**Universität Leipzig,**
**Fakultät für Mathematik und Informatik**
**Institut für Informatik**

Titel der Arbeit:
Nutzungsszenarien für Chatbots zur Verbesserung der unternehmensinternen Kommunikation

# Bachelorarbeit

Leipzig, März 2018

vorgelegt von
Frommert, Christian
Studiengang: Informatik B.Sc.

Betreuung: Dr.-Ing. Romy Elze
Institut für angewandte Informatik
Universität Leipzig

# Abstract

War die E-Mail für lange Zeit der einzige Weg, digital mit Kollegen zu kommunizieren, verwenden heutzutage immer mehr Unternehmen interne soziale Netzwerke und Messaging Anwendungen für die Kommunikation untereinander. Oftmals werden jedoch nicht alle Vorteile dieser Dienste ausgeschöpft.

Eine Möglichkeit, diese Systeme intelligent anzuwenden, stellen Chatbots dar, welche bislang hauptsächlich für die Kundenkommunikation verwendet werden. Diese Bachelorarbeit zeigt jedoch an zwei Nutzungsszenarien, dass sie ebenfalls einen großen Beitrag zur Assistenz unternehmensinterner Organisationsprozesse leisten und die interne Kommunikation verbessern können.

Die vorliegende Arbeit gibt eine kurze Einführung in Enterprise Social Networks, Natural Language Understanding und Chatbots. Basierend darauf werden Anforderungen an Chatbots für den unternehmensinternen Einsatz analysiert, bereits vorhandene Lösungen recherchiert, ein Konzept zur Nutzung innerhalb von Unternehmen entwickelt und denkbare Nutzungsszenarien erörtert. Schließlich wird ein Chatbot implementiert, der zwei Szenarien umsetzt und bei unternehmensinterne Prozessen assistieren kann.

Als Ergebnis wird festgestellt, dass die Möglichkeiten von Chatbots über die Kundenkommunikation hinausgehen und bereits jetzt einfache Unternehmensprozesse unterstützt oder sogar gänzlich automatisiert werden können.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

**AD** Active Directory.

**AIML** Artificial Intelligence Markup Language.

**API** Application Programming Interface.

**CRM** Customer-Relationship-Management.

**ERP** Enterprise-Resource-Planning.

**ESN** Enterprise Social Network.

**JSON** JavaScript Object Notation.

**MITIE** MIT Information Extraction.

**ML** Machine Learning.

**NER** Named-Entity Recognition.

**NLG** Natural Language Generation.

**NLP** Natural Language Processing.

**NLU** Natural Language Understanding.

**POS** Part-of-speech.

**SaaS** Software as a Service.

**XML** Extensible Markup Language.

# Inhaltsverzeichnis

# 1 Einleitung

Unternehmen setzen verstärkt auf den Einsatz von Chatbots. Die Gründe: So gut wie jedes Unternehmen ist in sozialen Netzwerken, wie Facebook, vertreten (Stelzner 2017). Mehr als drei Milliarden Menschen nutzen soziale Medien (Kemp 2018). Einer Umfrage unter 500 Verbrauchern aus dem Jahre 2015 zufolge erachten 62% aller Befragten die Erreichbarkeit des Anbieters, außerhalb der üblichen Geschäftszeiten, als wichtigstes Kriterium bei der Auswahl eines Dienstleisters (Rennebarth 2015). Dieses Ziel ist mit einem intelligenten Chatbot, welcher Kundenanfragen schnell und präzise beantwortet, leicht zu erreichen. Doch wie können Chatbots genutzt werden, um nicht nur die Kommunikation zum Kunden, sondern auch zwischen Mitarbeitern innerhalb von Unternehmen zu verbessern?

Die meisten Angestellten müssen heutzutage mit Informationssystemen arbeiten (Destatis 2017), wobei sich die Leistungsfähigkeit von IT-Systemen schneller entwickelt als die kognitiven Kapazitäten von Menschen. Wenn der Mensch an seine Grenzen stößt, resultiert dies in Überforderung und hat oftmals negative Auswirkungen, wie Aufregung, Frustration oder Stress. Deshalb greifen Menschen oft auf einfache Heuristiken zurück, um Entscheidungen einfacher fällen zu können. Diese Strategie führt jedoch oftmals zu Fehlentscheidungen. „Nutzerassistenzsysteme können hierbei helfen die Aufgaben besser, d.h. mit einem qualitativ besseren Ergebnis, schneller oder auch angenehmer für den Nutzer, auszuführen." (vgl. Morana u. a. 2017).

Diese Bachelorarbeit beleuchtet die Motivation, Chatbots innerhalb von Unternehmen einzusetzen, erklärt die Grundlagen der sozialen Netzwerke, Natural Language Unterstanding und Chatbots, analysiert sinnvolle Nutzungsmöglichkeiten für Chatbots und beschreibt die Implementierung anhand von zwei Beispielen.

## 1.1 Motivation

Die unternehmensinterne Kommunikation unterliegt einem kontinuierlichen Wandel. Konnte man früher ausschließlich auf das persönliche Gespräch, Telefon, Post und Fax zurückgreifen, machen heutzutage E-Mails einen Großteil der Kommunikation aus (Konrat 2008). Hierdurch konnte u.a. der Papierverbrauch minimiert werden, sodass sich einige Unternehmen der Vision „Papierloses Büro" nähern konnten bzw. bereits erfolgreich umsetzen (Pfaff 1995).

Die Konsequenz daraus ist jedoch ein sehr hohes E-Mail Aufkommen, da sowohl die interne als auch die externe Kommunikation zum und vom Kunden gebündelt im eigenen Postfach landet. Für die interne Kommunikation haben E-Mails einige Nachteile: Enthaltene Informationen könnten veraltet sein, ggfs. ist ein E-Mail Verteiler nicht auf dem aktuellsten Stand und die Mitarbeiter erhalten entweder zu viele oder nicht alle relevanten E-Mails. Zudem werden irrelevante Nachrichten mit der gleichen Priorität wie wichtige E-Mails empfangen. Jede neue E-Mail lenkt außerdem von der aktuellen Arbeitsaufgabe ab, sodass Untersuchungen zufolge nach der Unterbrechnung durch eine E-Mail im Durchschnitt 64 Sekunden benötigt werden, um die unterbrochene Arbeit mit der ursprünglichen Konzentration erneut aufzunehmen (Jackson, Dawson und Wilson 2002).

Für die interne Kommunikation verzichten deshalb immer mehr Unternehmen auf E-Mails (Sawall 2011) und setzen verstärkt Enterprise Social Network (ESN) und Messaging Dienste ein. Allerdings besteht bei der Nutzung dieser Programme ebenfalls die Möglichkeit der Unterbrechnung des Arbeitsablaufs und der Ablenkung. Diese Effekte können jedoch, im Gegensatz zur E-Mail Kommunikation, durch ein geeignetes Konzept guter digitaler Arbeit sowie der stärkeren Vernetzung aller Mitarbeiter untereinander aufgrund des ESN minimiert werden. Diese Überlegungen werden im Projekt „SB:Digital" des Instituts für angewandte Informatik e.V. Leipzig[1] sowie im Buch „Chat-Kommunikation in Beruf, Bildung und Medien: Konzepte, Werkzeuge, Anwendungsfelder" (Beißwenger und Storrer 2005) untersucht und sind nicht Bestandteil dieser Bachelorarbeit, bilden jedoch die essentielle Grundlage des effektiven Einsatzes von Chatbots zur Verbesserung der digitalen Zusammenarbeit.

Eine weitere Herausforderung der Kommunikation mithilfe von ESN und Messaging Programmen ist, dass diese Dienste weitere Werkzeuge, neben den bereits im Intranet vorhandenen Anwendungen, darstellen und somit bei fehlender Integration keine Kommunikation

[1] https://sbdigital.infai.org

untereinander gewährleistet wird. Als Resultat müssen die Mitarbeiter mit einer Vielzahl von unabhängigen Softwarelösungen arbeiten, was sehr mühselig und unmotivierend ist (Eckmüller 2016).

Hier können Chatbots Abhilfe schaffen, indem sie die Informationen der einzelnen Programmen bündeln, im ESN oder Messaging Dienst bereitstellen und so die unternehmensinterne Arbeit verbessern.

## 1.2 Zielsetzung

Ziel dieser Bachelorarbeit ist es, praxisnahe Szenarien zum Einsatz von Chatbots aufzuzeigen, um die Kommunikation innerhalb von Unternehmen zu verbessern. Hierfür werden Einsatzmöglichkeiten analysiert, ein Konzept entwickelt und ein Chatbot implementiert, der zwei vorab definierte Nutzungsszenarien bedienen kann.

Im Zuge der Anforderungsanalyse und Konzeptionierung wird ein ESN bzw. ein Messaging Dienst vorgegeben und ein Chatbot auf Basis von festzulegenden Kriterien ausgewählt. Die darauf aufbauende Implementierung eines Prototypen soll aber so ausgelegt werden, dass die erarbeiteten Erkenntnisse auf weitere Dienste und Chatbots übertragbar sind.

Schließlich werden die eruierten Nutzungsszenarien umgesetzt und Beispiel-Dialoge geführt, wie sie im Unternehmen auftreten können. Mithilfe der gewonnen Erkenntnisse während der Recherche sowie Umsetzung, wird folgend eine Zusammenfassung und ein Ausblick der weiteren Entwicklung gegeben.

Aufgrund des limitierten Umfangs geht diese Arbeit weder auf die Kommunikationsstrategie innerhalb von Unternehmen, noch auf die Kundenkommunikation mithilfe von Chatbots ein, sondern konzentriert sich ausschließlich auf die Verbesserung der internen Unternehmenskommunikation mithilfe von Chatbots. Hierfür ist innerhalb dieser Bachelorarbeit maßgeblich, dass das zu betrachtende Unternehmen bereits über ein Konzept zur Nutzung von ESN oder Messaging Diensten verfügt und auf weitere Datenquellen im Unternehmen zugreifen kann.

## 1.3 Struktur der Arbeit

Kapitel 1 stellt eine Einleitung in das Thema dar und definiert Motivation, Zielsetzung sowie Struktur dieser Arbeit. In Kapitel 2 werden die Grundlagen zu ESN, Natural Language Processing bzw. Understanding und Chatbots erläutert, mögliche Datenquellen in unternehmenseigenen Netzwerken aufgelistet sowie der aktuelle Entwicklungsstand von Chatbots dargestellt. Kapitel 3 erläutert die Methodik dieser Bachelorarbeit hinsichtlich der Anforderungserhebung sowie Konzeptionierung und Implementierung.

