rauschen -> das Modell generiert
    sehr gute Vorhersagen auf den Trainingsdaten aber sehr schlechte Vorhersagen

```

```

    auf den Testdaten -> das Gelernte kann nicht verallgemeinert werden -> hohe
    Varianz, niedriges Bias"
  ],
},
{
  "tag": "Ziel ML",
  "patterns": [
    "Ziel ML",
    "Ziel maschinelles Lernen"
  ],
  "responses": [
    "Ziel des maschinellen Lernens ist es, die Muster abzubilden und das Rau-
    schen zu ignorieren."
  ]
},
{
  "tag": "Supervised Learning Model Datensätze",
  "patterns": [
    "Supervised Learning Modell",
    "Überwachtes Lernmodell"
  ],
  "responses": [
    "Beispieldatensatz wie folgt aufteilen: 70% Trainingsdatensatz, 20% Vali-
    dierungsdatensatz, 10% Testdatensatz | Trainingsdaten: Algorithmus erlernt
    Muster und Zusammenhänge aus Trainingsdaten | Validierungsdaten: Während des
    Trainings wird anhand der Validierungsdaten überprüft, ob sich die Qualität
    des zu trainierenden Algorithmus verbessert | Testdaten: Der trainierte Algo-
    rithmus wird anhand für ihn noch unbekannte Daten getestet"
  ]
},
{
  "tag": "Regression und Klassifikation",
  "patterns": [
    "Regression und Klassifikation"
  ],
  "responses": [
    "Der Algorithmus (Regression, Klassifikation) erhält eine Reihe von Trai-
    ningsdaten mit entsprechenden Labels. Der Algorithmus lernt selbständig die
    Verbindung zwischen Trainingsdaten und Labels und kann diese dann auf belie-
    bige Eingaben verallgemeinern. Der Unterschied zwischen Klassifikation und
    Regression ist, dass wir bei der Regression kontinuierliche Werte und bei der
    Klassifikation diskrete Werte betrachten. Beispiele für kontinuierliche Werte
    sind Temperatur Geschwindigkeit oder die Krümmung einer Banane. Wohingegen
    Werte wie Farbe, Obstsorte oder Automarke diskrete Werte sind."
  ]
},
{
  "tag": "Mean shift Clustering",
  "patterns": [
    "Mean Shift Cluster",
    "Mean-Shift-Clustering"
  ],
  "responses": [
    "Um die Mean-Shift zu erklären, betrachtet man eine Reihe von Punkten im
    zweidimensionalen Raum. Man beginnt mit einem kreisförmigen Sliding-Window,
    welches an einem Datenpunkt (zufällig ausgewählt) zentriert ist und den Ra-
    dius r besitzt. Bei jeder Iteration wird das Sliding-Window in Richtung der
    höheren Dichte verschoben. Dabei wird das Zentrum des Sliding-Window auf den
    Mittelpunkt der Datenpunkte geschoben. Die Dichte innerhalb des Sliding-
    Window ist proportional zur Anzahl der Punkte darin. Durch Verschieben des
    Zentrums des Sliding-Window auf den Mittelwert der Datenpunkte im Window wird
    es allmählich in den Bereich der höheren Punktdichte geschoben. Man setzt das
    Verschieben des Sliding-Window solange fort, bis Konvergenz eintritt und so-
    mit der dichteste Bereich (lokales Maxima) der Datenpunkte gefunden wurde.
    Clusterzentren entsprechen schließlich den lokalen Maxima der Dichtevertei-
    lung der Datenpunkte im Merkmalsraum. Alle Datenpunkte, die denselben lokalen

```

