Entwicklung eines Chat-Bots für das Argumentationssystem D-BAS

Bachelorarbeit

von

Fabian Correnz

aus

Viersen

vorgelegt am

Lehrstuhl für Rechnernetze
Prof. Dr. Martin Mauve
Heinrich-Heine-Universität Düsseldorf

Dezember 2020

Betreuer:
Markus Brenneis, M. Sc.

Zusammenfassung

Im Rahmen der vorliegenden Arbeit wurde für das Online-Argumentationssystem D-BAS ein Frontend in Form eines Telegram-Chatbots realisiert.

Bei der Entwicklung des Chatbots wurde – ausgehend von einer elementaren Kommandostruktur, die zunächst zum Testen der Schnittstelle benutzt worden war – in mehreren Iterationsschritten unter Einbindung von fünf Testanwendern eine Kombination von Freitexteingaben und den in Telegram zur Verfügung stehenden Buttons realisiert (hybrider Ansatz). Während die Freitexteingaben hauptsächlich zur Eingabe von Argumenten für die Backend-Funktionen genutzt werden, werden die Buttons insbesondere zur Repräsentation der sprachlich nicht immer einfachen Argumentationslogik genutzt.

In der Aufgabenteilung zwischen Frontend und Backend wurden elementare sprachliche Vorverarbeitungen, die nicht zu den Formulierungen der Argumente gehören, im Frontend angesiedelt, während bei den inhaltlichen Argumenten die Formulierungen und grammatikalischen Umstellungen des Backends übernommen wurden.

Insgesamt hat sich der hybride Ansatz als sinnvolle Möglichkeit erwiesen, eine flüssige und einfache Kommunikation mit dem Benutzer zu erreichen, ohne die Präzision der mathematischen Logik der Argumentation zu verlieren.

Weiterhin wurden im Verlauf verschiedene Techniken zur textlichen Anpassung der Kommunikation an das Medium des Chatbots erörtert bzw. prototypisch implementiert. Zu diesen Techniken gehören die Dekomposition komplexerer Auswahlmöglichkeiten in ja/nein Fragen, die Einsetzung der Argument-Satzteile in die logischen Auswahloptionen sowie das Auslassen von Formulierungen, die aus dem Chatverlauf unmittelbar noch ersichtlich sind.

Um die Möglichkeiten dieses Ansatzes jedoch voll auszuschöpfen, wäre eine Erweiterung der seitens des Backends produzierten Satzphrasen im Hinblick auf eine Chatbot-Schnittstelle sinnvoll.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die vorliegende Arbeit erfolgt im Kontext des Themas „E-Particiation“. Generell versteht man darunter „die Online-Beteiligung von Bürgern an Entscheidungsprozessen in Politik und Verwaltung“ [Gro18, Kap. 1.1].

Zur Unterstützung dieses Prozesses wurde an der Heinrich-Heine-Universität Düsseldorf das „Dialog-Based Argumentation System“ (D-BAS) entwickelt, das die Argumente im Rahmen einer solchen Bürgerbeteiligung aufnimmt, strukturiert anlegt sowie wiedergeben kann. Ziel dabei ist es, bei einer Entscheidungsfindung zu unterstützen, nicht aber selbst eine Entscheidung oder einen Vorschlag zu liefern [Kra+18].

D-BAS ist in Form seines Sourcecodes sowie eines laufenden Benutzerinterfaces öffentlich zugänglich. Anhand eines ersten Feldtests mit 318 Teilnehmern sowie eines zweiten mit 142 Teilnehmern wurde empirisch belegt, dass das System geeignet ist, Diskussionen einer großen Gruppe von Studenten sinnvoll zu erfassen und zudem bei einer Entscheidungsfindung hilfreich sein kann[Ebb19].

Beim ursprünglichen Frontend von D-BAS handelt es sich um eine Website zur strukturierten Fragestellung und Beantwortung. Diesem Frontend wurde mit Jebediah[MEM18], einem Socialbot basierend auf dem Google DialogFlow Framework[1], bereits ein weiteres Frontend hinzugefügt. Ferner liegen mit discuss[MKM17], decide[EM20] und deliberate[BM20] weitere Arbeiten vor, bei denen D-BAS im Rahmen weiterer Frontends angebunden wurde. Ziel der vorliegenden Arbeit ist es, die bestehenden D-BAS Frontends um ein weiteres Frontend,

[1] https://cloud.google.com/dialogflow/docs, abgerufen am 17.12.2020

einen Telegram-Chatbot, zu erweitern.

Grundsätzlich ermöglicht die bestehende Oberfläche für alle gängigen Geräte einen Zugang zu D-BAS und erschließt damit rein technisch die gesamte potenzielle Anwenderschaft. Jedoch dürften über einen zusätzlichen Telegram-Chatbot mehr Nutzer erreicht werden, insbesondere weil Chatbots sich momentan großer Beliebtheit und schnell wachsender Verbreitung erfreuen.

Ein Telegram-Chatbot stellt mit seinen andersartigen Möglichkeiten, aber auch Einschränkungen, ein deutlich anderes Medium der Kommunikation als das ursprüngliche Web-Frontend dar. Dessen Eignung für D-BAS soll anhand der vorliegenden Arbeit genauer untersucht werden.

Im weiteren Verlauf der Arbeit werden zunächst einige Grundlagen erläutert. Danach wird aufgezeigt, welche Anforderungen an den Bot gestellt wurden und wie diese umgesetzt wurden. Anschließend werden die Implementierung, der Systemaufbau und die -architektur, Designentscheidungen sowie einige noch ungelöste Themen und Erweiterungsmöglichkeiten aufgezeigt. Zudem werden die Funktionen anhand einiger Beispiele erläutert.

Kapitel 2

Hintergrund

Das Wort „Chatbot“ setzt sich aus zwei Worten zusammen. Zum einen „chat“, englisch für „plaudern“ und zum anderen „bot“, was sich von „robot“ ableitet, englisch für „Roboter“. Im Allgemeinen ist somit ein System gemeint, das die Fähigkeit besitzt, mit einem Menschen zu kommunizieren und gewisse Prozesse eigenständig zu erledigen. Als Eingabe erhält das System eine Aufgabe mittels natürlicher Sprache (in geschriebener oder gesprochener Form), welche daraufhin bearbeitet wird und eine gewünschte Ausgabe in natürlicher Sprache zurück gibt [SDP20, S. 3 ff.].

Die Geschichte der Chatbots reicht viel weiter zurück, als man zunächst annimmt. Bereits in den 60er-Jahren wurden erste Versuche getätigt. Der Chatbot „Eliza“ wurde in den Jahren 1964-1966 von Weizenbaum programmiert und gilt als der Erste seiner Art. In Abbildung 2.1 ist eine Übersicht einiger Meilensteine zu sehen. Mit fortschreitendem Entwicklungsstand und einem immer größer werdenden Interesse an Messengerdiensten finden Chatbots in der heutigen Zeit immer mehr Anwendungsgebiete. So werden – wie auch in der vorliegenden Arbeit – beispielsweise immer mehr Prozesse in Bots verlagert, um die Bedienung zu erleichtern.

2.1 D-BAS

D-BAS (Dialog-Based Argumentation System, [Kra+18]) ist ein System zur Unterstützung einer zeitversetzten Diskussion zwischen mehreren Personen. Zu einem Thema gibt ein Be-

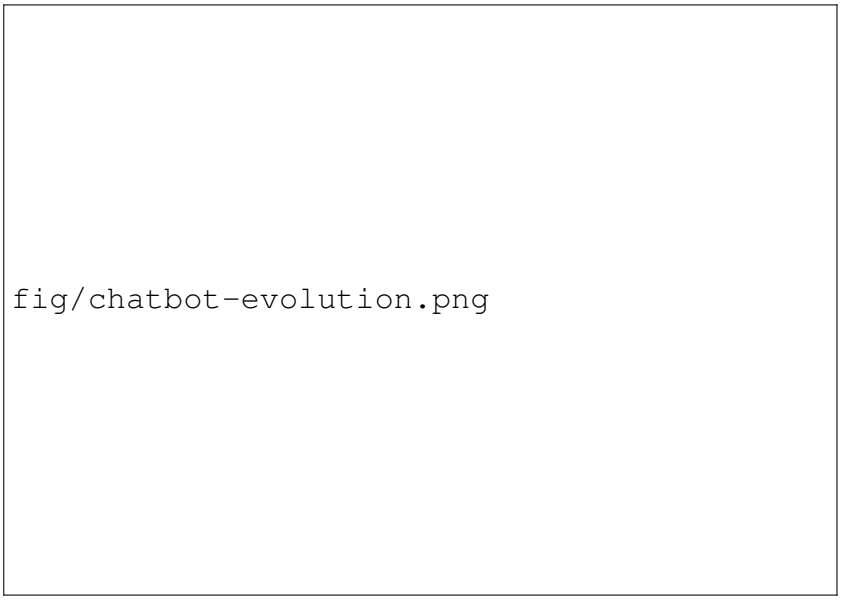


Abb. 2.1: Evolution der Chatbots [entnommen aus SDP20]

nutzer seine Meinung ein. Andere Nutzer können dann zeitversetzt auf diese Meinung reagieren. Entweder, indem sie diese unterstützen und mit Argumenten untermauern, oder indem sie Gegenargumente finden. Die Inhalte solcher Diskussionen können dann Entscheidern in wohlstrukturierter Form präsentiert werden und somit dazu beitragen, die Zügigkeit und Qualität von Entscheidungen zu verbessern.

2.1.1 D-BAS-Frontend

Abbildung 2.2 zeigt ein Beispiel der bestehenden D-BAS Anwenderoberfläche. Hierbei werden insbesondere die folgenden Techniken verwendet, die in den bisherigen empirischen Arbeiten erprobt wurden:

- Der Anwender wird mit nur einem Argument gleichzeitig konfrontiert.
- Das zuletzt angegebene Argument wird in Form einer Sprechblase noch einmal wiedergegeben.
- Das Erkennen der Bezüge zwischen einzelnen Bestandteilen in dieser Sprechblase wird durch farbliche Markierung sowie weiße Unterlegung logischer Bausteine der Argu-

mentation und ihrer Beziehung erleichtert.