Im 4. Kapitel werden die Anforderungen an ESN und Chatbots für den unternehmensinternen Einsatz beschrieben und zwei Use Cases zur Anwendung von Chatbots im unternehmensinternen ESN genauer definiert. Das 5. Kapitel beschreibt das Konzept der zur Umsetzung der Use Cases nötigen Schritte, unterteilt in Middleware, Chatbot und Natural Language Understanding. In Kapitel 6 wird die Implementierungsphase beschrieben, unterteilt in der Entwicklung einer Middleware, welche Nachrichten zwischen ESN und Chatbot vermittelt, der Entwicklung eines Chatbots allgemein sowie der Umsetzung der vorher definierten Use Cases. Abschließend werden in Kapitel 7 die Ergebnisse zusammengefasst und ein Ausblick zu Chatbots allgemein sowie für den unternehmensinternen Einsatz vermittelt.

# 2 Grundlagen

In diesem Kapitel werden die Grundlagen zum Thema dieser Bachelorarbeit erarbeitet. Es bietet eine kurze Einführung in ESN, Natural Language Processing (NLP) bzw. Natural Language Understanding (NLU) und Chatbots.

## 2.1 Enterprise Social Networks

Enterprise Social Networks (ESN) sind soziale Netzwerke, die in Unternehmen eingesetzt werden (Steingen und Mausz 2014). Sie sind der Öffentlichkeit normalerweise nicht zugänglich, sondern nur im unternehmensinternen Intranet verfügbar. Dabei bestehen ESN allgemein aus einem Mitgliederverzeichnis, Aktivitäts-Streams und Gruppen (Turban, Bolloju und Liang 2011). Das Mitgliederverzeichnis beinhaltet alle Mitarbeiter des Unternehmens und ist durchsuchbar. In Aktivitäts-Streams fließen die Aktivitäten der Mitarbeiter ein, die sich in den gleichen Gruppen befinden oder eine hohe Popularität aufweisen, gemessen durch „Gefällt mir"-Angaben. Mithilfe der Gruppenverwaltung kann die Unternehmensstruktur, idealerweise anhand eines Organigramms, abgebildet und den Mitarbeitern die entsprechenden Gruppen zugewiesen werden (Zhang, Yu und Lv 2015).

Integrierte Anwendungen und externe Services erweitern das ESN — beispielsweise um ein Umfrage-Tool (z.B. Humhub Polls[1]), einen Kalender (z.B. Integration von Google-Kalender mit Automate.io[2]), eine Datei- und Dokumentenverwaltung (z.B. Humhub Files[3]), ein Projekt- und Aufgabenmanagementsystem (z.B. Integration von Trello mithilfe von Unito[4]), ein Customer Relationship Managementsystem (z.B. Integration von Salesforce in

[1] https://www.humhub.org/de/marketplace/details?id=3
[2] https://automate.io/integrations
[3] https://www.humhub.org/de/marketplace/details?id=25
[4] https://unito.io/

Slack[5]) und viele weitere Programme. Mithilfe dieser Integrationen können ggfs. bislang benutzte Tools abgelöst oder unter Verwendung eines Chatbots intelligent verbunden werden (Back 2016).

Darüber hinaus sind Benutzerprofile ein zentraler Bestandteil eines ESN. In diesen können neben den Stammdaten des Mitarbeiters weitere Eigenschaften hinterlegt werden; beispielsweise Informationen aus dem vorhandenen Active Directory (AD) des Unternehmens oder benutzerspezifische Daten, wie die Kompetenzen des Angestellten (Koch und Richter 2009).

### 2.1.1 Datenquellen im Unternehmen

Neben den Informationen im ESN stehen weitere Datenquellen im unternehmensinternen Intranet zur Verfügung, die zur Verbesserung bzw. Automatisierung von Arbeitsprozessen mithilfe von Chatbots herangezogen werden können.

Beispielhaft stehen folgende Informationsquellen unternehmensintern zur Verfügung:

- **Active Directory**: Eigenschaften über Mitarbeiter, z.B. Geschlecht, Standort (Büro, Niederlassung), Sprache, benutzerdefinierte Attribute
- **E-Mail Server**, z.B. Exchange: Zugriff auf Kalender, E-Mails
- **Enterprise-Resource-Planning (ERP) System**, z.B. SAP: Warenein- und -ausgänge
- **Customer Relationship Management-Systeme**, z.B. Salesforce: Laufende Kundenprojekte, Termine, Meilensteine
- Informationen über **Produkte, Roadmaps**
- **ESN**: Mitarbeiter-Profile mit Kompetenzprofil, Telefonnummer, Ausbildung etc.
- **Internes Wiki**: Wissensbasis
- **Dokumentenmanagementsystem**

Bei der Verwendung dieser Datenquellen muss drauf geachtet werden, dass keine sensiblen Informationen nach außen gelangen und die Zugriffsrechte eingehalten werden. Ein Angestellter ohne Leitungsbefugnis sollte nicht auf Daten zugreifen dürfen, die Abteilungsleitern vorbehalten sind. Die Zugriffsrechte müssen entsprechend im Bot inkludiert und mit dem Rechteverwaltungssystem (z.B. vom AD) abgeglichen werden (De Capitani di Vimercati, Paraboschi und Samarati 2003).

[5] `http://coenraets.org/blog/2016/01/slack-salesforce-integration/`

## 2.2 Natural Language Processing / Understanding

Natural Language Processing (NLP) ist ein Themengebiet, welches sich mit der Verarbeitung von maschinellen Daten von und zu natürlicher Sprache auseinander setzt (Chowdhury 2003). NLP umfasst dabei Natural Language Understanding (NLU) (Schank 1972), womit menschliche Sprache in maschinenverständliche Anweisungen übersetzt wird, sowie Natural Language Generation (NLG), welches aus Daten sprachlich semantisch und syntaktisch sinnvollen Text generiert (Reiter und Dale 2000).

Für NLU, also dem Verständnis von gegebenem Text durch eine Maschine, werden Techniken wie Pruning (Entfernen unspezifischer Wörter, wie z.B. Stopwörter), Stemming (Grundformreduktion) und Part-of-speech (POS)-Tagging (Brants 2004) angewandt, sodass am Ende ein standardisierter Satz entsteht. Dieser wird klassifiziert, wodurch Intention und Datenobjekte einer Eingabe erkannt werden. Hierfür wird mittels Machine Learning (ML) ein Modell aus verschiedenen Trainingsdaten generiert, welches unbekannten Sätzen Wahrscheinlichkeiten für vorgegebene Intentionen zuordnet und so eine Klassifizierung vornimmt (Kuhn und De Mori 1995). Diesen Vorgang nennt man „Intent Classification", wobei die Klassen „Intents" genannt werden. Zusätzlich kann man Entitäten („Entities") bestimmen, die im Trainingsdatensatz als Variablen behandelt und später erkannt werden sollen. Den Vorgang der Entitätsbestimmung nennt man Named-Entity Recognition (NER) (Mohit 2014).

Als Beispiel werden die Sätze „Ich suche nach einem Ort im Norden der Stadt" und „Welcher Ort befindet sich im Norden" der Intention „Ortssuche" zugeordnet. Eine Entität kann „Norden" als Variable für die Himmelsrichtung sein. Wenn später der Satz „Ich suche nach einem Ort im Süden" analysiert wird, wäre nun „Süden" die Entität bei gleicher Intention „Ortssuche". Das NLU-Tool liefert Intention und ggfs. Entitäten, worauf basierend der Chatbot entsprechende Antwortsätze wählt, die zur erkannten Intention, zu den Entitäten und zum aktuellen Dialogzustand passen (Braun u. a. 2017).

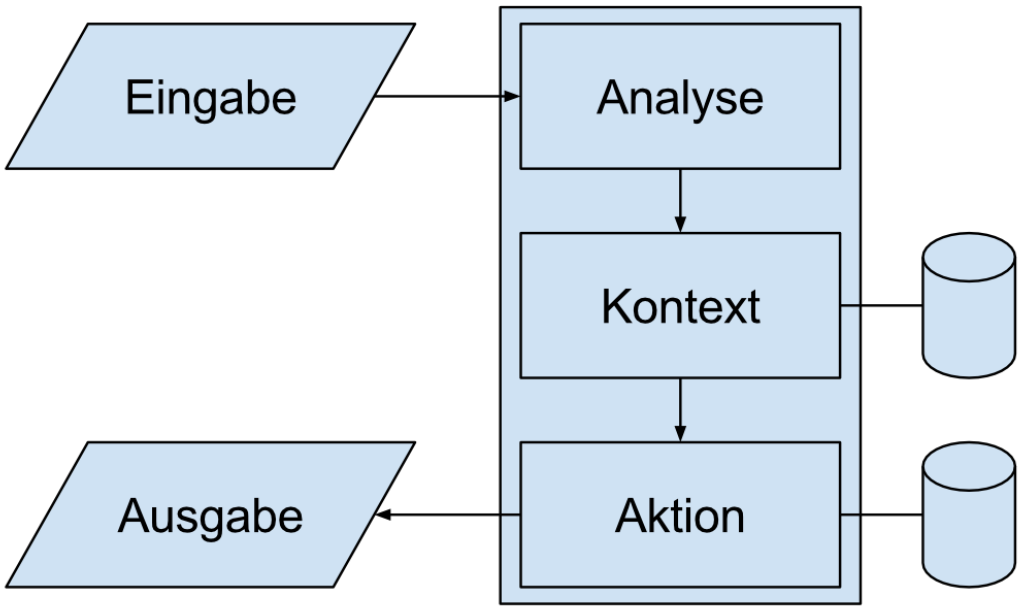
Die Erkennung von Entitäten erfolgt entweder durch das Markieren in den Trainingsdaten oder mithilfe von NER-Tools wie Duckling[6], welches beispielsweise numerische Werte erkennt und in einen geeigneten Datentyp überführt. Die erkannten Dimensionen umfassen Zeit- und Temperaturangaben, Zahlen (ausgeschriebene Zahl zu Integer bzw. dezimal), Entfernungen, Volumen, Zeitdauer, E-Mail Adressen, Telefonnummern und Webadressen.

[6] https://duckling.wit.ai/

## 2.3 Chatbots

Chatbots sind textbasierte Dialogsysteme, die mit menschlichen Benutzern in deren Sprache interagieren, indem sie mithilfe von definierten Regeln auf Benutzeraktionen reagieren (Shawar und Atwell 2007b).