```

Maxima zugewiesen werden können, gehören demselben Cluster an. Für eine Verteilung von Datenpunkten mit mehreren lokalen Maxima in der Dichtefunktion muss der Algorithmus von einer (möglichst großen) Anzahl verschiedener Anfangspunkte gestartet werden, um alle lokalen Maxima der Verteilung zu bestimmen."
]
},
{
"tag": "Kombination hierarchisches Clustern und K-means Clustering",
"patterns": [
"Kombination hierarchisches Clustern und K-means Clustering",
"hierarchische Clustern mit dem K-means Cluster Verfahren kombiniert",
"hierarchie-Clustern mit dem K-means Cluster Verfahren kombiniert"
],
"responses": [
"Durch gruppieren der beiden Clusterverfahren können die Nachteile der beiden Verfahren eliminiert werden. Daten werden initial mit hierarchischem Clusterverfahren gruppieren. Die gruppierten Daten werden anschließend mit dem k-means Verfahren verfeinert. Damit umgeht man den Nachteil des hierarchischen Clusters, dass die Zuordnung eines bestimmten Datenpunktes zu einem Cluster unverändert bleibt. Ebenso umgeht man den Nachteil des k-means Clustering, dass das Ergebnis von einem willkürlichen zufälligen Initialisierungswert abhängt."
]
},
{
"tag": "Hierarchisches Clustern",
"patterns": [
"Hierarchisches Clustern",
"Hierarchisches Clustering"
],
"responses": [
"Hierarchisches Clustern dient der Aufdeckung von Clusterstrukturen, wenn keine Kenntnisse über die Clusteranzahl verfügbar sind. | Vorgehen: 1. Erzeuge für jeden Punkt aus dem Datensatz ein separates Cluster | 2. Berechne den Abstand zwischen allen Clustern | Wiederhole: | 3. Verschmelze die beiden Cluster, die den geringsten Abstand zueinander haben | 4. Aktualisiere die Distanzen zwischen den Clustern bis alle Punkte in dem gleichen Cluster liegen | Da es in der Praxis nicht sinnvoll ist, als Endergebnis ein Cluster mit allen Datenpunkten zu erhalten, sollte der Algorithmus an geeigneter Stelle abbrechen. Eine verbreitete Methode hierfür ist der Abbruch, sobald der Abstand zwischen den Clustern eine bestimmte Distanz überschreitet. Vor- und Nachteile: + Keine Initialisierungsschwierigkeiten wie bei k-means | - Rechenaufwendig | - Je nach Wahl des Abstandsbegriffs erhält man unterschiedliche Clusterfolgen. | - Clusterzugehörigkeiten können im Laufe des Verfahrens nicht mehr verändert werden"
]
},
{
"tag": "K-means Clustering",
"patterns": [
"K Means",
"K-means"
],
"responses": [
"Die Aufgabe ist es unseren Datensatz in K-Cluster zu unterteilen. Dabei gehen wir wie folgt vor: Wir wählen uns K verschiedene Punkte als Anfangszentren. Dann ordnen wir jeden Datenpunkt den ihm nächsten Zentrum zu. Dann berechnen wir aus den jetzigen Clustern die Zentren neu. Dann ordnen wir wieder jeden Datenpunkt den ihm nächsten Cluster zu, usw. Das wird so lange wiederholt, bis sich die Zentren nicht mehr oder nur wenig verschieben. | Vor- und Nachteile: | + Einfach zu implementieren und zu verwenden | + Kommt mit vielen verschiedenen Datentypen zurecht | + Robust gegen verrauschte Daten | + Die KNN-Klassifizierung basiert ausschließlich auf lokalen Informationen | -

```