- Das System sucht auf Basis der Antwort des Anwenders das nächste Argument aus, mit dem Ziel, den Anwender weiter im Dialog zu halten und weitere Argumente auszutauschen.

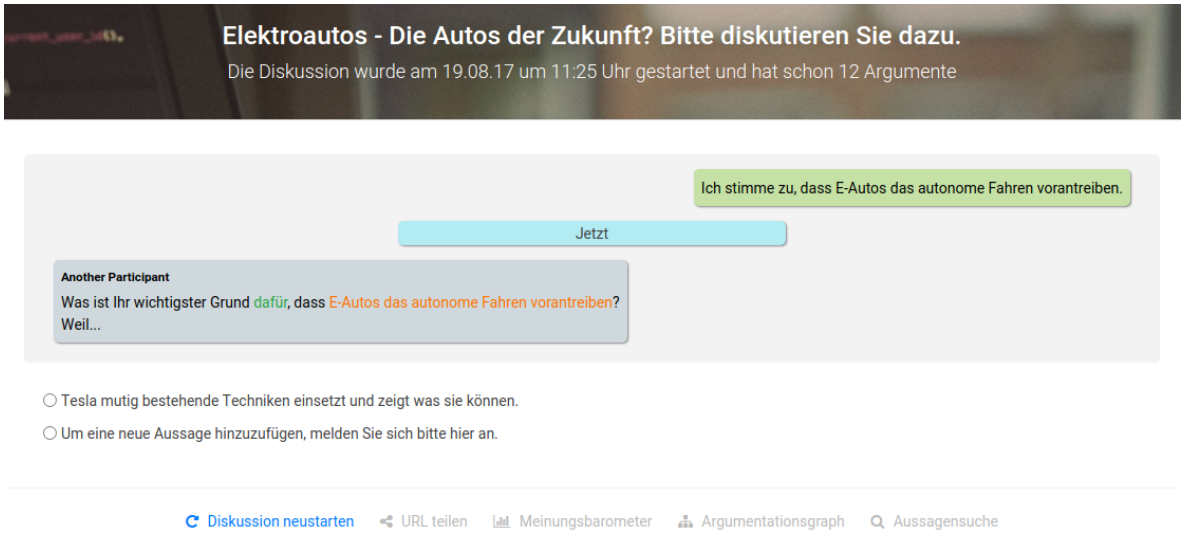


Abb. 2.2: Bestehende Anwenderoberfläche von D-BAS

2.1.2 D-BAS-Backend

D-BAS erstellt aus den Eingaben einen Argumentationsgraphen, welcher ermöglicht, alle Argumente strukturiert anzulegen und wiederzugeben [SM19]. Die Funktionen des D-BAS-Backends lassen sich wie folgt zusammenfassen:

- Verwalten eines Graphen, der das Wissen aus den Chats darstellt.
- Entgegennahme von Anfragen zu diesem Graphen.
- Erstellung von Antworten zu diesem Graphen.

Der Graph der Wissensrepräsentation sieht strukturell wie in Abbildung 2.3 dargestellt aus:

Ausgangspunkt jedes Argumentationsgraphen ist eine Thematik (im Graphen der einzelne graue Punkt im Zentrum), welcher man verschiedene Positionen (blaue Punkte) zuordnen

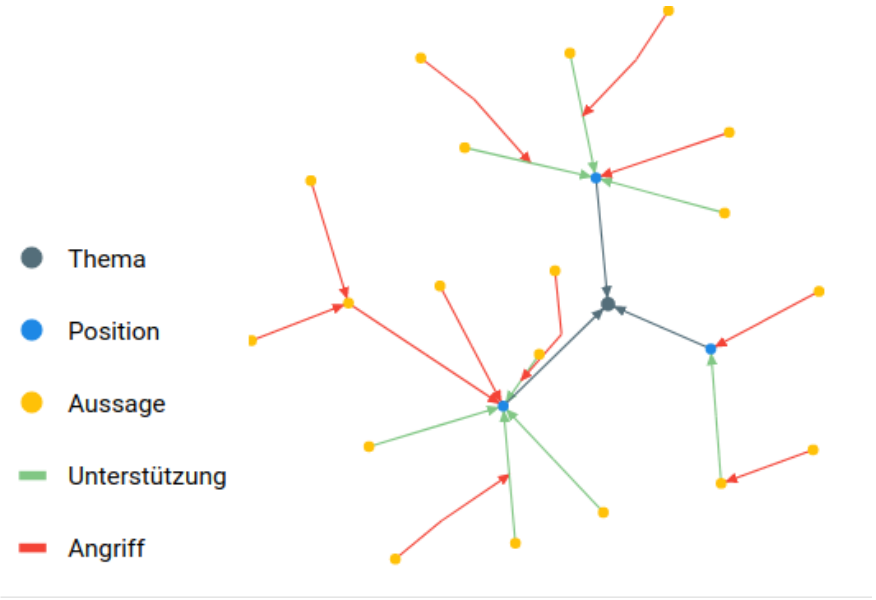


Abb. 2.3: Visuelle Darstellung eines Argumentationsgraphen von D-BAS

kann (graue Kanten). Diesen Positionen können daraufhin Aussagen (auch Prämissen, gelbe Punkte) zugeordnet werden, die entweder als zustimmend (grüne Kante), oder als ablehnend (rote Kante) klassifiziert werden. Weiterhin können Aussagen durch andere unterstützt werden (ebenfalls grüne Kanten) oder ihnen kann widersprochen werden (rote Kanten). Die Darstellung, dass zwei Aussagen in einer bestimmten Relation zueinander stehen, kann wiederum in Frage gestellt werden (im Graphen dargestellt als rote Kante, die auf eine andere Kante zeigt). D-BAS verknüpft auf Grund von Nutzerangaben die einzelnen Aussagen und Positionen mit entsprechend farblich markierten Kanten und speichert auf diese Weise die zum Teil sehr komplexen Strukturen. [Kra+18, S. 335 Figure 3 a)]

2.1.3 D-BAS-Inhalte

In D-BAS werden Diskussionen grundsätzlich in die folgenden Grundbausteine unterteilt [Kra+18]:

Thema („issue“): Beschreibt, worum es in der Diskussion geht (z.B: „Verbesserung des Informatik-Studiengangs“).

Aussage („statement“): Aussagen sind die kleinsten Bausteine in D-BAS. Sie können als

wahr oder falsch betrachtet werden (z. B.: „Der Zulauf zum Informatik-Studiengang ist zu hoch.“). Es gibt verschiedene Formen der Aussagen:

- Position („position“)
- Prämisse („premise“)
- Schlussfolgerung („conclusion“)

Position („position“): Positionen sind vorschreibende (präskriptive) Aussagen, also Aussagen, die eine bestimmte Aktion empfehlen oder dazu auffordern (z. B. „Es sollte eine Zulassungsbeschränkung für Informatik eingeführt werden.“).

Argument („argument“): Zwei Aussagen (Prämisse und Schlussfolgerung) bilden ein Argument (z. B. „Man ist beim Studium nicht auf sich alleine gestellt, weil es Tutor*innen gibt, die unterstützen und helfen.“).

Aus diesen Grundbausteinen lassen sich nun einfachere oder komplexere Argumentationen zusammen setzen. In einem einfachen Fall bestehen diese beispielsweise aus:

- einer Position (wie z. B. „Es sollte eine Zulassungsgrenze eingeführt werden,“),
- einer Prämisse, (wie z. B. „weil viele Studierende sich einschreiben, ohne die notwendigen Kompetenzen zu besitzen.“), welche die Position begründet,
- und einer zustimmenden oder ablehnenden Beziehung zwischen Position und Prämisse (Beziehungsrelation).

Die sprachliche Repräsentation einer Stellungnahme zu einer Argumentation ist nicht immer einfach. Allgemein werden die Reaktionen als Angriffe („attacks“) bezeichnet. Im D-BAS kann z. B. auf die Argumentation „Wir sollten keine Reinigungskraft engagieren, weil es viel Geld kostet.“ wie folgt reagiert werden (siehe Abbildung 2.4):

Wiedersprechen gegen die Prämisse („undermine“): Darstellung in der bestehenden Oberfläche: „Ich halte die Aussage für falsch und möchte widersprechen.“. Dargestellt wird dies in diesem Beispiel durch die rote Kante auf die Aussage „es kostet viel Geld“.

Widersprechen der Relation zw. Prämisse und Schlussfolgerung („undercut“): Darstellung in der bestehenden Oberfläche: „Ich halte deren Aussage für richtig, aber sie stützt nicht deren Behauptung.“. Dargestellt wird dies durch den roten Pfeil auf die Kante ausgehend von der Aussage „es erzeugt Freizeit, die man sich sonst nicht erkaufen kann“.

Entkräftung einer Argumentation („rebut“): Darstellung in der bestehenden Oberfläche: „Ich halte deren Behauptung für richtig und sie unterstützt auch deren Standpunkt, aber ich möchte meinen Standpunkt trotzdem verteidigen.“ (in diesem Graphen nicht abgebildet, springt zurück zu der im Vorhinein verwendeten Aussage).

Unterstützung einer Argumentation („support“): Darstellung in der bestehenden Oberfläche: „Ich halte die Aussage für richtig und finde sie überzeugend.“ (ebenfalls nicht abgebildet, führt zum Ende des Themas, wenn keine anderen Verhalte abgefragt werden).

Diese kurze exemplarische Darstellung zeigt, dass D-BAS hoch komplex ist und für eine Einordnung immer ein Kontext gegeben sein muss. Eine detaillierte Darstellung der mathematischen Hintergründe befindet sich in [Kra+18].