Nach heutigem Entwicklungsstand beinhalten sie einen Interpreter, der eingehende Nachrichten analysiert und einen Kontextspeicher, der mit einem Key-Value-Store verbunden ist und zu Benutzern zugeordnete Daten speichern kann. Auf Basis einer Chatbot Scripting Language, welche die Logik des Chatbots darstellt, können aufgrund der gegebenen Interpretation der Eingabe, sowie des aktuellen Kontextes, die Aktionen des Chatbots ausgewählt werden (Cahn 2017). Darüber hinaus bieten die Interpreter NLU-Möglichkeiten in Form von Klassifizierung, NER, regulären Ausdrücken und Substitutionen (beispielsweise steht ”I’m” für ”I am” oder ”NYC” für ”New York City”). Entwicklungswerkzeuge, um Chatbots zu definieren, nennt man auch „Chatbot Engine” (Shawar und Atwell 2004).




*Quelle:* Eigene Darstellung
Abbildung 2.1: Struktur eines Chatbots

Die wichtigste Vorbereitung zur Entwicklung eines Chatbots ist, dass dessen Funktion, sowie die zur Erfüllung des Ziels benötigten Daten und Gesprächsabläufe, vorab definiert werden. Anders, als im unternehmensfernen Kontext, sind in der unternehmensinternen Nutzung keine generalisierten Chatbots gefragt, also keine die allwissend sind und alle Fragestellungen beantworten können, sondern unternehmensspezifische, die beispielsweise eine Person mit bestimmten Kompetenzen suchen oder automatisiert mithilfe von E-Mail Inhalten Schlüsse ziehen können.

Chatbots kommunizieren mit den Nutzern. Bei alternativen Interaktionsformen, wie Formulare oder Apps, wird die Benutzereingabe nicht analysiert, sodass der Benutzer auf definierte Fragen nur mit vorgegebenen Antwortmöglichkeiten reagieren kann. Mit Chatbots hat der Anwender die Möglichkeit, einen umfassenden Datenstamm aufzubauen, welcher auf vergangene Gespräche Bezug nehmen und so Sequenzen konstruieren kann, die für Formulare unmöglich abzubilden wären. Durch die Texterkennung lassen sich zusätzlich Muster in den Benutzereingaben feststellen, die nicht mit statischen Methoden feststellbar wären, wie beispielsweise die Stimmung. Durch Texterkennung kann man, wenn man keine weiteren Daten hat, zudem die benutzte Sprache erkennen (Tromp und Pechenizkiy 2011) .

Chatbots lassen sich darüber hinaus sehr einfach in bestehende Messaging-Plattformen integrieren. Somit wird kein weiteres Tool für die tägliche Arbeit benötigt, sondern die bisher Benutzten verwendet und um ein Assistenzsystem erweitert. Aufgrund der guten Integrationsmöglichkeiten von Chatbots in die bestehenden Messaging-Systeme von Unternehmen, sowie der Zugriffsmöglichkeit auf Unternehmensressourcen und intelligenter Bündelung dieser, lässt sich im besten Fall die Anzahl der für die Arbeit benötigten Programme verringern. Dadurch verbessert sich die digitale Arbeit und die Mitarbeitermotivation kann gesteigert werden (Eckmüller 2016). Der Einsatz von Chatbots ist allerdings umstritten - vor allem, wenn Nachrichten „in die Cloud" gelangen und das Unternehmen die Kontrolle über die Daten verliert (Boys und Crawford 2012).

### 2.3.1 Aktueller Entwicklungsstand

Aktuell gibt es unzählige Chatbots und Chatbot-Engines auf dem Markt. Zudem gibt es verschiedene Ansätze für die Modellierung des Dialogs, die jedoch zumeist auf Artificial Intelligence Markup Language (AIML) aufbauen. AIML basiert auf Extensible Markup Language (XML) und beschreibt die Eingabe- („pattern", Eingabesatz des Nutzers) sowie Ausgabesätze („template", Reaktion des Chatbots) (Marietto u. a. 2013).

Der folgende Code modelliert beispielsweise die statische Antwort (template) auf eine definierte Frage (pattern):

```
<category>
    <pattern>WHAT ARE YOU</pattern>
    <template>
        I am the latest result in artificial intelligence,
        which can reproduce the capabilities of the human brain
        with greater speed and accuracy.
    </template>
</category>
```

Generell wird strikt in Eingabe, Ausgabe (was der Bot tun soll) sowie dem Message-Flow getrennt (vgl. Shawar und Atwell 2007a, S. 43ff). Die Eingabe enthält entweder die unbearbeitete Benutzereingabe oder die Intention und Entitäten eines NLU Interpreters. Die Ausgabe kann entweder ein Antwortsatz oder eine anwendungsspezifische Funktion sein, mit der beispielsweise Daten aus einer entfernten Datenbank abgefragt werden. Der Message-Flow wird mithilfe einer Chatbot Scripting Language definiert, wobei AIML, RiveScript, ChatScript und SuperScript die jeweils bekanntesten Vertreter dieser Beschreibungssprachen sind (Gregori 2017).

# 3 Methode

Diese Bachelorarbeit folgt dem Paradigma der Designforschung, die — im Gegensatz zur Verhaltensforschung — Lösungen erforscht, womit spezifische Problemstellungen mithilfe von existierenden Theorien und Wissen gelöst werden können (Peffers u. a. 2007). Durch die Entwicklung von neuen Technologien werden somit die Möglichkeiten von Organisationen und Menschen erweitert. Die Verhaltensforschung wiederum ergründet das von der IT beeinflusste Verhalten von Menschen bzw. Organisationen (Hevner u. a. 2004).

## 3.1 Methode der Anforderungserhebung

Basierend auf dem Wissensstand der Grundlagen von ESN, NLP und Chatbots werden Anforderungen analysiert, die im Kontext der Implementierung eines Chatbots in Unternehmen betrachtet werden müssen. Die Auswahl eines ESN beschränkte sich auf eine quelltextoffene Lösung aufgrund der festgestellten unternehmensspezifischen Anforderungen sowie der im späteren Verlauf der Bachelorarbeit zu entwickelnden Middleware zur Verbindung des ESN und des Chatbots.

Bei der Auswahl einer geeigneten Chatbot-Engine wurden zuerst alle bereits existierenden Lösungen recherchiert, gemeinsame Auswahlkriterien definiert und schließlich, basierend auf den unternehmensspezifischen Anforderungen, ebenso eine quelltextoffene Lösung gewählt, die gute Integrationsmöglichkeiten bietet, eine gute Dokumentation bereitstellt und einer aktiven Entwicklung unterliegt. Die analysierten Einsatzszenarien wurden auf Basis der vorab definierten Datenquellen sowie der Möglichkeiten des ESN ausgewählt und weiter definiert.

## 3.2 Konzeptionierung, Implementierung

Das Konzept wird designorientiert betrachtet, wobei die Gesamtlösung in die Konzepte des ESN, der Middleware und des Chatbots unterteilt ist. Hinsichtlich der Entwicklung des Chatbots sowie des nötigen NLU Interpreters wird auf Grundlage des Funktionsumfanges und der zugehörigen Dokumentationen eine Lösungsvariante vorgeschlagen, welche die Anforderungen der beiden vorher definierten Einsatzszenarien erfüllt. Darauf basierend kann eine Middleware konzipiert werden, die das ESN mit dem Chatbot verbindet.

Die Implementierung des entwickelten Konzepts erfolgt sequentiell nach dem Teilschritt der Gesamtlösung. Als erstes wird mithilfe der Middleware die Verbindung zwischen ESN und Chatbot hergestellt, wodurch gleichzeitig eine Interaktionsschnittstelle zum Testen des Chatbots entsteht und man die dadurch gewonnene Erfahrung in die weitere Implementierung einfließen lassen kann. Danach erfolge die schrittweise Umsetzung der Einsatzszenarien, indem die möglichen Intentionen des Nutzers, erwartete Aktionen des Chatbots sowie die hierfür benötigten Benutzerdaten mit einer jeweiligen Quelle definiert, Trainingsdaten der Benutzereingaben erstellt und grundlegende Aktionen zu bestimmten Benutzerintentionen zugeordnet werden. Mithilfe dieses Grundgerüstes werden verschiedene Dialoge getestet und ausgewertet, um so die Trainingsdaten sowie das Dialogmodell zu erweitern und weitere Fehler zu beheben.

# 4 Analyse

Mithilfe der Grundlagen aus Kapitel 2 werden die Anforderungen an ESN und Chatbot analysiert und zwei Anwendungsfälle eines Chatbots im unternehmensinternen Einsatz definiert.

## 4.1 Anforderungen

Eine zentrale Anforderung ist, dass die Daten das Unternehmen nicht verlassen. Es wird also auf eine Lösung Wert gelegt, die man auf einem Server des Unternehmens installieren kann und keine sensiblen Kommunikationsdaten nach Außen sendet. Hierfür ist es hilfreich, Lösungen mit offenem Quelltext (Open Source) zu verwenden, da man so jeden Arbeitsschritt nachvollziehen und kontrollieren kann.

### 4.1.1 Enterprise Social Network

Der Markt gibt einige ESN Produkte her - Beispiele sind Slack, Basecamp, Asana, Atlassian Hipchat, Microsoft Teams und viele weitere. Einige davon sind Allrounder, wie z.B. Basecamp, andere spezialisieren sich auf eine bestimmte Funktion, wie z.B. Slack (Messaging) oder Asana (Projektmanagement).

Nur die wenigsten sind uneingeschränkt kostenlos nutzbar. Ein ESN, welches kostenfrei und quelloffen verfügbar ist, ist Humhub. Anders als bei den meisten Social Networks gibt es keine Gruppen, sondern sogenannte Spaces, in denen sich Nutzer in ihrem Interessengebiet austauschen können. Das Unternehmen kann individuelle Profilfelder pro Benutzer definieren, sodass jedes Mitglied beispielsweise die eigenen Kompetenzen eintragen kann. Die Nutzerregistrierung kann wahlweise manuell erfolgen oder automatisiert durch eine AD Infrastruktur. Außerdem gibt es zahlreiche Module für Humhub, die das ESN um weitere

Funktionen, wie einen Instant Messenger, einen Kalender, ein Wiki oder ein Umfrage-Tool, erweitert[1].

Aufgrund der Quelloffenheit, der aktiven Weiterentwicklung und die Möglichkeit der Erweiterung mittels Plugins wird dieses ESN Grundlage dieser Bachelorarbeit sein.