```

    Distanzmaß kann von irrelevanten Daten dominiert werden | - K-means hat Probleme, natürliche Cluster zu erkennen, die keine kugelförmige Struktur haben."
  ],
},
{
  "tag": "Lernverfahren ML",
  "patterns": [
    "Lernverfahren"
  ],
  "responses": [
    "Unüberwachtes Lernen (unsupervised learning, clustering): Unüberwachtes Lernen nutzt einzig einen Rohdatensatz ohne vorgegebene korrekte Outputs. Der Algorithmus teilt nun die Daten in Cluster oder Kategorien auf, jedoch ohne zu wissen, um welche Kategorien es sich handelt.",
    "Überwachtes Lernen (supervised learning): Algorithmus trainiert mit gelabelten Daten, sprich mit Eingaben und deren korrekten Ausgaben. Der Algorithmus vergleicht seinen kalkulierten Output mit den vorgegebenen Ausgaben. So erkennt er seine Fehler und passt seine Vorgehensweise entsprechend an.",
    "Bestärkendes Lernen (reinforcement learning): Anders als beim überwachtem und dem unüberwachtem Lernen gibt es beim Reinforcement Learning zunächst keine Daten. Der Algorithmus generiert Lösungen und Strategien auf Basis von erhaltenen Belohnungen im Trial-and-Error-Verfahren. Das Wissen und die Intelligenz entsteht während vielen verschiedenen Simulationsdurchläufen."
  ]
},
{
  "tag": "Dimensionsreduktion",
  "patterns": [
    "Dimensions Reduktion",
    "Dimensionsreduktion"
  ],
  "responses": [
    "Dimensionsreduktion: aussagelose Informationen loszuwerden, dabei aber die entscheidenden Teile zu behalten. z. B. Merkmalsauswahl (Feature Selection) Ein- bzw. Ausschließen von vorhandenen Merkmalen. Nur aussagekräftige und nützliche Merkmale werden beibehalten. Irrelevante oder redundante Merkmale werden entfernt",
    "Dimensionsreduktion: aussagelose Informationen loszuwerden, dabei aber die entscheidenden Teile zu behalten. z. B. Principal Component Analysis Erstellen von neuen Merkmalen (Hauptkomponenten). Vielzahl der Merkmale wird durch eine geringere Anzahl von Hauptkomponenten angenähert. Die PCA projiziert hochdimensionale Daten mit minimalem Informationsverlust in einen Raum mit geringer Dimension."
  ]
},
{
  "tag": "Z-Transformation",
  "patterns": [
    "Z-Transformation",
    "Z Transformation"
  ],
  "responses": [
    "Bei einer Z-Transformation werden die Abweichungen der ursprünglichen Werte von ihrem Mittelwert durch ihre Standardabweichung geteilt. Z-transformierte Variablen haben einen Mittelwert von 0 und eine Streuung von 1. Die Verteilung der z-Werte sieht daher immer gleich aus."
  ]
},
{
  "tag": "One-hot-encoding",
  "patterns": [
    "One-hot-encoding",
    "One hot encoding"
  ],
  "responses": [
    "One-hot-encoding: Hier wird die ganzzahlig codierte Variable entfernt und für jeden eindeutigen ganzzahligen Wert eine neue binaere Variable (Spalte)

```

```

    hinzugefügt. Die Zahlen werden durch Einsen und Nullen ersetzt, je nachdem,
    welche Spalte welchen Wert hat."
  ]
},
{
  "tag": "Label-encoding",
  "patterns": [
    "Label-encoding",
    "Label Encoding"
  ],
  "responses": [
    "Label-encoding: Bei dieser Codierung wird jeder Kategorie ein Wert von 1
    bis N zugewiesen (hier ist N die Anzahl der Kategorien für das Merkmal).
    Wichtig: die codierten kategorialen Werte bleiben kategoriale Werte, und es gibt keinerlei
    Beziehung zwischen den Werten."
  ]
},
{
  "tag": "Pivotisieren",
  "patterns": [
    "Pivotisieren"
  ],
  "responses": [
    "Häufig liegen Daten in Form von einer Liste vor. Diese Form der Daten kann
    für das trainieren eines ML-Modells nicht verwendet werden. Dafür müssen die
    Daten in einer Kreuztabelle vorliegen. Die Spalten repräsentieren die Merk-
    male und die Zeilen die Datensätze -> 2-dimensionales Array. Pivotisieren:
    Aus der Liste der Daten eine Kreuztabelle erstellen."
  ]
},
{
  "tag": "Fehlende Features",
  "patterns": [
    "Fehlende Features",
    "Features fehlen",
    "Fehlenden Features"
  ],
  "responses": [
    "Ignorieren: Löschen von Zeilen und Spalten",
    "Singuläre Imputation: Ersetzen mit Mittelwert / Median / Modus",
    "Singuläre Imputation: Ersetzen mit einer eindeutigen Kategorie",
    "Multiple Imputation: Vorhersagen der fehlenden Werte"
  ]
},
{
  "tag": "Ordinale Daten",
  "patterns": [
    "Ordinale Daten"
  ],
  "responses": [
    "Ordinale Daten haben eine logische Reihenfolge, z. B. Hoch, Mittel, Nied-
    rig"
  ]
},
{
  "tag": "Nominale Daten",
  "patterns": [
    "Nominale Daten"
  ],
  "responses": [
    "Nominale Daten haben keine bestimmte Reihenfolge, z. B. Rot, Gelb, Pink,
    Blau"
  ]
},
{
  "tag": "Kategoriale Daten",
  "patterns": [

```