2.2 Telegram

Telegram ist ein kostenloser Instant-Messenger, welcher auf Smartphones, Tablets, PCs und Smartwatches benutzt werden kann und seit 2013 auf dem Markt ist. Der Service zählt mit ca. 400 Mio. Nutzern zu den großen mobilen Messaging Plattformen weltweit. Mit ca. 7,8 Mio. Nutzern handelt es sich in Deutschland um einen der größten Messenger (an Platz 5 nach WhatsApp, Facebook Messenger, Apple Messages und Skype)[Mesb].

Weiterhin ist es jedem Telegram-Nutzer seit 2015 möglich, kostenfrei einen Chatbot zu erstellen, ohne Einschränkungen. Zwar hat WhatsApp beispielsweise deutlich höhere Nutzerzahlen, ein Chatbot ist allerdings eine kostenpflichtige Premiumkomponente (lediglich öffentliche Vorschau mit eingeschränkten Funktionen kostenfrei zugänglich[1]). Auch eine Unterstützung zu Linux Systemen ist bei WhatsApp und anderen Messengern nicht vorhanden.

[1] https://developers.facebook.com/docs/whatsapp, abgerufen am 12.12.2020

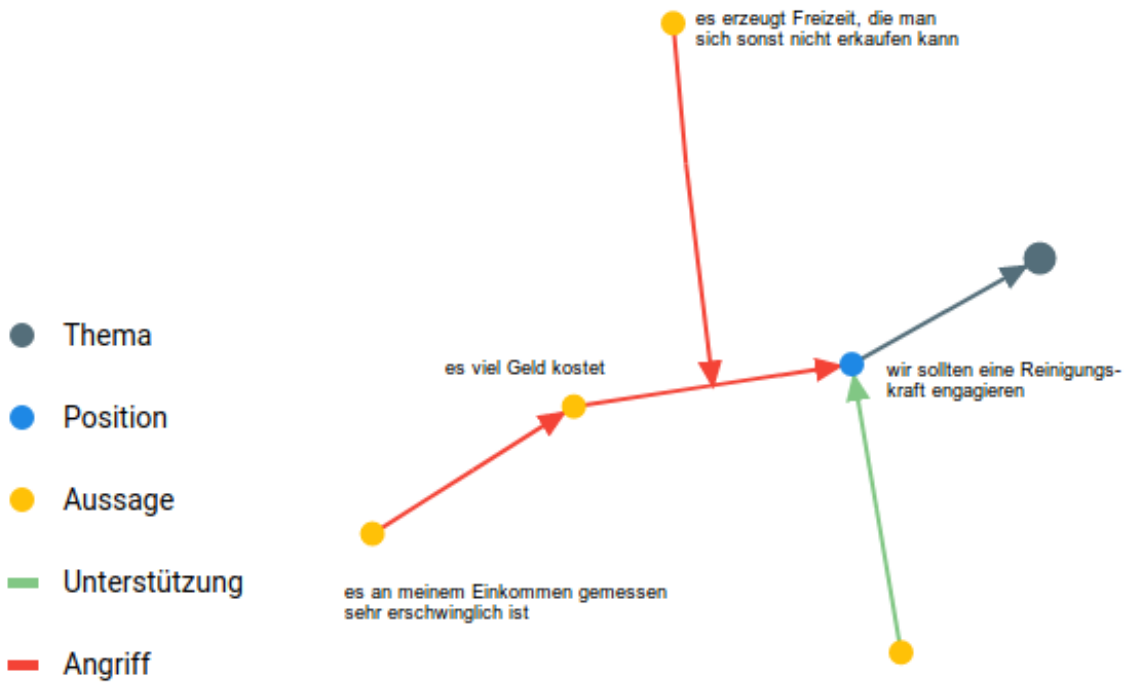


Abb. 2.4: Darstellung möglicher „attacks“

Aus diesen Gründen ist die Wahl auf Telegram gefallen.

Über Telegram lassen sich Textnachrichten, Bilder, Videos, Sprachnachrichten und Dateien an andere Nutzer versenden. Außerdem bietet Telegram eine Sprach- und Videotelefonie-funktion an.

Eine App bzw. ein Programm wird für alle geläufigen Betriebssysteme (Android, Firefox OS, iOS, Linux, macOS, Ubuntu Touch, Windows, Windows Phone) angeboten.

2.3 Verwandte Arbeiten

Das D-BAS-Projekt ist Teil einer größeren Entwicklungsreihe mit mehreren weiteren Komponenten, die aktuell auch noch erweitert werden. Dazu gehören insbesondere „Jebediah“ [MEM18] (verbessert die Benutzererfahrung, indem es einen Agenten für soziale Netzwerke bereitstellt, der die Verarbeitung natürlicher Sprache unterstützt [SM19]), „discuss“ [MKM17]

(ermöglicht die Einbettung des Interfaces in beliebige Websites [SM19]), das im zweiten Feldversuch verwendete „decision“ (hilft bei der Entscheidungsfindung mit D-BAS)[EM20], sowie „deliberate“ (einer Full-Stack-Web-Application zum Austausch von Argumenten)[BM20].

Über die Verarbeitung natürlicher Sprache kann Jedebiah die D-BAS-Funktionen in einem Messengerdialog darstellen. Da die Auswahl verschiedener Möglichkeiten in einem Freitextansatz schwierig darzustellen ist, haben sich die Autoren dazu entschieden, dass Jedebiah den Nutzer durch den Dialog leitet, um so die nötige Tiefe in der Diskussion zu erreichen. Dabei ist Jebediah gleichzeitig flexibel genug, um auf die Reaktionen des Nutzers einzugehen.

Eine ebenfalls vergleichbare Arbeit stammt von Chalaguine und Hunter[Pra+20]. Hierbei handelt es sich um eine sehr ähnliche Arbeit, wobei im Kern auf die Verbesserung der Überzeugungskraft des Bots eingegangen wird. Als Grundlage wird ein Argumentationsgraph (vergleichbar mit den Graphen, die D-BAS erstellen kann) verwendet. In der Arbeit wurden zwei Varianten des Bots entwickelt. Einer nutzt – falls vorhanden – vorstrukturierte Gegenargumente aus dem Argumentepool (Basis-Bot), der andere analysiert das Argument des Nutzers und gibt – falls vorhanden – ein inhaltlich passendes Gegenargument zurück (strategischer Bot). Ausschlaggebend für den Grad der Überzeugung ist hierbei die Größe des verwendeten Argumentgraphen. Je größer der Graph, desto mehr potentielle Gegenargumente sind vorhanden, um den Nutzer zu überzeugen.

Bei der Umsetzung der beiden Bots wurden zwei verschiedene Techniken verwendet. Der Basis-Bot verwendet einen „menu-based“ Ansatz. Das bedeutet, dass der Nutzer eine Liste möglicher Antworten vom Bot vorgegeben bekommt, aus der er seine Antwort auswählen kann. Der „free-text“ Ansatz hingegen, welcher beim strategischen Bot verwendet wird, verarbeitet die Texteingaben des Nutzers, indem analysiert wird, welche Position dieser eingenommen hat, um entsprechend zu reagieren. Im Endeffekt hat der Freitextansatz bessere Ergebnisse in den Punkten Verständnis und Überzeugungskraft erzielen können [Pra+20, siehe Tabellen S. 8 und 10]. Insofern stellt der in der vorliegenden Arbeit verfolgte Ansatz eines Chatbots für D-BAS eine schlüssige Erweiterung des bestehenden Backends dar.

Kapitel 3

Anforderungen und Chatbot-Design

Im Folgenden werden die Anforderungen, welche an den Bot gestellt werden, grob skizziert. Hierbei wird zunächst kurz abgewogen, welche Funktionen von D-BAS implementiert werden sollen, bevor die Zielgruppe und -plattform analysiert wird.

3.1 Funktionale Anforderungen

Um die Eignung von Chatbots als Oberfläche für D-BAS evaluieren zu können, soll für einige Kernfunktionen von D-BAS ein Chatbot realisiert werden. Zu den Kernfunktionen, die hierfür erforderlich sind, gehören:

- Auswahl eines Themas
- Auflistung der bereits vorhandenen Themen und Argumente
- Untermauern bzw. Entkräften der vorhandenen Argumente
- Hinzufügen neuer Themen und Argumente
- Einloggen eines registrierten Nutzers

Während diese Kernfunktionen einen kompletten Chatverlauf abbilden können, gehören zu D-BAS noch weitere Funktionen, wie z. B. die Administration des zuvor skizzierten Argu-

mentationsgraphen oder ein Moderationssystem. Die Eignung dieser anderen Funktionsbereiche für einen Chatbot wäre zumindest anders zu beurteilen als der hier zunächst untersuchte typische Chatverlauf und ist daher nicht Bestandteil der prototypischen Implementierung.

3.2 Vermeidung unnötigen Moderationsaufwands

Es gibt in D-BAS ein verteiltes Moderationssystem, bei dem Anwender in Abhängigkeit ihrer im System erworbenen „Reputation“ neue Aussagen bewerten, verdächtige Aussagen melden, oder darüber abstimmen können, ob Aussagen gelöscht werden sollen. Je mehr Reputation ein Nutzer hat, desto mehr diesbezügliche Rechte bekommt er.

An einigen Stellen kann der Chatbot für die Moderatoren unnötigen Aufwand ersparen. Wenn beispielsweise in einer Argumentation Inhalte eingegeben werden, die allgemein akzeptierten Ansprüchen an Etikette zuwider laufen, kann das Frontend eine Speicherung dieser Inhalte unter Information des Anwenders direkt verhindern, ohne in die Logik der eigentlichen Argumentation einzugreifen.