### 4.1.2 Chatbot

Im Zuge der Bachelorarbeit wurden die bekanntesten Chatbots recherchiert (siehe Tabelle 4.1). Die Kategorisierung erfolgte nach offenem Softwarequelltext („Open Source"), offenen Schnittstellen zu Messaging-Diensten („Open Services") sowie der Möglichkeit, das Programm auf eigenen Servern bereitzustellen („On-Premises").

Aufgrund der aufgestellten Anforderung, dass die benutzten Lösungen auf eigenen Servern gehostet werden sollen („On-Premises"), entfallen SaaS Anwendungen, die bei externen IT-Dienstleistern betrieben werden. Weiterhin soll die Software quelloffen sollen, weswegen auf Open-Source Chatbots eingeschränkt wird.

Lita[2], Errbot[3], Bottr[4], Botkit[5], Hubot[6] und Program-O[7] besitzen keine NLU-Möglichkeiten, sodass alle Eingabesätze fest trainiert oder zumindest durch einen regulären Ausdruck hinterlegt werden müssen - ein Maschine Learning Ansatz fehlt. Die Tools könnten allerdings um einen NLU Interpreter erweitert werden.

ChatterBot[8] besitzt keine Abstraktion zwischen Eingabesätzen und Conversation Flow, was das Training erschwert. RiveScript[9], ChatScript[10] und SuperScript[11] sind Chatbot Scripting Languages, welche AIML erweitern oder ersetzen - die nötigen Module zum sinnvollen Einsatz als unternehmensinterner Chatbot (Kontext-Speicher, Integrationsmöglichkeiten) fehlen und müssten entwickelt werden.

[1] https://www.humhub.org/de/marketplace
[2] https://docs.lita.io/getting-started/
[3] http://errbot.io/en/latest/
[4] https://bottr.co/
[5] https://github.com/howdyai/botkit/blob/master/docs/readme.md
[6] https://hubot.github.com/docs/
[7] https://github.com/Program-O/Program-O/wiki
[8] http://chatterbot.readthedocs.io/en/stable/
[9] https://www.rivescript.com/about
[10] https://github.com/bwilcox-1234/ChatScript
[11] http://superscriptjs.com/

| Name | Open Source | Open Services | On-Premises |
|---|---|---|---|
| Botkit | X | X | X |
| Botpress | X | X | X |
| Bottr | X | X | X |
| Chatfuel | - | - | - |
| ChatScript | X | X[1] | X |
| ChatterBot | X | X | X |
| Dexter | - | - | - |
| Dialogflow | - | - | - |
| E.D.D.I. | X | X | X |
| Errbot | X | X | X |
| Facebook Bots | - | - | - |
| Hubot | X | X | X |
| IBM Watson Conversation Service | - | X | X[2] |
| IKY | X | X | X |
| Lita | X | X | X |
| Microsoft Bot Framework | X | X | X[2] |
| Pandorabots | - | - | - |
| Program-O | X | X[1] | X |
| Rasa Core | X | X | X |
| Rebot | - | - | - |
| RiveScript | X | - | X |
| Superscript | X | - | X |
| wit.ai | - | X | - |

[1] Anbindung von Services nicht nativ in Form von Plugin-Modell, Middleware notwendig

[2] Bot-Implementierung On-Premise, Chatbot-Framework Software as a Service (SaaS)

Tabelle 4.1: Chatbot Übersicht (Stand 30.01.2018)

IKY[12] und E.D.D.I.[13] fehlt es an einer umfangreichen Dokumentation, wodurch der Einstieg erschwert wird. Botpress[14] ist eher auf Facebook als primäre Zielgruppe ausgelegt. Microsoft Bot Framework[15] ist zu sehr mit dem SaaS-Tool Cortana verknüpft, sodass zur Klassifizierung alle Eingabesätze hochgeladen werden - was nicht den gesetzten Anforderungen der Arbeit entspricht.

Aufgrund der getroffenen Vorauswahl stellt sich Rasa Core als System heraus, welches von den gleichen Machern wie der NLU Interpreter Rasa NLU ist. Dadurch ist eine nahtlose Integration untereinander möglich. Rasa Core empfängt den Eingabesatz, sendet es an Rasa NLU, erhält dadurch Intention und Entitäten und findet die zugehörige Aktion je nach Kontext. Zudem stehen eine gute Dokumentation, umfangreiche Plugin-Möglichkeiten und eine große Community dahinter[16].

## 4.2 Use Cases

Aufgrund der in Kapitel 2.1.1 genannten Datenquellen lassen sich viele verschiedene Use-Cases definieren, die stets das Ziel verfolgen, ein externes Tool bzw. die Datenquelle aus dem Messaging-Dienst heraus zu beeinflussen.

Im Zusammenhang mit einen ERP-System könnte die Softwarelösung bei der Lieferung von Inventar automatisiert eine Nachricht über den Inhalt der Lieferung verfassen. Ein anderer Use-Case im Zusammenhang mit ERP-Systemen wäre ein Request-Approval-Modell für die Anfrage von bestimmten Equipment sowie der Zustimmung hierfür von der entsprechenden Person mit Befugnis - alles über den Messaging-Dienst statt dem ERP-System.

Bei Customer-Relationship-Management (CRM) Systemen könnten Aktualisierungen von Kundenprojekten an alle betroffenen Personen im Messaging-Dienst weitergeleitet werden, ähnlich der Integration von beispielsweise Github in Slack: Hier können alle Github-Neuigkeiten wie Pull Requests oder neuen Issues eines Repositories in einen Slack-Channel kommuniziert werden. Die Nutzer können daraufhin die Änderung kommentieren und so neue Inhalte zum CRM hinzufügen .[17]

[12] https://github.com/alfredfrancis/ai-chatbot-framework
[13] https://labsai.atlassian.net/wiki/spaces/EDDI
[14] https://botpress.io/
[15] https://dev.botframework.com/
[16] https://core.rasa.ai/
[17] https://github.com/integrations/slack

Aber auch ohne eine Datenquelle lassen sich Use-Cases finden, die den Unternehmensalltag erleichtern. Beispielsweise hat die Firma „avocado" einen Chatbot für ein internes Event der Firma Zalando konzipiert, der die Angestellten vor und während der Veranstaltung unterstützte, indem Informationen zum Veranstaltungsort, zum Ablauf hinsichtlich der Vorträge und zu weiteren Dingen bereitgestellt wurden - beispielsweise das WLAN-Passwort, Standort der Toiletten und Fragen zur Verpflegung. Falls der Chatbot keine Antwort zur Frage geben konnte, wurde automatisch ein menschlicher Mitarbeiter zur Hilfe hinzugezogen. Das Ergebnis: Es gab 466 Anfragen an den Bot mit einer Erfolgsquote von 98%. Ein menschlicher Mitarbeiter musste nie einbezogen werden. Der Vorteil gegenüber einer eigenen Website oder App: Die Benutzer können die Infos, die sie wirklich benötigen, schnell erhalten, anstatt auf der Website oder in der App zu suchen. Zudem können sie direkt im gleichen Programm mit den Kollegen chatten. Bei Verwendung einer eigenen App nur für das Event müsste man sich erst die App installieren - dies entfällt beim Einsatz eines Chatbots (Day 2017).

Um den Rahmen dieser Arbeit nicht zu übersteigen, werden die beiden folgenden Use Cases umgesetzt.

### 4.2.1 Kalender

Der Aufwand, einen freien Termin unter mehreren Kollegen zu finden, soll durch diesen Bot minimiert werden. Aktuell würde die Suche nach einem freien, passenden Termin zwischen Mitarbeiter A und Mitarbeiter B so verlaufen, dass A in seinen Kalender schaut, freie Termine für die nächsten Tage sucht und diese B vorschlägt. B würde daraufhin seinen Kalender öffnen und nachsehen, ob die vorgeschlagenen Termine bei ihm ebenfalls verfügbar sind. Wenn nicht, würde er an einem anderen Tag suchen und der Prozess würde sich analog wiederholen. Der Aufwand steigt dabei exponentiell mit jedem weiteren Kollegen.

Diesen Prozess könnte man mit einer einfachen Abfrage auf die Kalender der beteiligten Personen stark vereinfachen. Ein Chatbot könnte auf die Kalender zugreifen und freie Termine abstimmen. Im freien Markt gibt es hierfür bereits Tools und Apps wie „Doodle"[18], „Kulibri"[19], „Dudle"[20] oder der direkt für Unternehmen ausgelegte „DFN Terminplaner"[21], die jedoch ein weiteres Tool nur für diese eine Verwendung darstellen und damit die Komplexi-

[18] https://doodle.com/de/
[19] http://kulibri.com/
[20] https://dudle.inf.tu-dresden.de/?lang=de
[21] https://www.dfn.de/dienstleistungen/dfnterminplaner/

tät der verwendeten Programme erhöht. Der Vorteil des Chatbots liegt darin, dass man direkt im Messaging-Dienst, in dem man sich währenddessen ohnehin mit den Kollegen unterhält, den freien Termin finden und einen Kalendereintrag hierfür erstellen lassen kann.

In der freien Wirtschaft wird für die Termin- und Raumbuchung überwiegend Microsoft Exchange benutzt, auf das in dieser Bachelorarbeit jedoch nicht zurückgegriffen werden kann. Bei einer Umsetzung mit Microsoft Exchange könnte man neben den freien Terminen in den Kalendern zusätzlich einen freien Raum für diese Zeit suchen oder den Status eines spezifischen Raumes abfragen und in den Terminvorschlag einfließen lassen. Denkbar wäre auch die Arbeitszeiten abzufragen und zusätzlich zu beachten.

### 4.2.2 Kompetenzen

Die Suche nach Personen im Unternehmen mit bestimmten Kompetenzen gestaltet sich umso schwieriger, je mehr Personen im Unternehmen arbeiten, obwohl die Anzahl und Vielfältigkeit der Kompetenzen steigt. Das Resultat ist, dass Aufgaben unzureichend erledigt werden oder unbearbeitet bleiben, nur, weil man keine Kenntnis der Kompetenzen der Mitarbeiter besitzt. Aktuell kann man nur durch eine umfangreiche Befragung oder durch Zufall herausfinden, dass der Kollege aus der anderen Abteilung die Kompetenz besitzt, die man für eine bestimmte Aufgabe benötigt. Diese Information könnte sowohl für die Hilfesuche, als auch für die interne Personalrekruitierung und -entwicklung nötig sein.