```

    "Kategoriale Daten"
  ],
  "responses": [
    "Kategoriale Daten umfassen eine endliche Anzahl von Kategorien oder eindeutigen Gruppen."
  ]
},
{
  "tag": "Standard-Prozess-Modell Data Mining",
  "patterns": [
    "Standardprozessmodell Data Mining",
    "Data Mining"
  ],
  "responses": [
    "1. Business Understanding (Geschäftsverständnis): Festlegung der Ziele und Anforderungen; Ableitung der konkreten Aufgabenstellung und der groben Vorgehensweise. | 2. Data Understanding (Datenverständnis): Datensammlung bzw. erste Sichtung der zur Verfügung stehenden Daten; Ermittlung möglicher Probleme mit Datenqualität, vertraut machen mit den Daten. | 3. Data Preparation (Datenvorbereitung): Konstruktion des finalen Datensatzes für die Modellierung. Dabei werden Variablen bei Bedarf kodiert oder transformiert. Geeignete Verfahren für fehlende Daten können angewendet werden. Erfahrungsgemäß benötigt man für diese Phase ein Großteil der Zeit. | 4. Modeling (Modellierung): Anwendung geeigneter Data Mining-Verfahren, Optimierung der Parameter; gewöhnlich Ermittlung mehrerer Modelle. | 5. Evaluation (Evaluierung): Auswahl des Modells, das die Aufgabenstellung am besten erfüllt. Sorgfältiger Abgleich mit der Aufgabenstellung. | 6. Deployment (Bereitstellung): Aufbereitung und Präsentation der Ergebnisse; evtl. Integration des Modells in einen Entscheidungsprozess des Auftraggebers."
  ]
},
{
  "tag": "Geschaeftsprozesse mit KI",
  "patterns": [
    "Geschaeftsprozesse AI",
    "Geschaeftsprozesse KI"
  ],
  "responses": [
    "Big Data: Große Menge an strukturierten oder unstrukturierten Daten werden im Unternehmen täglich generiert. | Cloud - Computing: Die große Datenmenge wird am Effizientesten in einer Cloud gespeichert, verarbeitet und ausgewertet. Bei diesem Vorgehen werden Computing Ressourcen nicht mehr besessen, sondern über das Internet bezogen. | Maschinelles Lernen: In der Cloud kann auf der Grundlage der großen Datenmenge (historische Daten) ein intelligenter Algorithmus trainiert werden. Der Algorithmus ist in der Lage Muster und Gesetzmäßigkeiten in extrem großen Datenmengen zu erkennen. | Edge Computing: Trainiertes ML Modell wird auf einer Fertigungslinien-nahen Architektur (geringe Latenzzeit) betrieben. Hier können die antrainierten Gesetzmäßigkeiten auf gerade angefallenen Daten angewendet werden."
  ]
},
{
  "tag": "Digitalisierung",
  "patterns": [
    "Digitalisierung",
    "Business Intelligence"
  ],
  "responses": [
    "Unter Digitalisierung verstehen wir die Umwandlung von analogen Informationen in digitale Formate. Es geht darum, eine digitale Repräsentation von analogen Informationen, physischen Objekten oder Ereignissen zu schaffen. Die digitalisierten Informationen für sich alleine gesehen liefern noch keinen großen Mehrwert. Nur wenn es uns gelingt, die verschiedensten digitalen Informationen zusammenzuführen, also zu vernetzen, können wir einen Mehrwert

```