3.3 Zielgruppe

Bei den in der Einführung erwähnten Feldversuchen wurde gezeigt, dass D-BAS für Informatikstudenten ohne Schulung verständlich und nützlich ist. Es ist jedoch fraglich, ob dies für Nutzer aus allen Bereichen und Altersklassen gilt. Durch die zusätzliche Schnittstelle, den Telegram-Chatbot, können Personen aus allen Bereichen und Altersgruppen über einen weiteren Kanal erreicht werden. Insbesondere ist der Umgang mit Messengern bei der jüngeren Generation stark verbreitet.

Der Erfolg von D-BAS als Unterstützungssystem zur Entscheidungsfindung steht und fällt mit der Bereitschaft vieler Bürger, sich an entsprechenden Diskussionen zu beteiligen. Daher ist es besonders wichtig, die Bedienung der Oberfläche einfach und verständlich zu gestalten. Bestmöglich sollte die Bedienung eine motivierende Wirkung haben.

3.4 Ziel-Plattform

Für die Realisierung des Chatbots wurde der Messenger-Service Telegram ausgewählt. Betrachtet man Abbildung 3.1, so sieht man deutlich, dass der Großteil der Nutzung an den Smartphones erfolgt. Dies sollte bei der weiteren Implementierung beachtet werden.

Abb. 3.1: Messengernutzung nach Endgeräten 2020 [Mesa]

Kapitel 4

Implementierung

Im nachfolgenden Kapitel werden die Einzelheiten der Implementierung ausführlich dargestellt. Bevor diese genauer erläutert werden, wird der bisherige Nachrichtenfluss von D-BAS sowie der vom Chatbot verwendete Nachrichtenfluss aufgezeigt. Als Arbeitstitel wurde für den Chatbot der Name „TIBAS“ (**T**elegram **I**nterface for Dialog-**B**ased Online **A**rgumentation **S**ystem) gewählt.

4.1 Nachrichtenfluss

In Abbildung 4.1 ist dargestellt, wie der Nachrichtenfluss zwischen dem Nutzer, den Telegram-Servern und dem Bot funktionieren. Wenn Alice an ihrem Smartphone eine Nachricht über die Telegram-App an den Bot sendet, bzw. Bob eine Nachricht in der Telegram-Desktopanwendung verschickt, so werden diese Nachrichten zunächst mit einer Server-Client Verschlüsselung an die Telegram-API gesendet. Diese kann die Nachrichten auf Grund der mitgesendeten ChatID für den entsprechenden Chat in der Cloud abspeichern.

Der Bot fordert die Nachricht von Alice bzw. Bob von den Telegram-Servern an. Nachdem er die Nachricht verarbeitet hat, stellt er eine Anfrage an den lokalen D-BAS Server und fordert neue Daten an (in diesem Fall die Antwort auf Alices und Bobs Nachrichten). Hat er die Daten von der lokalen Instanz erhalten, verarbeitet er diese in eine Antwortnachricht und schickt sie zurück zur Telegram-API. Erneut kann diese die Nachricht mittels der mitgeschickten ChatID zu Bobs bzw. Alices Chat zugeordnet und in der Cloud abgespeichert

Abb. 4.1: Nachrichtenfluss zwischen den Nutzern, Telegram, D-BAS und dem Bot

werden. Die Anwendungen von Bob und Alice bemerken nun, dass eine neue Nachricht vorliegt und rufen diese ab. So schließt sich der Kreis des Nachrichtenflusses und beginnt von vorne, sobald die beiden eine neue Nachricht an den Bot senden.

Chris und Daina nutzen hingegen den aktuellen Stand von D-BAS. Sie kommunizieren beide direkt über einen Browser mit den D-BAS-Servern. Die Implementierung der vorliegenden Arbeit betrifft den rot gekennzeichneten Teil der Grafik. In den folgenden Kapiteln werden die aktuell realisierten Komponenten sowie deren mögliche Erweiterungen erläutert.

4.2 Eingabeverarbeitung

Der Telegram-Chatbot nimmt Eingaben eines Nutzers entgegen, interpretiert diese und extrahiert daraus die Intention des Anwenders sowie weitere Informationen. Auf Basis dieser Intention entscheidet er, ob die eingegebenen Informationen in einen Aufruf des D-BAS Backends umgesetzt werden können, oder ob dazu noch weitere Rückfragen an den Anwender erforderlich sind.

Sobald alle erforderlichen Rückfragen geklärt sind, stellt der Bot eine entsprechende Anfrage an D-BAS und bereitet das zurück gelieferte Resultat wieder für den Anwender auf.

Bei der Kommunikation mit dem Anwender macht der Bot von zwei Dialogtechniken Gebrauch: Freitexteingaben und Menü-Buttons. Dabei kann der Anwender jederzeit Freitexte eingeben, auch wenn eine Button-Auswahl vorgelegt wurde. Beispielsweise kann er mit /start

in jeder Situation zurück zur Themenübersicht navigieren.

In Situationen, in denen beide Eingabetechniken sinnvoll als Anwenderintentionen interpretiert werden können, analysiert der Chatbot die Intention des Anwenders und reagiert entsprechend. Beispielsweise interpretiert der Bot bei der Themenauswahl einen Freitext, der nicht mit den vorgeschlagenen Themen übereinstimmt, als neuen Themenwunsch.

Tabelle 4.1 gibt einen Überblick über die seitens des Bots verarbeiteten Freitextnachrichten.

Eingabe	**Reaktion des Bots**
/start (überall möglich)	Springt (zurück) zur Themenübersicht.
/login (überall möglich)	Startet die Login Prozedur, stellt anschließend den vorherigen Zustand wieder her.
Freitexteingabe (allgemein)	Prüft, ob Schimpfwörter enthalten sind oder die maximal erlaubte Länge von 142 Zeichen überschritten wurde (nachdem Füllwörter eliminiert wurden).
Freitexteingabe (in der Themenübersicht)	Prozedur, um ein Thema hinzuzufügen wird gestartet (Titel, Zusammenfassung und Beschreibung werden abgefragt).
Freitexteingabe (in der Übersicht der Positionen eines Themas)	Prozedur, um eine Position hinzuzufügen wird gestartet (Position und Prämisse werden abgefragt). Springt anschließend in die neu angelegte Position.
Freitexteingabe (wenn letzte Botnachricht „Weil...“ bzw. „Because...“ war)	Prozedur, um neue Prämisse hinzuzufügen wird gestartet. Wählt die neue Prämisse als Antwort aus und wartet auf den nächsten Schritt von D-BAS.
Freitexteingabe (außerhalb einer Prozedur in allen oben nicht genannten Kontexten)	Weist den Nutzer darauf hin, dass an dieser Stelle nichts hinzugefügt werden kann. Der Kontext (evtl. Buttons sowie die aktuelle URL des Nutzers) bleiben erhalten.

Tabelle 4.1: Reaktionen des Bots auf Freitexteingaben

4.3 Architektur des Chatbots

Wie in Abbildung 4.2 zu sehen ist, kann man den Chatbot grob in die zwei Hauptkomponenten „Chat-Kontext“ und „Backend-Kontext“ unterteilen, welche jeweils kleinere Komponenten enthalten. Deren Aufgaben werden im Folgenden genauer erläutert. Der „Parser“ bereitet eine eingehende Nachricht auf deren Weiterverarbeitung vor und die „Ausgabeerstellung“ bereitet das Versenden einer Antwortnachricht vor.

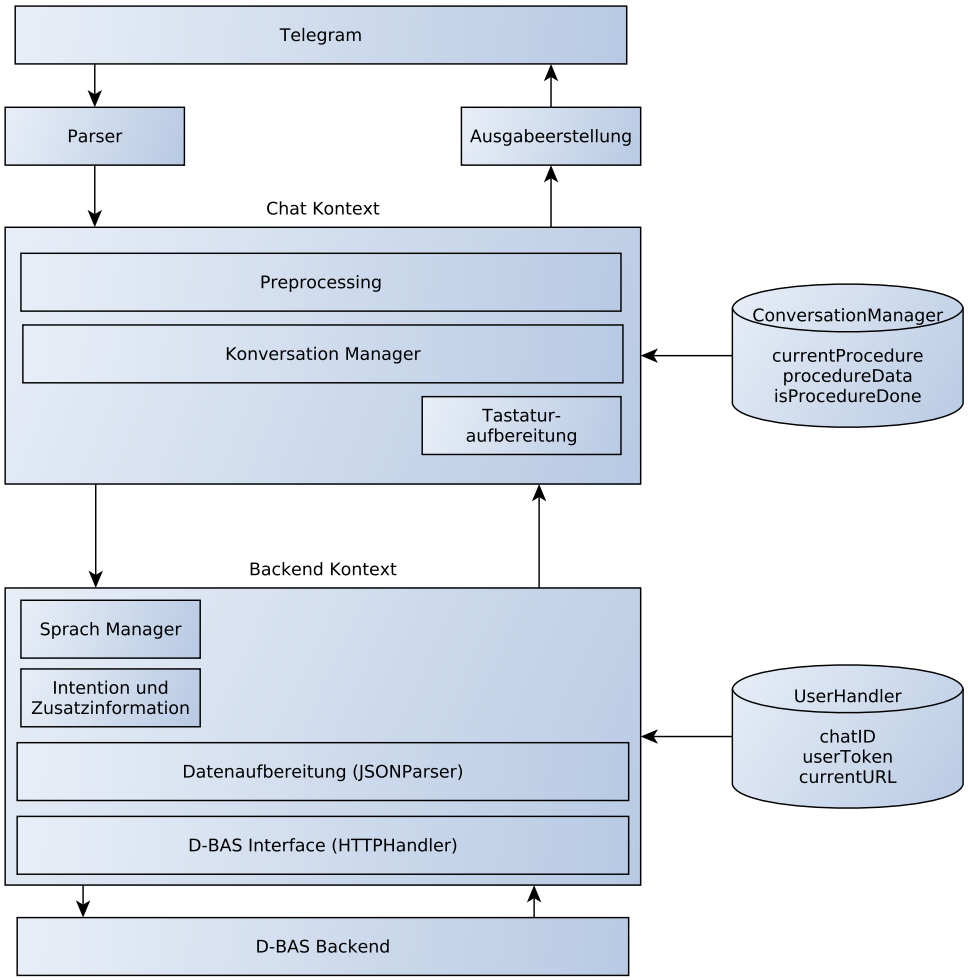


Abb. 4.2: Architekturüberblick der einzelnen Komponenten des Chatbots

4.3.1 Chat-Kontext

Die Aufgabe des Chat-Kontext ist es, eine eingehende Nachricht auf verschiedene Fälle zu untersuchen. So gibt es einige Nachrichten, bei denen der Bot eigenständig antworten kann, ohne eine Antwort von D-BAS anzufordern. Im Chatkontext sind im Einzelnen die folgenden Komponenten realisiert:

Preprocessing: Dient zur Prüfung und ggf. Aufbereitung der eingegebenen Texte für die spätere Weitergabe. Zunächst wird die eingehende Nachricht auf *Füllwörter* untersucht. Sind Füllwörter vorhanden, werden diese ersatzlos gestrichen, da sie nicht notwendig sind, um die Aussage abzubilden. Beispielsweise werden „eigentlich“, „meiner

Meinung nach“ und „in my opinion“ gestrichen. Weiterhin wird die Nachricht auf Wörter untersucht, die allgemeinen Standards der *Etikette* widersprechen. Ist ein solches Wort enthalten, so fängt der Bot dies ab und verlangt eine erneute Eingabe vom Benutzer, ohne hierbei den Gesprächskontext zu ändern. Beispielsweise werden „Idiot“ und „crap“ vom Bot abgewiesen. Außerdem wird die *Länge* der Nachricht überprüft. Überschreitet sie eine Länge von 142 Zeichen, nachdem eventuell vorhandene Füllwörter gelöscht wurden, so weist der Bot diese zurück. Dadurch wird zum Einen gewährleistet, dass alle Themen und Argumente in den Buttons dargestellt werden können. Zum Anderen sind kurze Formulierungen für das Verständnis oft besser und benutzerfreundlicher, was besonders unter dem Aspekt der Hauptnutzung von Messengern auf Smartphones von Bedeutung ist.

Conversation-Manager: In der `ConversationManager`-Klasse sind die bereits erwähnten Prozeduren zur Zusammensetzung mehrerer Dialogschritte aus dem Anwenderdialog zu einem D-BAS Aufruf realisiert. Möchte der Nutzer zum Beispiel ein neues Thema hinzufügen, so ist dem Bot bekannt, dass der Nutzer die Parameter „title“, „summary“ und „description“ angeben muss. Diese werden vom Konversation Manager nacheinander abgefragt und zwischengespeichert, bevor alle Daten gesammelt an D-BAS weitergeleitet werden.

Tastaturaufbereitung: Diese Komponente erstellt eine personalisierte Tastatur für den Benutzer. Alle bereits vorhandenen, möglichen Antworten werden in sogenannten „Buttons“ abgebildet (siehe Abbildung 4.3). Der Nutzer hat so die Möglichkeit, auf einen der Buttons zu klicken, anstatt den Text per Hand einzugeben. Weiterhin ist die Komponente dafür zuständig, dass eben diese Buttons wieder verschwinden, wenn sie benutzt wurden bzw. der Nutzer an eine andere Stelle springt. Umgesetzt wurde dies in der `KeyboardHandler`-Klasse.

4.3.2 Backend-Kontext

Das Komponentenbündel Backend-Kontext ist für die Aufbereitung und Weiterverarbeitung von Anfragen bzw. Antworten zuständig. Es beinhaltet die folgenden Komponenten:

Sprach-Manager: Diese Komponente wird aktuell ausschließlich beim Anlegen eines neu-

en Themas benutzt. Hierzu wird aus einer Beschreibung, welche von einem Anwender zu einem Thema angegeben wird, die Sprache ermittelt. Dafür wird das Tool „Language Detection“ aus der „Tika“-Bibliothek[1] verwendet. Es wird bisher auf englische und deutsche Sprache überprüft.

Intention und Zusatzinformationen: Da nun alle Daten extrahiert und geladen wurden, kann auf Grund dieser entschieden werden, welche Absicht die Nachricht hat und sie können anschließend an das D-BAS-Interface weitergeleitet werden.

Datenaufbereitung: D-BAS sendet und empfängt Daten jeweils in JSON-Files. Daher wird die Komponente `JSONParser` benötigt, um die teilweise sehr komplexen Daten für/von D-BAS zu erstellen/extrahieren. Extrahierte Daten werden an die weiteren Komponenten gereicht, damit diese sie weiterverarbeiten können.

D-BAS-Interface: Das D-BAS-Interface ist für die Kommunikation zwischen Bot und D-BAS-Server zuständig. Auf Grund der Anwenderintention erstellt es eine HTTP-Anfrage an D-BAS, baut dazu eine Verbindung zum lokal laufenden Server auf und sendet bzw. empfängt die Daten, welche im D-BAS-Backend weiterverarbeitet werden. Als Antwort auf die Anfrage erhält der Chatbot ein JSON-File, in dem die Antwort des D-BAS-Servers abgespeichert ist, oder einen Statuscode. Nach Erhalt der Datei ist die Aufgabe des `HTTPHandlers` erfüllt. Sendet der Nutzer beispielsweise „/start“ an den Bot, so baut die Klasse `HTTPHandler` eine `HTTPURLConnection` mit der URL „http://localhost:4284/api/issues“ auf und erhält darüber als Antwort eine Liste der bereits bestehenden Chat-Themen.

4.3.3 Kontextspeicher

Zum Management der unterschiedlichen Kontexte zwischen Chatbot und Anwender sowie Chatbot und D-BAS-Backend werden zwei Datenspeicher verwendet.

So werden auf Seiten des Backendkontextes die aktuelle URL und ein Token, welcher von D-BAS angefordert wird, wenn ein Nutzer etwas hinzufügen möchte, abgespeichert. Eindeutig zuweisen lassen sich die Daten über die chatID, welche von Telegram geliefert wird.

[1] https://tika.apache.org/1.16/api/org/apache/tika/language/detect/LanguageDetector.html, abgerufen am 17.12.2020

Abb. 4.3: Kontextabhängige Keyboard-Buttons

Auch im `ConversationManager` werden die Daten über die chatID zugewiesen. Hier werden weiterhin die Daten `currentProcedure`, `isProcedureDone` sowie vorherige Eingaben des Nutzers, welche für eine Anfrage an D-BAS zwischengespeichert werden, abgelegt.

4.4 Design-Entscheidungen

Im Folgenden werden für die bereits aufgezählten Anforderungen Design-Entscheidungen getroffen und erläutert, aus welchen Gründen das implementierte Design gewählt wurde. So konnten z. B. nicht alle Funktionen des bestehenden D-BAS-Frontends übernommen werden.

4.4.1 Berücksichtigung von Spezifika des Telegram-Frontends

Die von Telegram bereitgestellte Schnittstelle unterscheidet sich in ihren Möglichkeiten signifikant von den für das bestehende Userinterface benutzten Tools. In Tabelle 4.2 sind die wesentlichen Unterschiede aufgelistet:

D-BAS	**Telegram**
Multiple-Choice-Auswahlmenü	Buttons
Neue Themen, Positionen und Prämissen in extra Textfeldern	Parameter werden vom Bot im Dialog abgefragt
Färben einzelner Argumentationsbausteine	nicht verfügbar
Highlighting einzelner Argumente, um Kontext zu verdeutlichen	nicht verfügbar

Tabelle 4.2: Übersicht der verwendeten Instrumente von D-BAS und deren Realisierung in Telegram

Die Positionen und Prämissen, sowie Reaktionen auf Argumentationen sind bei D-BAS durch ein *Multiple-Choice-Auswahlmenü* dargestellt. Bei der Umsetzung in Telegram hat sich die Implementierung über Buttons als sehr geeignet erwiesen. Der Nutzer wählt über diese seine Antwort aus und sie wird daraufhin in den Chat geschrieben. Es ermöglicht eine schnelle und simple Bedienung, einzig bei der Länge der in den Buttons enthaltenen Texte musste ein Abstrich gemacht werden. Es werden von Telegram aktuell nur 142 Zeichen abgebildet. Ist ein Argument oder eine Antwort länger, so wird der Text in den Buttons nur verkürzt dargestellt. Dadurch kann es zu Verständnisschwierigkeiten kommen, da der Nutzer ggf. nicht das gesamte Argument sehen kann. Eine Analyse des ersten Feldversuches (hierbei wurden die Positionen eines englischen[2] und deutschen[3] Themas untersucht) hat ergeben, dass zehn der insgesamt 47 vorhandenen Positionen die Länge von 142 Zeichen überschritten haben (etwa zwanzig Prozent). In Teilen war dies jedoch darauf zurückzuführen, dass Position und Prämisse zusammen eingegeben worden waren. In den anderen Fällen hätte sich die Eingaben sprachlich auch verkürzen lassen. Weiterhin wurde in den Anwendertests des Autors kein Fall konstruiert, in dem die Länge von 142 Zeichen nicht ausreichend gewesen wäre. Aus diesem Grund hat der Autor sich dazu entschieden, die zulässige Länge der Eingaben der Nutzer zu beschränken. Sendet ein Nutzer eine zu lange Nachricht an den Chatbot, so bittet dieser den Nutzer, die Nachricht kürzer zu formulieren. Dies führt zu einem weiteren positiven Ergebnis: An manchen Stellen bildet D-BAS sehr lange Antwortnachrichten, bei denen

[2] https://dbas.cs.uni-duesseldorf.de/discuss/improve-the-course-of-computer-science-studies, abgerufen am 17.12.2020

[3] https://dbas.cs.uni-duesseldorf.de/discuss/verbesserung-des-informatik-studiengangs, abgerufen am 17.12.2020

mehrere Argumente verbaut sind. Je kürzer die einzelnen Argumente sind, desto kürzer (und leichter lesbar) werden solche Satzkonstrukte.