Wenn man die Benutzerprofile von Humhub um Kompetenzen erweitert, kann man Schlüsselwörter aus Chatnachrichten extrahieren und diese den geforderten Kompetenzen zuordnen, um so schnell eine geeignete Person zu finden. Der Vorteil des Einsatzes des Chatbots gegenüber der manuellen Variante (Umfrage im Unternehmen, E-Mail an alle Angestellten, etc.) liegt hier klar in der Geschwindigkeit sowie der globalen Suche.

In den Unternehmen wird überwiegend Microsoft AD eingesetzt, mit welchem man benutzerspezifische Daten, wie beispielsweise Kompetenzen, hinterlegen kann. Man könnte die Daten also, anstatt aus Humhub, direkt aus dem AD abfragen. Weiterhin sollte es möglich sein, den Online-Status der Person in Betracht zu ziehen und so ausschließlich oder vorrangig Nutzer vorzuschlagen, die zwar nicht zu 100% die geforderten Kompetenzen erfüllen, jedoch in dem Moment online sind.

# 5 Konzept

Aufgrund der vorab definierten Anforderungen und Use Cases gilt es ein Konzept zu entwickeln, Humhub um die Integration eines Chatbots zu erweitern. Der Chatbot soll schließlich mit den Nutzern kommunizieren und auf Daten von Humhub zurückgreifen können.

## 5.1 Middleware

Bevor man die Use-Cases umsetzen kann, müssen zuerst ESN, Chatbot und NLU verbunden werden.

Inhalte müssen aus dem ESN zum NLU gegeben werden, um Intention und Entitäten zu klassifizieren. Diese sollen dann an den Chatbot gegeben werden, um Reaktion zu finden. Schließlich soll die Antwort des Chatbots wieder in das ESN fließen. Rasa Core bietet bereits eine Integration für Rasa NLU[1], sodass während der Implementierung lediglich die Verbindung zwischen Humhub und Rasa Core entwickeln werden muss.

Da das Mail-Modul von Humhub keine Application Programming Interface (API) mitbringt[2] müssen die Datenbank, in der alle Nachrichten gespeichert sind, beobachtet, neue Nachrichten automatisch an Rasa Core weitergeleitet und die Antworten von Rasa Core in der Datenbank gespeichert werden. Deswegen wird zur Umsetzung der Use Cases eine Middleware benötigt, die die Kommunikation zwischen Humhub und Chatbot gewährleistet. Die Nachrichten müssen an eine Sender-ID geknüpft werden, sodass eine Zuordnung der Antwortsätze auf die richtige Konversation erfolgen kann. Eine wichtige Anforderung hierbei ist, dass die Datenübermittlung ausschließlich innerhalb des Unternehmens passiert, also keine Daten beispielsweise für die Klassifikation, ins Internet gesendet werden müssen.

[1] https://core.rasa.ai/interpreters.html
[2] https://github.com/humhub/humhub/issues/643

Außerdem sollte der Chatbot als Middleware-Lösung mit verschiedenen ESN bzw. auch Instant Messaging Services und NLU Interpretern kompatibel sein, da diese meist nur wenige Sprachen unterstützen und so für andere Umgebungen andere NLU Werkzeuge eingesetzt werden müssen. Als Programmiersprache wird die des ausgewählten Chatbots gewählt, um zwischen den implementierten Chatbot-Aktionen und den Middleware-Funktionen Homogenität herzustellen.

## 5.2 Chatbot: Rasa Core

Rasa Core ist ein Open Source Tool für das Dialogmanagement. Es kann mit verschiedenen NLU Interpretern zusammenarbeiten und implementiert einen Tracker, der pro Konversation die Zustände in Form des Nachrichtenverlaufs, den letzten Bot-Aktionen sowie Slots speichert. Slots sind Speicherplätze für Variablen, die vom Benutzer während der Konversation genannt und extrahiert werden. Beispiele sind ein Stadtname bei einer Routenbestimmung, Zeitangaben oder Namen anderer Personen (Bocklisch u. a. 2017).

Der Chatbot muss um die Funktionalität erweitert werden mit der Middleware zu kommunizieren und als Reaktion auf bestimmte Eingabesätze Unternehmensressourcen heranzuziehen, wie beispielsweise den Kalender oder eine Kompetenz-Matrix. Da die Aktion und der Antwortsatz direkt im Chatbot generiert wird ist dies Aufgabe des Chatbots statt der Middleware.

Schließlich müssen verschiedene Dialog-Abläufe für die Use-Cases gefunden werden, die möglichst viele Fälle abdecken, die vom „happy path" abweichen. Der „happy path" stellt dabei den Dialogfluss dar, welcher zum gewünschten Ergebnis führt. In der Praxis kann dieser Pfad jedoch nur selten eingehalten werden, da die Benutzer eventuell eine Änderung vornehmen möchten oder Informationen in unvorhergesehener Kombination und Reihenfolge übermitteln.

Innerhalb des Dialoges werden dabei klassifizierte Intentionen und extrahierte Entitäten von Rasa NLU sowie Slotwerte und Aktionsnamen von Rasa Core verarbeitet. Im Zuge des Trainings wird mithilfe der Trainingsdatensätze und der Open Source Bibliothek „Tensorflow" Maschine Learning eingesetzt, um die Dialoge in einzelne Berechnungsgraphen zu zerlegen, zusammenzufügen und anhand dessen ein Modell zu trainieren. Mithilfe dieses Modells kann der Chatbot in Dialog-Zuständen, die vom „happy path" abweichen, Wahrscheinlichkeiten für die vom Benutzer gewünschte Reaktion des Chatbots berechnen.

Essentiell wichtig ist die richtige Zuordnung zu den Gesprächspartnern, sodass die Datensicherheit gewährleistet wird und der Kontext der einzelnen Gespräche erhalten bleibt. Hierfür muss die Middleware eine Sender-ID weitergeben und Rasa Core diese mit der Antwortnachricht zurücksenden. Aufgrund der prototypischen Umsetzung des Chatbots und der ausschließlich netzinternen Übertragung der Nachrichten wird auf eine Verschlüsselung verzichtet, sollte im produktiven Einsatz jedoch implementiert werden.

Der zu entwickelnde Chatbot wird „Humbot" genannt und bedient beide vorher definierten Use-Cases.

## 5.3 NLU: Rasa NLU

Rasa NLU ist ein Open Source Tool für die Klassifzierung der Intention einer Nachricht und Extraktion zugehöriger Entitäten. Es basiert auf den „FastText" Ansatz, der Trainingssätze mit zugehörigen Intentionen als Vektoren repräsentiert und mithilfe eines Binary Trees hierarchisch klassifiziert (Joulin u. a. 2016).

Dabei implementiert Rasa NLU eine Pipeline für mehrere NLP Tools, wie zum Beispiel Spacy[3] (POS Tagging, NER), MIT Information Extraction (MITIE)[4] (Alternative zu Spacy), ner_synonyms (NER, Ersetzen von Synonymen) und ner_duckling[5] (NER, Umwandlung von Text zu numerischen Daten wie z.B. Zeitangaben, Temperaturen, etc.).

Im Zuge der Implementierung muss Rasa Core zur Verwendung mit Rasa NLU konfiguriert und Trainingsdaten für die Klassifikation der Intentionen und Entitäten ausgearbeitet werden. Die Auswahl und Benennung der Intentionen sowie die Zuordnung der Trainingssätze zu diesen sollte so erfolgen, dass sie zu den gestellten Nutzungsszenarien passen.

Außerdem ist zu beachten, dass möglichst viele Variationen der Eingabesätze trainiert werden, um die Genauigkeit der Klassifizierung zu maximieren. Ebenso müssen Beispiele für Entitäten, die je nach Situation unterschiedlich ausfallen und durch Rasa Core verarbeitet werden müssen, und die möglichen Positionen dieser markiert werden.

[3] https://spacy.io/
[4] https://github.com/mit-nlp/MITIE
[5] https://duckling.wit.ai/

# 6 Implementierung

Auf Basis des Konzeptes werden die Middleware, der Kanal zum Chatbot sowie der Chatbot selbst implementiert. Dabei ist die Middleware generalisiert gehalten, sodass auch andere Dienste und Chatbots angebunden werden können. Die Umsetzung der Use-Cases als Chatbot erfolgt spezifisch für Rasa Core bzw. Rasa NLU, wobei die Trainingsdaten auch für andere Chatbots und NLU Systeme verwendet werden können.

## 6.1 Rasahub

Es wurde eine Middleware umgesetzt, welche sich mit der Humhub-Datenbank verbindet, die ID der neuesten Nachricht abfragt, speichert und kontinuierlich die neueste ID abfragt und vergleicht. Wird nun eine ID gefunden, die von der gespeicherten abweicht, werden die neuen Chat-Nachrichten abgefragt und der Beginn der Nachricht überprüft. Sollte die Nachricht mit einem Bot-Trigger, der konfiguriert werden kann, übereinstimmen, wird die Nachricht per Socket-Verbindung an den RasahubInputChannel gegeben. Hierfür muss eine Server-Socket-Verbindung eingerichetet werden, womit sich RasahubInputChannel als Client verbinden kann.

Damit der Chatbot als Benutzer auftreten kann, benötigt man einen entsprechenden Benutzeraccount in Humhub. Dieser muss dort der Benutzergruppe ”Bots” zugeordnet sein.

Rasahub kann direkt aus dem Python Paketmanager (Python Package Index, PyPI) installiert werden:

```
$ pip install rasahub
```

Es wird automatisch Rasa_Core, Rasa_NLU und ein MySQL-Connector installiert. Alternativ kann Rasahub über ein öffentliches Github Repository bezogen werden[1].

[1] https://github.com/DServSys/Rasahub

Folgende Parameter sind nun zu berücksichtigen:

| Parameter | Langform | Beschreibung | Erforderlich | Standardwert |
|---|---|---|---|---|
| -dbu | –dbuser | Datenbank Benutzername | X | |
| -dbp | –dbpwd | Datenbank Benutzerpasswort | X | |
| -dbh | –dbhost | Datenbank Hostadresse | - | 127.0.0.1 |
| -dbn | –dbname | Datenbank Name | X | |
| -t | –trigger | Trigger Zeichenkette | - | !bot |
| -rh | –rasahost | Rasa Hostadresse | - | 127.0.0.1 |
| -rp | –rasaport | Rasa Port | - | 5020 |

Tabelle 6.1: Konfigurationsparameter Rasahub

Rasahub kann direkt von der Kommandozeile gestartet werden. Ein beispielhafter Aufruf könnte so aussehen:

```
$ rasahub -dbu humhubuser -dbp secretpassword -dbn humhub
```

Nun wird auf eine Verbindung vom RasahubInputChannel gewartet:

```
Connected to database.
Waiting for socket connection from Rasa on port 5020
```

## 6.2 RasahubInputChannel

RasahubInputChannel verbindet sich per Socket-Verbindung zu Rasahub und verwaltet die Nachrichten zum und vom Chatbot. Ein InputChannel hat nach Maßgabe von Rasa-Core die Klassen „InputChannel” und „OutputChannel” zu definieren, wobei der „InputChannel” auf den „OutputChannel” in der Methode „_record_messages” zugreift. Weiterhin muss der Inputchannel die Methoden „__init__”, „start_async_listening” und „start_sync_listening” implementieren.