```

    aus den Daten ziehen. Ziel ist es die digitalen Informationen zu speichern,
    zu verteilen und maschinell zu verarbeiten."
  ],
},
{
  "tag": "Methoden Datenanalyse",
  "patterns": [
    "Datenanalyse",
    "Analytics-Reifegradmodell"
  ],
  "responses": [
    "Beschreibende Analyse = Descriptive Analytics - Bei der Descriptive Analytics (deskriptiven bzw. beschreibenden Datenanalyse) geht es um Daten aus der Vergangenheit, die dabei helfen, die Frage zu beantworten: Was ist passiert?",
    "Diagnostische Analyse = Diagnostic Analytics - In dieser Stufe können historische Daten mit anderen verglichen werden, um die Frage zu beantworten: Warum ist etwas passiert? Mit der Diagnostic Analytics (diagnostischen Analyse) ist es möglich, Ursachen und Aus- sowie Wechselwirkungen zu klären, Folgen zu analysieren und Muster zu identifizieren. Unternehmen entscheiden sich für diese Methode der Datenanalyse, um tiefgehende Einblicke in ein bestimmtes Problem zu erhalten.",
    "Vorhersagende Analyse = Predictive Analytics - Die Predictive Analytics (prädiktive Analyse bzw. Vorhersageanalyse) wird verwendet um in die Zukunft zu blicken, und versucht das Folgende herauszufinden: Was könnte bzw. wird in Zukunft passieren? Diese Methode der Datenanalyse ermöglicht es, basierend auf den Ergebnissen von deskriptiven und diagnostischen Analysen, Tendenzen zu ermitteln, Abweichungen von Normwerten frühzeitig zu erkennen und zukünftige Trends möglichst genau vorherzusagen. Die Predictive Analytics verwendet ausgefeilte Algorithmen und moderne Technologien, um zukünftige Prognosen zu erstellen. Aber obwohl diese Methode zahlreiche Vorteile mit sich bringt, ist es wichtig zu verstehen, dass Prognosen nur Schätzungen sind, deren Genauigkeit in hohem Maße davon abhängt, wie hoch die Datenqualität ist und inwiefern die Situation stabil bleibt.",
    "Präskriptive Analyse = Prescriptive Analytics - Die Prescriptive Analytics (präskriptive bzw. verordnende Analyse) ist darauf abgezielt, buchstäblich zu verordnen: Welche Maßnahmen sind zu treffen, um ein künftiges Problem zu beseitigen oder zu verhindern und das Potenzial vielversprechender Trends voll auszuschöpfen."
  ]
},
{
  "tag": "MapReduce",
  "patterns": [
    "MapReduce"
  ],
  "responses": [
    "MapReduce ist ein Verfahren, mit dem sich große strukturierte oder unstrukturierte Datenmengen mit hoher Geschwindigkeit verarbeiten lassen. MapReduce nutzt die Parallelisierung von Aufgaben und deren Verteilung auf mehrere Systeme. So wird die Aufgabe in kleine parallelisierbare Arbeitspakete zerlegt und die Ergebnisse anschließend zusammengefasst."
  ]
},
{
  "tag": "Hadoop Distributed File System",
  "patterns": [
    "HDFS",
    "Hadoop"
  ],
  "responses": [
    "HDFS ist ein hochverfügbares, verteiltes Dateisystem zur Speicherung von sehr großen Datenmengen, welches in Clustern von Servern organisiert ist. Dabei werden die Daten auf mehreren Rechnern (Nodes) innerhalb eines Clusters

```