Das *Hinzufügen neuer Themen, Positionen und Prämissen* wird in D-BAS über die Eingabe in einzelne dafür vorgesehene Textfelder realisiert. Hierbei hat der Autor sich dazu entschieden, alle notwendigen Parameter in einem Dialog abzufragen und zwischenzuspeichern. Sind alle Daten gesammelt, so werden sie gebündelt als Anfrage an D-BAS weitergeleitet.

Das *Färben und Highlighten einzelner Argumentationsbausteine* bietet Telegram in Nachrichten nicht an. Das redundante Wiederholen der Eingaben, sowie die für einen Messenger teilweise etwas künstlich klingenden Formulierungen, wirken für einen Chatverlauf nicht authentisch und der Nutzer könnte schnell das Interesse verlieren. Eine exemplarische Abwandlung der von D-BAS generierten Texte wird im folgenden Abschnitt gezeigt.

4.4.2 Aufteilung der Sprachverarbeitung zwischen Frontend und Backend

Im Rahmen der Arbeit wurde an verschiedenen Stellen erwogen, Formulierungen, die aus dem Backend kommen, anzupassen.

Ein Beispiel dafür ist in Tabelle 4.3 aufgeführt: Auf den ersten Blick scheint es einfach, vom förmlichen Sie auf ein chattypisches Du zu wechseln und die Nachrichten von D-BAS zu kürzen. Betrachtet man das Ganze genauer, so stellt man allerdings fest, dass dies größere strukturelle Auswirkungen hat. Nicht nur die Antworten von D-BAS müssen dafür abgeändert werden, sondern auch die Auswahlmöglichkeiten für den Nutzer.

Ein zweites Beispiel sind die Fälle, in denen Bezüge zwischen verschiedenen Textteilen durch Einfärbung / farbliche Unterlegung hervorgehoben werden. In einem auf Bots angepassten Dialogverlauf wie im Beispiel unten dargestellt, sind diese Bezüge einfach heraus gekürzt, da der Anwender die Referenz normalerweise ohne Verwechselungsgefahr noch im Kopf hat. Selbst wenn nicht, z. B. weil er inzwischen gestört wurde, entspricht es einem natürlichem Chat-Verhalten, den Kontext einfach im Chatverlauf nachzusehen.

Der Autor hat sich generell gegen eine Anpassung der vom D-BAS-Backend gelieferten Texte im Frontend entschieden, da auf das bestehende System aufgebaut werden sollte. Weiter-

hin scheint es dem Autor auch nicht sinnvoll, Backendtexte der Argumente und ihrer Logik im Frontend zu manipulieren. Zwar liefert die vorhandenen HTML-Schnittstelle Statusinformationen mit, allerdings hält es der Autor für eine sauberere Architektur, die sprachliche Aufbereitung in Verbindung mit der Argumentationslogik auch im Backend zu belassen und dort eine Chatbot-Schnittstelle zu implementieren.

Auf Grund der erprobten Beispiele geht der Autor davon aus, dass eine komplett andersartige Aufbereitung für den Chatbot möglich ist, ohne die erforderliche Präzision der Logik für die D-BAS Schnittstelle zu verlieren. Dies müsste jedoch ähnlich wie bei der bestehenden Schnittstelle im Rahmen eines Feldtests verifiziert werden.

jetziger Text	**vorangegangene Nachricht**	**angepasster Bot-Text**
Was halten Sie von der Aussage, dass [Position]?	[Position]	Was hältst du von dieser Aussage?
Was ist ihr wichtigster Grund [dafür/dagegen], dass [Position]? Weil...	[dafür/dagegen]	Warum denkst du das? Weil...
Ich stimme zu, dass [Prämisse]. Aber ich glaube, dass es keine gute Begründung dafür ist, dass [Position]. Ich denke, dass [andere Prämisse]. Was denken Sie darüber?	[Prämisse]	Ich denke, dass das keine gute Begründung ist. Ich denke, dass [andere Prämisse]. Was hältst du davon?
Was ist ihr wichtigster Grund gegen die Aussage, dass [Prämisse]? Weil...	Ich halte deren Aussage für falsch und möchte widersprechen.	Warum glaubst du, dass die Aussage falsch ist? Weil...

Tabelle 4.3: Exemplarische Umwandlung in Bot Text

In Tabelle 4.3 sind einige Beispiele für entsprechende Umformulierungen aufgeführt. Hierbei wird Gebrauch von zwei wesentlichen Techniken gemacht:

- Vermeidung der Wiederholung von Inhalten, welche in der oder den vorangegangenen Nachrichten stehen
- Ersetzen der Bezeichnung von Aussagen, Positionen und Aussagen durch den tatsächlichen Inhalt

Mit der vorgeschlagenen Erweiterung des Backends ließe sich auch eine Vereinfachung der Darstellung der mathematischen Logik der Argumentation für das Medium des Chatbots erreichen. Genauer gesagt eine Dekomposition der Fragen an den Anwender zur Logik seines Standpunkts.

Diese bietet sich insbesondere für Abfragen an, bei denen seitens des Backends vier oder mehr Optionen vorgelegt werden sollen. Die daraus entstehende Komplexität der Formulierung kann durch Dekomposition in mehrere Chat-Schritte für den Chatbot vereinfacht werden und damit u. a. auch zu einer bessern Verständlichkeit für andere Nutzergruppen als Informatik-Studenten beitragen.

Ein Beispiel dafür sind die Optionen, die bei der Begründung zu einer Argumentation vorgelegt werden (Ausgangssatz: „Ich denke, dass es richtig ist, dass wir einen neuen Fahrradständer brauchen, weil der alte fast zusammenbricht. Was denken Sie darüber?“):

- Ich halte deren Behauptung für falsch und möchte widersprechen.
- Ich halte deren Behauptung für richtig, aber sie wird nicht von der Begründung gestützt.
- Ich halte deren Begründung für falsch und möchte widersprechen.
- Ich halte deren Behauptung für richtig und finde sie überzeugend.

In Kombination der Dekomposition mit Ja/Nein Fragen und der zuvor dargestellten Einsetz-Technik könnte die Konversation wie in Tabelle 4.4 abgewandelt werden:

<table>
<tr><td colspan="4">Ich denke, dass es richtig ist, dass wir einen neuen Fahrradständer brauchen, weil der alte fast zusammen bricht. Stimmt das?</td></tr>
<tr><td>Ja</td><td colspan="3">Nein.</td></tr>
<tr><td rowspan="3"></td><td colspan="3">Warum stimmt das denn nicht?</td></tr>
<tr><td>Ich halte die Begründung für falsch.</td><td>Ich halte die Behauptung für falsch.</td><td>Ich halte die Behauptung für richtig, aber sie wird nicht von der Begründung gestützt.</td></tr>
<tr><td>Warum denn? Weil ...</td><td>Warum denn? Weil ...</td><td>Warum denn? Weil ...</td></tr>
</table>

Tabelle 4.4: Beispiel eines dekomponierbaren Szenarios

Zusammenfassend hat der Autor vor den dargestellten Hintergründen den Ansatz verfolgt, im Frontend sprachliche Vorverarbeitungen anzusiedeln, die für die Logik des Backends unschädlich sind. Weiterhin werden sprachliche Änderungen im Zusammenhang mit den Argumenten und ihrer Logik als Erweiterungen für das Backend vorgeschlagen.

4.5 Sicherstellung der Softwarequalität

Bei der Sicherstellung der Softwarequalität wurden zahlreiche Unit-Tests geschrieben, welche die korrekte Funktionsweise der wichtigsten Methoden sicherstellen. Hierbei wurde jeweils der Gesamtkontext abstrahiert, um nur auf der Ebene der Methode die Funktion zu prüfen. Geprüft wurde nur die eigenständig implementierte Logik, die korrekte Funktion beispielsweise der benutzten Telegram-Funktionen wurde vorausgesetzt.

Nachdem eine erste Version des Chatbots mit funktionierenden Schnittstellen sowie einer simplen Kommandostruktur erstellt war, wurde diese im Rahmen mehrerer Iterationszyklen mit Unterstützung von fünf Testanwendern weiter entwickelt.

Bei der Implementierung wurde das Augenmerk auf intuitives Verständnis bei der Bedienung gelegt. Dafür wurde mehreren Versuchspersonen der Bot vorgestellt und nach einer kurzen Einweisung haben diese mit dem Bot gechattet. Bei der Implementierung wurden die Testpersonen angewiesen, bei Neuerungen mit dem Bot zu chatten und ein Feedback zu geben. Gegenstand des Feedbacks waren:

- Rückmeldung zur Verständlichkeit
- Anregungen für leichtere Verständlichkeit
- Rückmeldung zur Bedienerfreundlichkeit
- Erstellen von Fehlerberichten

Ein Feedback-Bericht enthielt dabei einen Screenshot aus dem Chatverlauf sowie einen kleinen beschreibenden Text mit der Fehlerbeschreibung.

Im Rahmen der Entwicklung und iterativen Verbesserung der jeweiligen Versionen des Chatbots wurde jeweils das Tool „SonarLint“[4] zur systematischen Überprüfung der Einhaltung gängiger Standards der Sourcecode-Qualität eingesetzt.