In der „_record_messages” Methode wird die Nachricht der Socket-Verbindung empfangen. Im Outputchannel ist neben „__init__” die Methode „send_text_message” nötig, in der die Antwort von Rasa-Hub an die Socket-Verbindung gesendet wird.

Sobald der InputChannel über das „run.py” Skript initialisiert wurde verbindet sich Rasa_Core mit Rasahub:

```
Connection established: ('127.0.0.1', 47794)
```

Bei jeder neuen Nachricht in Humhub erhält man Debug-Informationen in Rasahub:

```
Input from db: {
  u'message': u'Ich habe ein Problem mit meiner Tastatur',
  u'message_id': 4
}
Reply from rasa: {
  u'reply': u'Christian Frommert könnte bei dem Anliegen helfen.',
  u'message_id': 4
}
```

## 6.3 Chatbot

Es wird ein Chatbot für alle Use-Cases entwickelt, anstatt jeweils einen Chatbot pro Use-Case. Dementsprechend sind die Intentionen der Eingabesätze der Benutzer klar abzugrenzen und dedizierte Aktionen zu implementieren. Im Folgenden wird auf die benötigten Daten eingegangen.

### 6.3.1 Domain

Die Domain-Datei beschreibt die grundlegende Struktur des Chatbots. Sie beinhaltet alle Slots, Intentionen („intents”), Entitäten („entitites”), Nachrichten-Vorlagen („templates”) sowie Aktionen („actions”) [2].

Slots werden benutzt, um Daten während des Dialoges zu speichern. Diese können entweder automatisch durch Entitäten oder mithilfe von benutzerdefinierten Aktionen gesetzt werden. Unter „intents” und „entities” werden alle Daten aufgelistet, die der NLU Interpreter extrahieren kann. Im Falle der Verwendung von Rasa NLU muss man diese zusätzlich in Rasa NLU trainieren [3].

[2] https://core.rasa.ai/domains.html
[3] http://nlu.rasa.ai/python.html

„Templates" stellen die generellen, textlichen Reaktionen auf Nachrichten dar. Hier können Antwortsätze gespeichert und einem „Template"-Namen zugeordnet werden („utter_<actionname>"). Die einfachen Antwort-Aktionen müssen dann jeweils unter „Actions" aufgelistet werden, zusammen mit „Custom Actions", die sich insofern von Antwortaktionen unterscheiden, als dass hier eigene Funktionen aufgerufen werden können, die weit mehr als nur einen textlichen Antwortsatz zurückgeben. Die „Custom Actions" werden in 6.3.4 genauer beschrieben.

Eine Domain-Datei für einen Chatbot, der einen freien Termin unter Angabe des Tages und der gewünschten Dauer sucht, könnte beispielsweise so aussehen:

```
slots:
  suggestedDate:
    type: text
  extractedDay:
    type: text
  extractedDuration:
    type: text

intents:
  - greet
  - affirm
  - deny
  - findappointment
  - inform_day
  - inform_duration
  - book_date

entities:
  - day
  - duration

templates:
  utter_default:
    - "Das habe ich nicht verstanden."
    - "Könntest du das bitte wiederholen?"
  utter_greet:
    - "Hallo, wie kann ich dir helfen?"
    - "Hey!"
  utter_ask_day:
    - "An welchem Tag soll der Termin stattfinden?"
    - "Für welchen Tag?"
```

```
  utter_ask_duration:
    - "Wie lange soll der Termin in Anspruch nehmen?"
    - "Wie viel Zeit soll eingeplant werden?"

actions:
  - utter_default
  - utter_greet
  - utter_ask_day
  - utter_ask_duration
  - actions.ActionSearchAvailableDate
  - actions.ActionSetDay
  - actions.ActionSetDuration
  - actions.ActionBookDate
```

### 6.3.2 NLU - Trainingsdaten

NLU Trainingsdaten liegen im Markdown-Format vor. Jeder Benutzerintention werden Beispielsätze vorgegeben. Auf Basis dessen lernt das NLU Tool die verschiedenen Intents zu klassifizieren. Außerdem werden Positionen von Entitäten vorgegeben, die variabel besetzt werden können. Dabei ist zu beachten, dass sowohl Wert als auch Name der Entität angegeben werden, ohne Beachtung der Groß- und Kleinschreibung. Die Entität "day" mit dem Wert "freitag" wird zum Beispiel wie folgt markiert: [freitag](day).

Ein Trainingsdatensatz könnte als Beispiel wie folgt aussehen:

```
## intent:affirm
- ja
- korrekt
- ok
- okay

## intent:deny
- falsch
- keine
- nein

## intent:inform_day
- bitte vereinbare einen termin am [dienstag](day)
- gibt es [freitag](day) einen freien termin
- ist [heute](day) ein termin frei
```

```
## intent:inform_duration
- [60 minuten](duration)
- [eine stunde](duration)
- [zwei stunden](duration)
```

Um geeignete Ergebnisse zu erhalten, sind mindestens 10 Beispiele pro Intent[4] nötig. Mehr Trainingsdaten verbessern das Resultat der Klassifikation.

### 6.3.3 Dialog - Trainingsdaten

Das Training des Dialoges findet mithilfe der Intents, Slots und Actions statt. Der Trainingsdatensatz dazu ist im Markdown-Format. Hier als Beispiel ein passender Dialog:

```
## Story 001
* inform_time[day=heute]
  - action_set_day
  - slot{"extractedDay": "2017-11-23T00:00:00.000Z"}
  - utter_ask_duration
* inform_duration[duration=60 minuten]
  - action_set_duration
  - slot{"extractedDuration": "60"}
  - action_search_availableDate
  - slot{"suggestedDate": "2017-11-23 14:00:00"}
* affirm
  - action_book_date
  - action_restart
```

Es wird zuerst über den Tag informiert, wodurch die Aktion „action_set_day" aufgerufen und der Slot „extractedDay" gefüllt wird. Folgend wird mit „utter_ask_duration" die gewünschte Dauer abgefragt, woraufhin der Benutzer mit der gewünschten Information antwortet.

Nun wird mit „action_set_duration" die erhaltene Dauer als Slot gespeichert und danach mit „action_search_availableDate" ein freier Termin gesucht und die gefundene, freie Zeitspanne als Slot gesetzt („suggestedDate"). Stimmt der Benutzer zu, wird der Termin gebucht und der Dialog startet neu („action_restart").

Dies ist nur ein Dialog-Zweig. Die weiteren Routen, beispielsweise, wenn der Benutzer dem Terminvorschlag nicht zustimmt, müssen ebenso trainiert werden, damit der Chatbot in die-

[4] http://nlu.rasa.ai/faq.html

sen Fällen korrekt reagieren kann. Hierfür können auch nur Teile des Dialogs trainiert werden, die nicht unbedingt im Zusammenhang mit vorherigen Aktionen stehen (beispielsweise wird nach „inform_time" immer die Aktion „action_set_day" ausgeführt, egal, in welchem Kontext sich die Nachricht befindet). Im Trainingsprozess werden diese kleineren Dialoge dynamisch miteinander kombiniert, um sehr viele verschiedene Pfade zu generieren.

Das Ergebnis des Trainings ist ein Baum, anhand dessen der Bot über die Aktionen und Reaktionen entscheidet. Dieser kann von Rasa Core automatisch visualisiert werden und sieht in diesem Beispiel wie folgt aus:

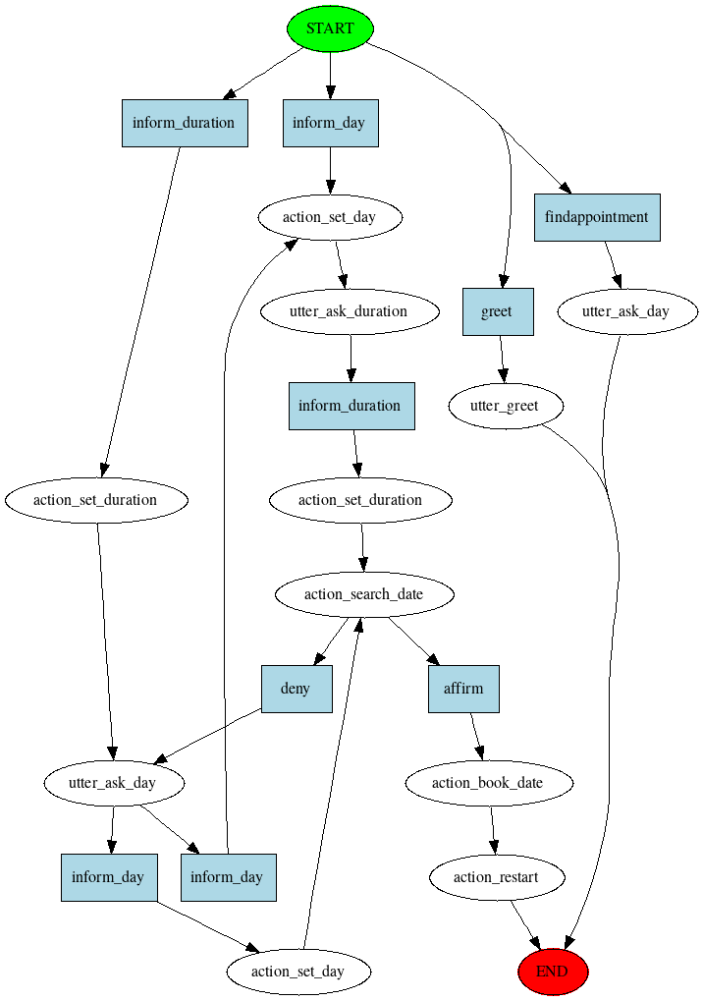



Abbildung 6.1: Graph des Chatbot-Dialogmodells

### 6.3.4 Custom Actions

In Custom Actions können benutzerdefinierte Python Funktionen ausgeführt werden. Man kann insbesondere auf den Message Tracker von Rasa Core und somit auf die letzte empfangene Nachricht, die letzten Entitäten und allen Slot-Werten zugreifen[5]. Eine Custom Action wird beispielsweise wie folgt implementiert:

```
class ActionSetDay(Action):
  def name(self):
    return 'action_set_day'

  @classmethod
  def run(self, dispatcher, tracker, domain):
    # extrahiert Zeitangaben aus der letzten Nachricht
    dates = extractTime(tracker.latest_message)
    # wenn Zeitangabe gefunden setze Slot
    if len(dates) == 1:
      return[SlotSet('extractedDayFrom', dates)]
```

Neben dem Setzen von Slots ist es auch möglich, Nachrichten an den Benutzer zurückzusenden („dispatcher.utter_message(string)") oder mithilfe des mysql.connector-Pakets auf Datenbanken zuzugreifen.