```

    abgespeichert, das passiert in dem die Dateien in Datenblöcken mit fester
    Länge zerlegt und redundant auf den Knoten verteilt werden."
  ]
},
{
  "tag": "Distributed Computing",
  "patterns": [
    "Distributed Computing"
  ],
  "responses": [
    "Bei dem verteilten Rechnen werden mehrere Computer, die im Vergleich oft-
    mals über nur mittelmäßige Leistungsmerkmale verfügen, über ein Computer-
    Netzwerk verbunden und parallel angesteuert. Der große Vorteil der horizonta-
    len Skalierung ist der kostengünstige Einstieg, denn praktisch könnte bereits
    mit einem einzelnen Computer (Node) begonnen werden und dann nach und nach
    mit weiteren Nodes die Leistungsfähigkeit des Clusters (Verbund von Nodes =
    Clusters) linear gesteigert werden. Ungefähr linear wachsen auch die Kosten
    an, so dass diese weitaus besser planbar sind. Cluster können weitaus höhere
    Leistungen erreichen als es einzelne Server."
  ]
},
{
  "tag": "Data Warehouse",
  "patterns": [
    "Data Warehouse"
  ],
  "responses": [
    "Ein Data Warehouse speichert nur Daten die bereits strukturiert sind. Da-
    bei dient es oft als einzige Quelle der Wahrheit, da diese Plattform histori-
    sche Daten speichert, die bereinigt und kategorisiert sind. Dabei ist ein
    Data Warehouse vielseitig und für alle unterschiedlichen Anwendungsfälle ge-
    dacht."
  ]
},
{
  "tag": "Serviceebenen Cloud",
  "patterns": [
    "Serviceebenen Cloud"
  ],
  "responses": [
    "Infrastructure as a Service: liefert Rechenkapazität, Speicher und Netz-
    werkanbindungen | Platform as a Service: liefert Datenbanken und Webdienste
    auf denen eigene Anwendungen entwickelt werden können | Software as a Ser-
    vice: liefert fertige Anwendungen z. B. Email, Office, CRM"
  ]
},
{
  "tag": "Vorteile und Nachteile Software as a Service",
  "patterns": [
    "Software as a Service",
    "Software-as-a-Service"
  ],
  "responses": [
    "+ Hardware-Anforderungen: Der verwendete PC oder Laptop bedarf keiner ho-
    hen Leistungsfähigkeit, da das Computing im Rechenzentrum stattfindet. | -
    Internetzugang: Wenn die Verbindung ins Internet schwach oder nicht vorhanden
    ist, kann man nicht mit der Software arbeiten.",
    "+ Wartung: Das Aufspielen von Updates wird von Fachpersonal des Herstel-
    lers übernommen und bedeutet daher keinen Mehraufwand für den Nutzer. | - Ab-
    hängigkeit: Beendet der Anbieter der Software den Betrieb der Cloud, ist die

```

```

        Software nicht mehr einsetzbar. Unmöglich ist, bei Problemen wieder neu zu
        beginnen."
    ]
},
{
    "tag": "Vorteile und Nachteile On-Premise",
    "patterns": [
        "On-Premise",
        "On Premise"
    ],
    "responses": [
        "Vor- und Nachteile von On-Premise: | + Kauf: Durch On-Premises-Software
        entstehen keine laufenden Kosten. | + Zugriff: Da man die Software auf dem
        eigenen PC nutzt, kann man jederzeit uneingeschränkt mit dem Programm arbei-
        ten. | - Mehrarbeit: Die Wartung der Software erzeugt zusätzlichen Aufwand. |
        - Schlecht skalierbar: Lizenzen sind kostenintensiv und müssen langfristig
        gekauft werden."
    ]
},
{
    "tag": "Definition Cloud Computing",
    "patterns": [
        "Cloud Computing"
    ],
    "responses": [
        "Beim Cloud Computing geht es vereinfacht gesprochen darum IT-Ressourcen
        nicht mehr zu besitzen und selbst zu betreiben, sondern irgendwo in einem
        Kommunikationsnetz (oft im Internet) bereitgestellte IT-Kapazitäten bei Be-
        darf temporär zu nutzen - sprich als Dienstleistung bzw. Service zu beziehen.
        Merkmale: Schnelle und flexible Ressourcen-Verfügbarkeit (in Echtzeit, On-
        Demand) | Ressourcen-Zuordnung im Self Service durch den Nutzer -> Hohe Auto-
        matisierung/Standardisierung | Kurze Vertragsbindung | Abrechnung nach Nut-
        zen, variable Kosten -> Vollständige lastabhängige Skalierbarkeit -> Messbar-
        keit des IT-Verbrauchs | Keine / minimale Vorabinvestitionen -> Zugriff via
        Internet Technologie (Browser oder App)"
    ]
},
{
    "tag": "Unterschiede Big Data und Small Data",
    "patterns": [
        "Big Data Small Data",
        "Unterschiede Big Data und Small Data"
    ],
    "responses": [
        "Langlebigkeit: Small Data wird üblicherweise nach dem Projektabschluss für
        einen begrenzten Zeitraum (ca. 5-7 Jahre) aufbewahrt. Bei Big Data bleiben
        die Daten unbegrenzt gespeichert, da die Datenprojekte in weiterführende Pro-
        jekte übergehen. | Reproduzierbarkeit: Für den Fall, dass etwas fehlschlägt,
        lässt sich Small Data in der Regel voll reproduzieren. Big Data liegt in so
        vielen Formen vor und stammt aus so vielen Quellen, dass es unmöglich ist,
        bei Problemen wieder neu zu beginnen."
    ]
},
{
    "tag": "Metadaten",
    "patterns": [
        "Metadaten",
        "Daten ueber Daten"
    ],
    "responses": [
        "Metadaten sind gewissermaßen Daten über andere Daten. Metadaten enthalten
        Informationen zu weiteren Daten und fassen diese zusammen oder ergänzen diese

```