[4] https://www.sonarlint.org/, abgerufen am 17.12.2020

4.6 Sicherheit

Wenn man eine Loginanfrage an den Bot sendet, so fragt dieser in einem Dialog „nickname“ und „password“ ab. Diese werden dann in eine Loginanfrage an den D-BAS-Server weiterverarbeitet. Als Antwort erhält er entweder den HTTP-Statuscode 401 zurück, wenn die Daten falsch sind, oder er erhält einen sogenannten „Bearer Token“. Bei einem Bearer Token handelt es sich hier um einen „JSON Web Token“, welcher einen kompakten und abgeschlossenen Weg, um JSON-Objekte zwischen zwei Teilnehmern sicher zu übermitteln [Jwt], bietet. Diesen speichert der Bot für jeden erfolgreich eingeloggten Benutzer ab, um ihn bei Anfragen, welche einen Login benötigen, mitzusenden. Dieser für die vorliegende prototypische Anwendung zunächst gewählte Ansatz ist natürlich nicht sicher, da man seine Logindaten in einer Nachricht an den Bot sendet. Zwar sind die Daten in der Telegram-Cloud verschlüsselt [5], sollte es jedoch gelingen, diese zu entschlüsseln, kann mittels Textsuche schnell die Stelle, an der die Daten stehen, gefunden werden. Eine mögliche Funktion, dass der Bot diese Nachricht editiert, oder gar aus dem Verlauf löscht, ist nicht vorhanden. Telegram erlaubt dies nur für die eigenen Nachrichten (sprich der Nutzer müsste seine Nachricht selbst editieren bzw. löschen). Eine eventuelle Erhöhung der Sicherheit müsste zu einem späteren Zeitpunkt ggf. noch berücksichtigt werden.

4.7 Probleme

An der Schnittstelle zu D-BAS sind mir zwei kleinere Punkte aufgefallen, die zum Teil die Entwicklung des Bots erschwert haben:

Fehler in der „Stepback“-Funktion: Zunächst gibt es einen Bug in der „Stepback“-Funktion. Bei dieser Funktion handelt es sich um die Möglichkeit, während einer Diskussion einen Schritt zurück zu gehen. Dies kann sehr hilfreich sein, wenn man mit dem Angebot an Argumenten, welche durch den Bot bereitgestellt werden, nicht zufrieden ist und evtl. doch lieber eine andere Option auswählen möchte. Auf der „/discuss“-Seite funktioniert diese Funktion einwandfrei. Auf der vom Bot verwendeten „/api“-Seite produziert die hinterlegte URL allerdings einen 404-er Fehler. Aus diesem Grund wurde die Funktion nicht eingebunden.

[5] https://telegram.org/faq/de, abgerufen am 17.12.2020

Fehlerhafte Suchfunktion: Weiterhin bietet D-BAS eine Suchfunktion an. So kann man die Themen und Argumente auf gewisse Suchworte durchforsten. Diese Funktion scheint fehlerhaft, beispielsweise wurden bei der Suche nach „Winterzeit“ Texte angezeigt, die dieses Suchwort gar nicht enthalten (z. B. das Thema „Make the world better“). Von einer Implementierung wurde aus diesem Grund zum jetzigen Zeitpunkt abgesehen.

Kapitel 5

Funktionen und mögliche Erweiterungen

In diesem Kapitel wird mit einigen Beispielen genauer auf die Funktionen des Chatbots eingegangen. Weiterhin werden Erweiterungsmöglichkeiten aufgezeigt und der Bot in den Kontext der bereits in Kapitel 2 erwähnten verwandten Arbeiten eingeordnet.

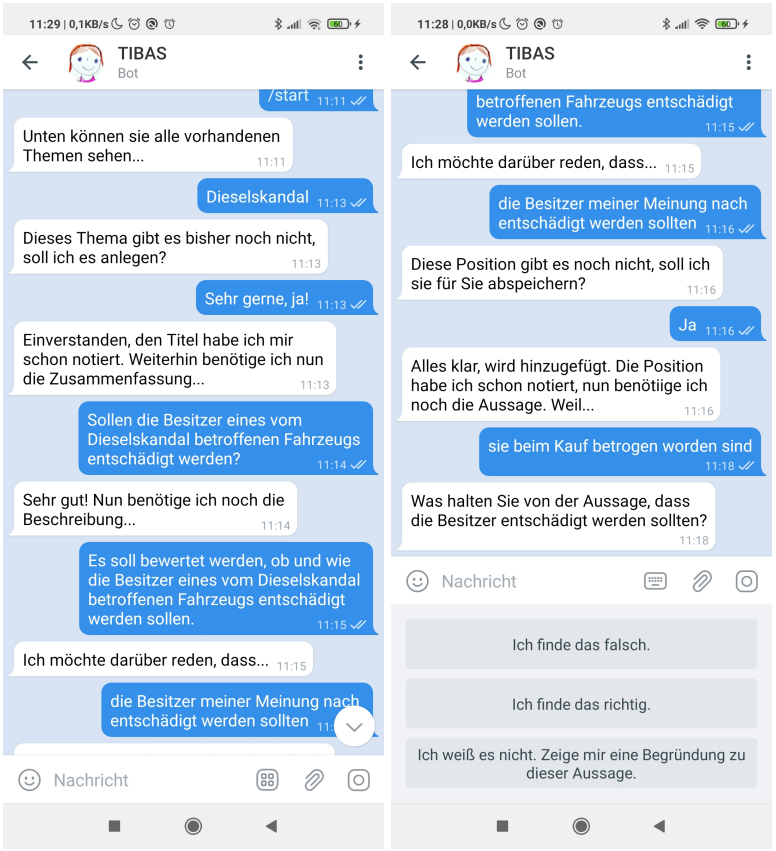
5.1 Funktionsbeispiele

Bei den Funktionsbeispielen werden einige Features dargestellt. Weiterhin möchte der Autor darauf hinweisen, dass der Chatbot versucht, bei wiederkehrenden Prozeduren (Eingabe von zu langen Argumenten, Verwendung von Schimpfwörtern und weiteren) durch variierende Antwortmöglichkeiten den Fluss des Gespräches aufzulockern. Dies würde durch Illustrierungen zu viel Redundanz führen, wird beim chatten mit dem Bot allerdings schnell deutlich.

5.1.1 Anlage neuer Daten

Im Folgenden wird das Beispiel aus 5.1(a) und 5.1(b) genauer erläutert, in dem es um den Dieselskandal gehen soll:

Der Nutzer befindet sich in der Themenübersicht und gibt „Dieselskandal“ ein. Der Chatbot stellt nach einer Prüfung fest, dass das Thema noch nicht in der Datenbank angelegt ist



(a) Beispieltext für neues Thema (b) Beispieltext für neue Position

Abb. 5.1: Anlage neuer Daten

und startet die Prozedur, um dieses Thema zu erstellen. Hierfür wird zunächst sichergestellt, dass der Benutzer tatsächlich ein neues Thema anlegen möchte. Nach der Zustimmung des Nutzers und nachdem Titel, Zusammenfassung und Beschreibung abgefragt wurden, werden die Daten an D-BAS weitergeleitet und das Thema angelegt. Im Anschluss erstellt der Nutzer eine erste Position mit Begründung. Hierbei ist zu beachten, dass der Bot einen Teil der Begründung, nämlich „meiner Meinung nach“, streicht, bevor die Daten auf den D-BAS-Servern abgelegt werden.

5.1.2 Hinzufügen neuer Argumente

In Abbildung 5.2 ist zu sehen, wie der Nutzer versucht, zu einer vorhandenen Position ein neues Argument hinzuzufügen. Im ersten Versuch probiert er ein sehr langes, verschachteltes

und schwer verständliches Argument hinzuzufügen. Der Chatbot achtet darauf, dass die Argumente, die er entgegen nimmt, nicht zu lang werden und kontert mit „Ich habe den Anfang der Nachricht schon wieder vergessen. Fasse dich bitte kurz!“. Bei diesem Satz handelt es sich um eine von mehreren möglichen Antworten, welche bei zu langen Eingaben erscheinen kann.

Auch beim zweiten Versuch, bei dem der Nutzer das Schimpfwort „Idiot“ im Text schreibt, weist der Chatbot diese Nachricht zurück. Der anschließende Versuch vom Nutzer gelingt und wird auf dem D-BAS-Server abgespeichert.

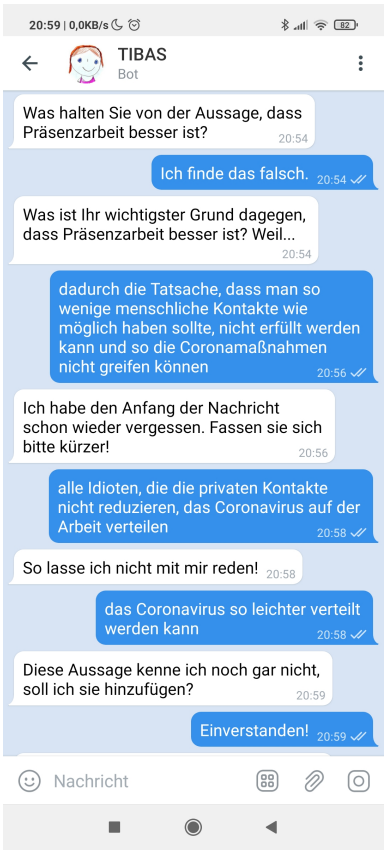


Abb. 5.2: Filtering der Eingabe vom Chatbot

5.1.3 Anmeldeprozess

In dem Beispiel aus Abbildung 5.3 versucht der Anwender eine neue Prämisse hinzuzufügen, obwohl er sich vorher noch nicht authentifiziert hat. Nachdem der Bot ihn darauf hingwiesen

hat, zögert er nicht lange und startet über „/login“ die Login-Prozedur (Nickname und Passwort wurden hier nachträglich abgeändert!). Bevor die Prozedur gestartet wird, speichert der Bot den Zustand der Diskussion, um nach einem erfolgreichen sowie erfolglosen Anmeldeversuch die Unterhaltung fortzuführen.