Im Falle unseres Bots werden so die Benutzer-IDs von den Chat-Teilnehmern, deren Kalendereinträge sowie Kompetenzen in der Aktion „action_search_availableDate" ausgelesen und bei einer erfolgten Terminbuchung (Aktion „action_book_date") die Termine in die entsprechenden Benutzerkalender eingetragen.

## 6.4 Humbot

Es wurde ein Chatbot umgesetzt, der die beiden Use Cases „Finden eines freien Termins" und „Suche nach Kompetenzen" bedient. Die Trainingsdaten für das Dialog-Training und NLU Fähigkeiten sowie Custom Actions sind zusammengefasst und werden nachfolgend einzeln, getrennt nach Use Case, betrachtet.

[5] https://core.rasa.ai/domains.html

### 6.4.1 Use Case: Finden eines freien Termins

Der umgesetzte Chatbot kann freie Termine für alle in der Unterhaltung beteiligten Personen finden. Als Eingabedaten werden der gewünschte Tag oder die gewünschte Zeitspanne, in welcher der Termin stattfinden soll und der Umfang des Termins in Minuten oder Stunden vorausgesetzt. Optional kann ein zu buchender Raum sowie weitere einzuladende Personen bestimmt werden. Die Daten werden zuerst gesammelt und anschließend zusammengefasst.

Eine große Herausforderung bei der Datensammlung waren Fälle, in denen mehrere Intentionen vorkamen. Beispielsweise gibt es die Intention „inform_time" für die Angabe des gewünschten Tages und „inform_duration" für die gewünschte Dauer des Termins. Fällt nun der Satz „Ich benötige morgen einen freien Termin für eine Stunde" werden beide Intentionen bedient, der Chatbot kann jedoch nur auf eine Intention reagieren.

Eine Lösung wäre ein weiteres Intent „inform_time_and_duration" einzuführen, was jedoch zu einer exponentiellen Steigerung der Anzahl von Intents und somit einen sehr hohen Trainingsaufwand - sowohl für NLU, als auch für das Dialog-Management - bedeutet und ebenso die Genauigkeit der Klassifikation der Intention verringert. Die bessere Lösung war, alle relevanten Eingabesätze mit Informationsgehalt für den Kalendertermin nur einer einzigen Intention - „inform_calendar" - zuzuordnen und mithilfe einer Custom Action die beinhalteten Daten herauszufiltern. Schließlich wird mithilfe von Boolean-Slots (Variablen, die ausschließlich „wahr" oder „falsch" speichern) bestimmt, welche benötigten Daten bereits vorhanden sind oder noch fehlen. Auf Basis der Slot-Zustände kann Rasa Core die nächste Aktion zielgerecht bestimmen.

Am Ende der Sammlung der Informationen werden die Daten zusammengefasst und dargestellt. Stimmt der Nutzer nun zu, wird ein Termin gesucht, der die vorab gesammelten Kriterien erfüllt. Lehnt der Nutzer ab hat dieser die Möglichkeit, einzelne Attribute zu ändern. Hierfür wird wieder aus dem Eingabesatz analysiert, welche Information der Nutzer ändern möchte. Nach der Änderung werden die Daten wieder zusammengefasst.

Bei der Terminsuche wird der gewünschte Tag intern als zweidimensionales Array dargestellt. Hierbei ist der erste Index die Stunde und der zweite Index jeweils ein Abschnitt von 15 Minuten, sodass der zweite Index die Werte 0, 1, 2 und 3 annehmen kann. Als Beispiel ist die Uhrzeit 12:00 der Index [12][0] und 15:45 der Index [15][3]. Alle Elemente des Arrays haben standardmäßig den Wert 0. Je nach den Daten der bereits belegten Termine jedes beteiligten Nutzers werden die Werte auf 1 gesetzt, sodass dieser Termin nicht mehr verfügbar ist. Mithilfe der gewünschten Termindauer wird nun ein freier Termin vorgeschlagen. Der

Benutzer kann nun entweder zustimmen, womit der Termin gebucht und in der Datenbank unter jedem Benutzer gespeichert wird, oder ablehnen, womit der nächste passende freie Termin gesucht wird.

Der fertig implementierte Chatbot wurde anschließend mit der Rasahub Middleware gestartet, einige Kalendereinträge in Humhub bei zwei Benutzerprofilen vorgenommen und eine Beispielkonversation geführt.