```

        (z. B. werden Fotos beim Abspeichern der Aufnahme mit solchen weitergehenden
        Infos wie Belichtungszeit, Ort und Blendenzahl versehen)."
    ]
},
{
    "tag": "Chatbot",
    "patterns": [
        "Chatbot"
    ],
    "responses": [
        "Bei einem Chatbot handelt es sich um ein technisches Dialogsystem, mit dem
        per Texteingabe oder Sprache kommuniziert werden kann. Chatbots werden häufig
        eingesetzt, um Anfragen automatisiert und ohne direkten menschlichen Eingriff
        zu beantworten oder zu bearbeiten."
    ]
},
{
    "tag": "Predictive Maintenance",
    "patterns": [
        "Predictive Maintenance",
        "Vorausschauende Wartung"
    ],
    "responses": [
        "Predictive Maintenance lässt sich mit dem Begriff vorausschauende Wartung
        ins Deutsche übersetzen. Das Verfahren nutzt Mess- und Produktionsdaten von
        Maschinen und Anlagen für die Ableitung von Wartungsinformationen. Ziel ist
        es, die Maschinen und Anlagen proaktiv zu warten und Störungszeiten zu mini-
        mieren. Im Optimalfall lassen sich Störungen vorhersagen, bevor es zu Auswir-
        kungen oder Ausfällen kommt. Die Vorteile von vorausschauender Wartung sind:
        Kenntnis über den aktuellen Zustand der Anlage, ungeplante Maschinenausfälle
        können vermieden werden, Aussendienstesätze von Mitarbeitern planbar, plan-
        bare Wartungs- Serviceintervalle und Ersatzteilemanagement, durch Datenana-
        lyse kann Leistung der Anlage verbessert werden, höhere Produktivität wird
        erreicht. Die Strategien von Predictive Maintenance sind: Regressionsmodelle
        zur Vorhersage der verbleibenden Nutzungsdauer, Klassifikationsmodelle zur

```

```

    Vorhersage von Fehlern innerhalb eines bestimmten Zeitfensters, Anomales Verhalten Kennzeichen."
  ],
},
{
  "tag": "KI im Qualitätsmanagement",
  "patterns": [
    "KI Qualitätsmanagement",
    "AI Qualitätsmanagement"
  ],
  "responses": [
    "Rückläuferanalyse",
    "Kundenzufriedenheit",
    "Fraud Detection"
  ]
},
{
  "tag": "Teilbereiche KI",
  "patterns": [
    "Teilbereiche KI",
    "Teilbereiche AI"
  ],
  "responses": [
    "Mustererkennung: Bildern, Text, Sprache",
    "Vorhersagen: Big Data, Predictive Analytics",
    "Knowledge Representation: Expert / Knowledge Systems",
    "Planung, Terminierung, Optimieren",
    "Natural Language Processing: Kommunikation, Lesen und Verstehen von Sprache",
    "Robotics: Anwendung, Übertragung, Aktion, Navigation, Lokalisierung, Zuordnung",
    "Machine Learning: Clustering, Classification, Regression"
  ]
}
]
}

```

## Ehrenwörtliche Erklärung

Ich versichere, dass ich die vorliegende Bachelorarbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben, sowie wörtliche und sinngemäße Zitate gekennzeichnet habe.

Ulm, den 24.07.2023

Ort, Datum

  
\_\_\_\_\_  
Unterschrift