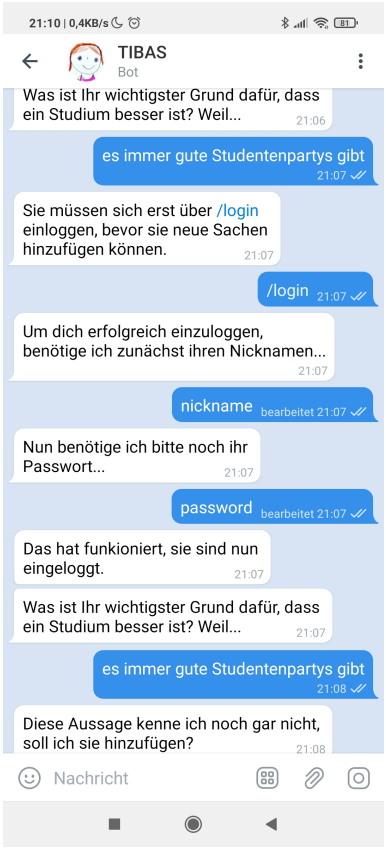


Abb. 5.3: Geforderter Login um neue Daten anzulegen

5.2 Erweiterungsmöglichkeiten

Ein Hauptpunkt, der aus Sicht des Autors signifikant zu einem flüssigen und als natürlich empfundenen Chatverlauf beitragen würde, wäre eine Erweiterung der momentanen sprachlichen Aufbereitung im Backend um eine chatbot-spezifische Schnittstelle (siehe Tabelle 4.3). Weiterhin bestehen noch viele Möglichkeiten, die Funktion des Chatbots weiter auszubauen.

Rechtschreibprüfung: Auch hier könnte die Qualität der im Backend angelegten Argu-

mentationstexte weiter verbessert werden, indem korrigierbare Rechtschreibfehler im Rahmen des Preprocessing bereinigt würden.

Erkennung und Abbildung weiterer Anwenderintentionen: Über die im vorigen Abschnitt hinaus bereits beschriebene Erkennung von Zustimmung / Ablehnung würde für zahlreiche weitere Anwenderintentionen die Erkennung textlicher Eingaben den Chatbot aufwerten.

Ergänzung des Etikette-Services um weitere Services: Neben der hier exemplarisch verwendeten Liste[1] liegen noch weitere, vergleichbare Listen vor, die man ebenfalls berücksichtigen könnte.

Reminder: Außerdem könnte der Bot nach längerer Inaktivität des Nutzers eine automatisierte Nachricht, welche zum Weitermachen animieren soll, versenden.

Chatbot für Guppenchats: Weiterhin finden viele Diskussionen nicht in privaten Chats, sondern in Gruppenchats statt. Es wäre denkbar, dass ein Chatbot in einer solchen Gruppe die Positionen und Aussagen automatisiert aufnehmen kann und eventuell Rückfragen an die Nutzer stellt bzw. deren Meinung abfragt, um so einen Argumentationsbaum anzulegen.

Sicherheit bei der Anmeldung: Für die momentane Lösung sollte ein anderer, sicherer Ansatz untersucht werden.

5.3 Abgrenzung zu verwandten Arbeiten

Mit dem vorliegenden Chatbot wurde den bereits bestehenden Frontends für D-BAS ein weiteres hinzugefügt.

Gegenüber dem ursprünglichen Web-Frontend wird in der vorliegenden Arbeit insbesondere ein speziell auf die Kommunikation in einem Chatbot ausgerichteter Ansatz zur Modifizierung der Kommunikationsinstrumente und -texte vorgeschlagen. Die wesentlichen Merkmale dieses Ansatzes sind:

[1] https://www.woltlab.com/attachment/3615-schimpfwortliste-txt/, abgerufen am 17.12.2020

- hybride Aufteilung der Kommunikation in Texteingaben und Buttons
- Verwendung der Einsetz-Technik als Ersatz für die im Web-Frontend verwendete Wiederholungstechnik
- Dekomposition einer Auswahl in mehrere Kommunikationsschritte
- Weglassen von Inhalten, die aus dem Chatverlauf unmittelbar klar erkennbar sind

Ferner wurden einige Techniken zur Vermeidung unnötiger Arbeitslast in den Korrektur-Queues im Backend vorgeschlagen:

- Etikette-Prüfung
- Eliminierung bestimmter Sprachhülsen
- Erzwingen kürzerer Argumentationen
- Rechtschreibprüfung

In Abgrenzung zu trainierbaren Tools wie Jebediah handelt es sich bei dem vorliegenden Ansatz um eine Individualentwicklung mit einfachen, heuristischen Methoden. Diese hat gegenüber der Verwendung trainierbarer Tools den Nachteil, dass die Verarbeitungsschritte (wie z. B. Erkennung der Intentionen) manuell entwickelt werden müssen. Von Vorteil ist jedoch die stringentere, und dennoch gut verständliche Darstellungsmöglichkeit der mathematischen Logik, die durch die Nutzung von Telegram-Features möglich wird.

Kapitel 6

Fazit

In der vorliegenden Arbeit wurde ein Telegram-Interface für D-BAS als Prototyp eines Chatbot-Interfaces realisiert. Die Arbeit erfolgte im Rahmen einer Reihe von Forschungsarbeiten über die Onlinebeteiligung von Bürgern an Entscheidungsprozessen in Politik und Verwaltung. Die Ergebnisse sind aus Sicht des Autors auf andere Umfelder von Entscheidungsprozessen (z. B. Wirtschaft, Schulen, Vereine, etc.) übertragbar.

Die Realisierung von D-BAS über einen Chatbot ist sinnvoll, um die Beteiligung der Bürger zu erhöhen, da der tägliche Umgang mit Messengern gängiger ist, als das Abrufen bestimmter Websites. Weiterhin finden viele Diskussionen mittlerweile ohnehin in Messengern statt.

Außerdem hat sich die Ausgangshypothese, dass D-BAS in einem Messenger umgesetzt werden kann, bestätigt. Allerdings fordert die Nutzung eines Chatbots eine signifikante sprachliche Umgestaltung, welche sich auf die stark unterschiedlichen Medien zurückführen lässt. Die sprachliche Umgestaltung kann letztlich sogar vorteilhaft sein, da sie die Logik des Backends in bürgernäherer Sprache einfacher und nachvollziehbarer macht. Auch hat sich der hybride Ansatz (Kombinierung von Buttons und Freitexteingaben) bewährt. Er verbindet die Vorteile einer freisprachigen Eingabemöglichkeit mit der Erfordernis, manchmal Logik-Zusammenhänge exakter abzufragen, als dies bei einem rein freisprachlichen Ansatz möglich wäre.

Aus diesen Gründen schlägt der Autor vor, sowohl die Verwendung eines Chatbots, als auch die Verwendung des hybriden Ansatzes weiter zu verfolgen. Dabei müsste insbesondere die sprachliche Zusammenstellung der Antworten im Backend auf die hier vorgeschlagenen,

chatbot-spezifischen Kommunikationstechniken (Weglassen, Einsetzen und Dekomponieren) erweitert werden, da deren Implementierung innerhalb des Chatbots unter Architektur-Gesichtspunkten nicht sinnvoll erscheint.

Eine derartige Erweiterung des D-BAS-Backends ist aus Sicht des Autors der naheliegendste Schritt, die bisherigen Arbeiten in Richtung einer Erweiterung des Frontend-Zugangs fortzuführen.

Darüber hinaus hält der Autor für die Zukunft die Ausweitung auf weitere Messenger für sinnvoll, um die Reichweite weiter zu erhöhen.

Literatur

[BM20] Markus Brenneis und Martin Mauve. „deliberate–online argumentation with collaborative filtering“. In: *Computational Models of Argument* 326 (2020), S. 453–454.

[Ebb19] Björn Ebbinghaus. „Decision Making with Argumentation Graphs“. In: *arXiv preprint arXiv:1908.03357* (2019).

[EM20] Björn Ebbinghaus und Martin Mauve. „decide: Supporting Participatory Budgeting with Online Argumentation“. In: *Submitted to the 8th International Conference on Computational Models of Argument (COMMA 2020)(cit. on p. 66)*. 2020.

[Gro18] Katharina Große. *Benutzerzentrierte E-Partizipation*. Springer, 2018.

[Jwt] *JSON Web Token*. `https://jwt.io/introduction/`. [Accessed: 2020-11-17].

[Kra+18] Tobias Krauthoff u. a. „D-BAS - A Dialog-Based Online Argumentation System.“ In: *COMMA*. 2018, S. 325–336.

[MEM18] Christian Meter, Björn Ebbinghaus und Martin Mauve. „Jebediah-Arguing with a Social Bot.“ In: *COMMA*. 2018, S. 467–468.

[Mesa] *Studie Messenger Nutzung 2020 Deutschland*. `https://www.messengerpeople.com/de/studie-messenger-nutzung-2020-deutschland/`. [Accessed: 2020-11-29].

[Mesb] *Weltweit beliebteste mobile Messenger Apps Juli 2020*. `https://www.messengerpeople.com/de/weltweite-nutzer-statistik-fuer-whatsapp-wechat-und-andere-messenger/`. [Accessed: 2020-12-17].

[MKM17] Christian Meter, Tobias Krauthoff und Martin Mauve. „discuss: Embedding dialog-based Discussions into Websites“. In: *International Conference on Learning and Collaboration Technologies*. Springer. 2017, S. 449–460.

[Pra+20] H Prakken u. a. „A Persuasive Chatbot Using a Crowd-Sourced Argument Graph and Concerns“. In: *Computational Models of Argument: Proceedings of COMMA 2020* 326 (2020), S. 9.

[SDP20] Toni Stucki, Sara D'Onofrio und Edy Portmann. *Chatbots gestalten mit Praxisbeispielen der Schweizerischen Post*. Springer, 2020.

[SM19] ALEXANDER SCHNEIDER und CHRISTIAN METER. *Various Efforts of Enhancing Real World Online Discussions*. 2019.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 18. Dezember 2020 Fabian Correnz