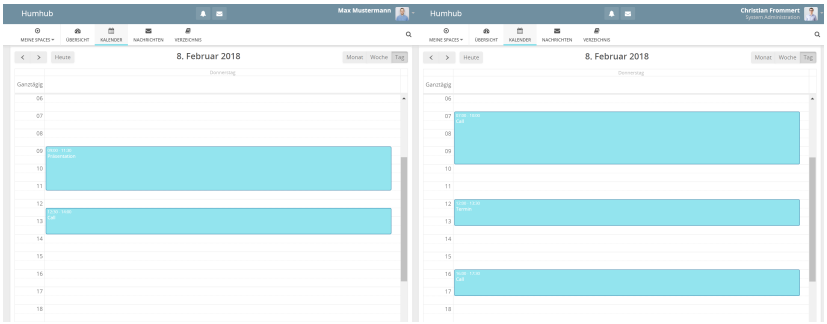



Abbildung 6.2: Kalender davor, links: Person 1, Rechts: Person 2

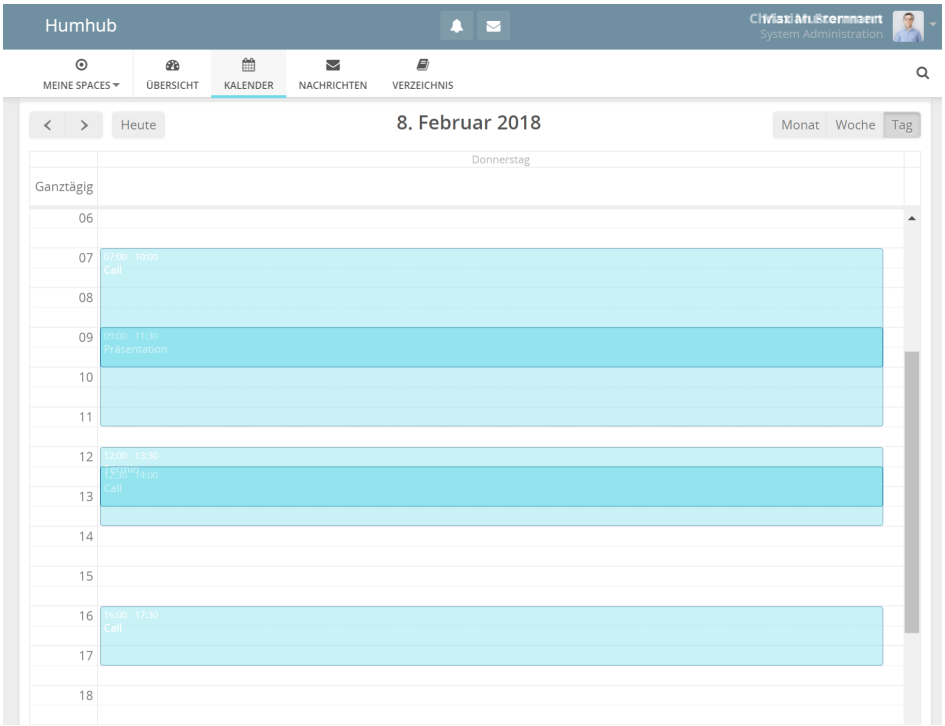



Abbildung 6.3: Kalender kombiniert

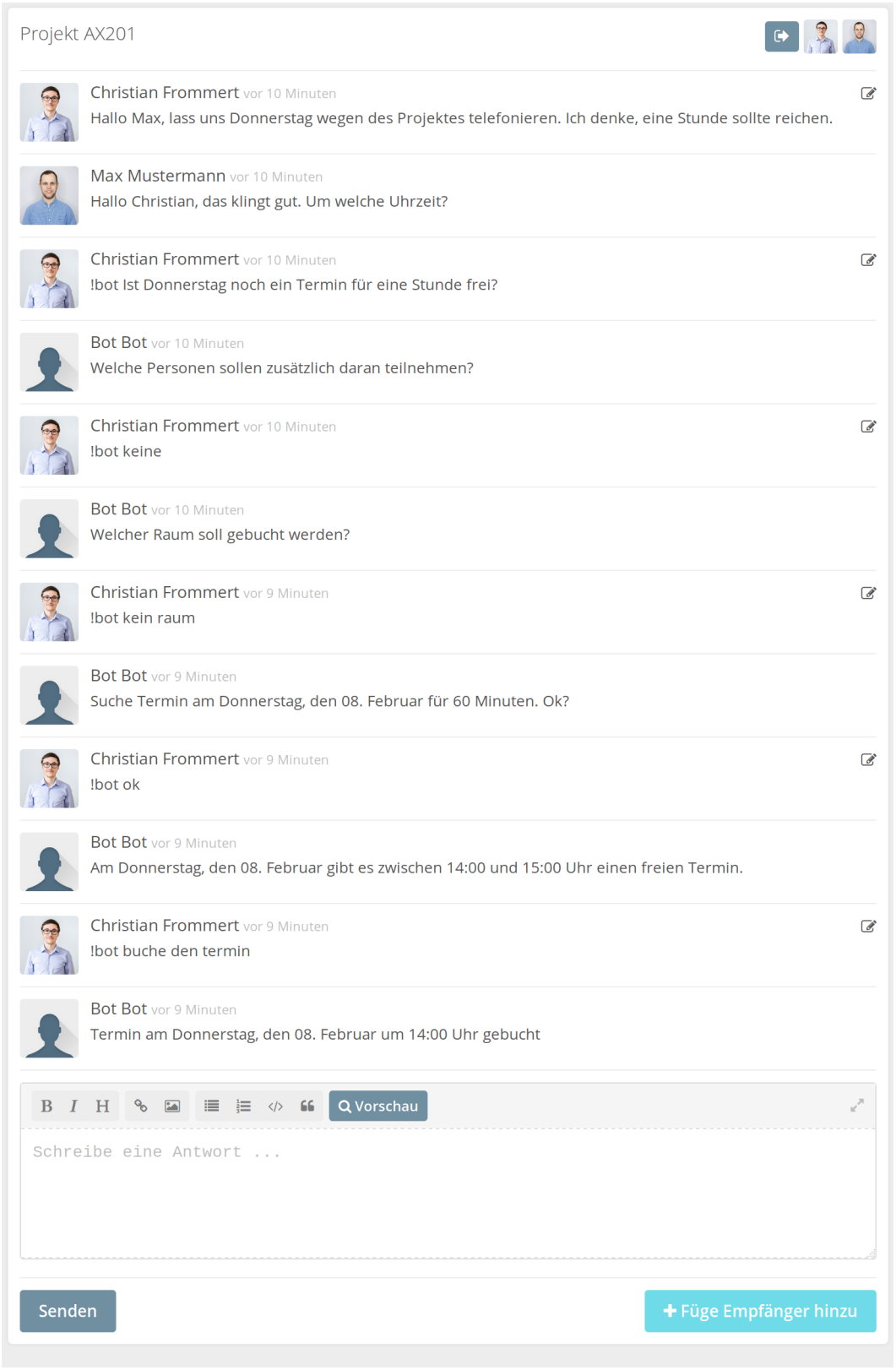



Abbildung 6.4: Chat-Unterhaltung

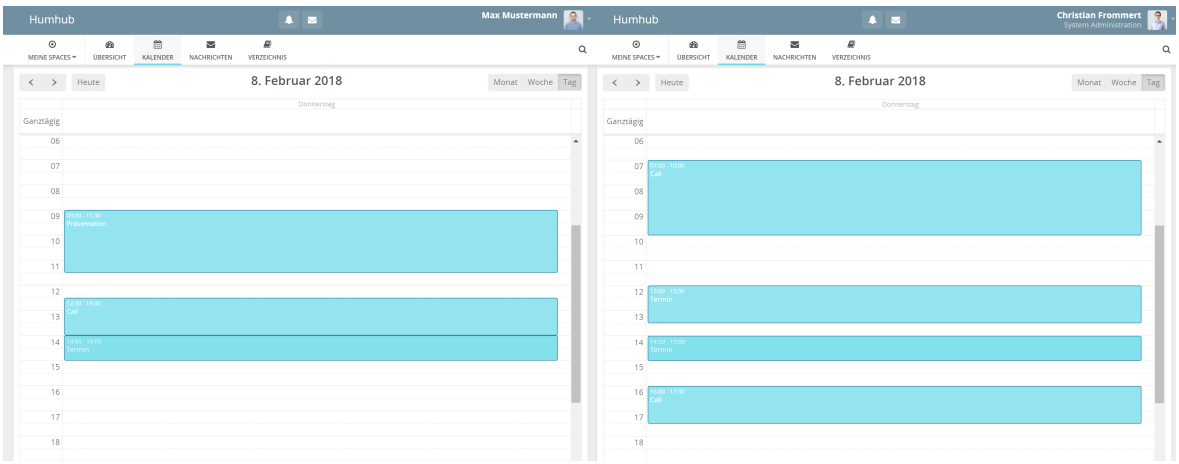



Abbildung 6.5: Kalender danach, links: Person 1, Rechts: Person 2

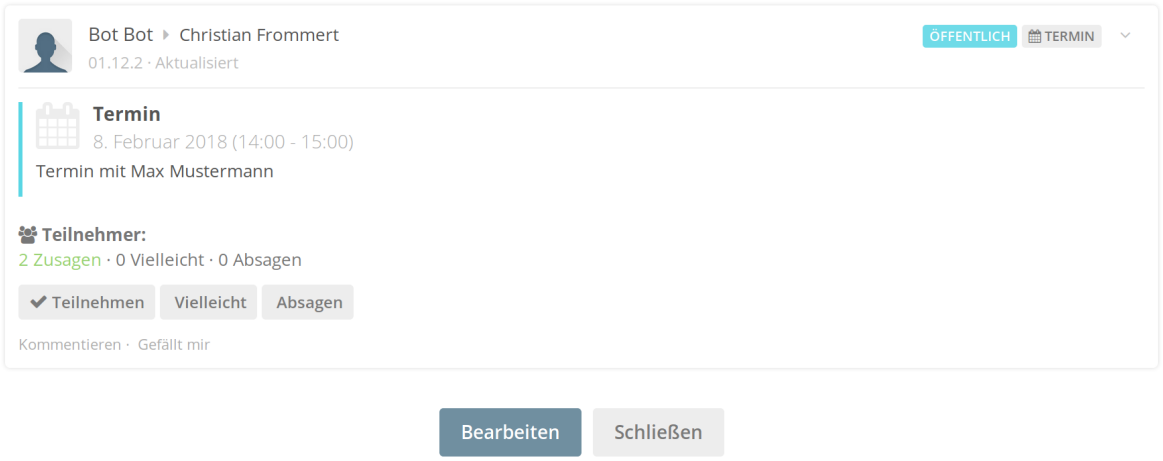



Abbildung 6.6: Kalender Eintrag

Optional könnte noch implementieren werden, dass der gewünschte Raum ebenfalls gebucht wird. Da Humhub diese Funktionalität jedoch nicht zur Verfügung stellt, wurde dies nicht berücksichtigt. Alternativ könnte der Raum als Humhub-Benutzer mit eigenem Kalender angelegt und als „weitere Person” behandelt werden.

### 6.4.2 Use Case: Suche nach Kompetenzen

Im Beispiel der Kompetenz-Suche bestand das Problem nicht im Sammeln der benötigten Informationen und der Auswahl der passenden Aktion, sondern vornehmlich in der Daten-Repräsentation.

Um nach Benutzern mit bestimmten Kompetenzen suchen zu können, benötigt die Software einen Nutzerbestand mit zugeordneten Kompetenzen. Dieser kann sich entweder in einen ausgelagerten Dienst, wie dem unternehmenseigenen AD, oder - wie in dieser Bachelorarbeit verwendet - direkt in der Humhub Datenbank befinden. Die Zuordnung von Benutzern zu Kompetenzen lässt sich wie folgt darstellen:

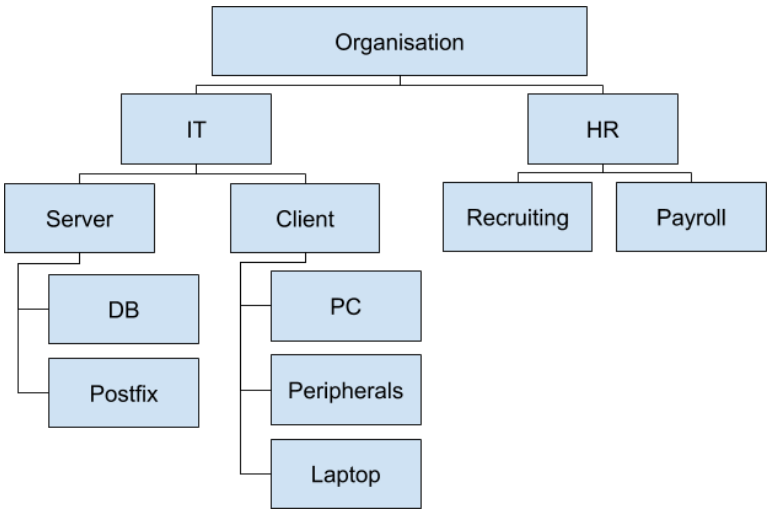
$$
\begin{aligned}
BenutzerA &\Rightarrow Kompetenz1 \\
BenutzerB &\Rightarrow Kompetenz1,\ Kompetenz2 \\
BenutzerC &\Rightarrow Kompetenz2 \\
BenutzerD &\Rightarrow Kompetenz3
\end{aligned}
$$

Bei Anfragen wird jedoch von einer gesuchten Kompetenz ausgegangen, sodass der Datenbestand transponiert und zusammengefasst werden muss:

$$
\begin{aligned}
Kompetenz1 &\Rightarrow BenutzerA,\ BenutzerB \\
Kompetenz2 &\Rightarrow BenutzerB,\ BenutzerC \\
Kompetenz3 &\Rightarrow BenutzerD
\end{aligned}
$$

Schließlich wird noch eine Repräsentation aller Kompetenzen und der Hierarchie benötigt. Wenn beispielsweise nach einer Person mit der speziellen Kompetenz „Notebooks” gesucht wird, niemand diese Kompetenz aufweist oder aktuell keine Person mit dieser Kompetenz online ist, aber eine Person mit der hierarchisch höher gelegenen bzw. generalisierten Kompetenz „IT” vorhanden oder online ist, sollte diese Person vorgeschlagen werden.

Folgend wird als Beispiel ein Ausschnitt eines Organigramms eines Unternehmens sowie die Darstellung des Organigramms mit Synonymen zu jeder Stelle als JavaScript Object Notation (JSON) Datenmodell gezeigt.




*Quelle:* Eigene Darstellung

Abbildung 6.7: Ausschnitt Organigramm

Ausschnitt des hierarchischen Datenmodells als JSON:

```
[
  {
    "competence": "it",
    "synonyms": ["technik"],
    "subcategories": [
      {
        "competence": "server",
        "subcategories": [
          {
            "competence": "db",
            "synonyms": ["datenbank"]
          },
          {
            "competence": "Postfix",
            "synonyms": ["email", "e-mail", "mail", "mailserver"]
          }
        ]
      },
      {
        "competence": "client",
        "subcategories": [
          {
            "competence": "pc",
```

```
            "synonyms": ["computer"]
          },
          {
            "competence": "laptop",
            "synonyms": ["notebook"]
          },
          {
            "competence": "peripherals",
            "synonyms": ["tastatur", "maus", "drucker", "monitor"]
          }
        ]
      }
    ]
  },
  {
    "competence": "hr",
    "synonyms": ["personal", "personalwesen"],
    "subcategories": [
      {
        "competence": "payroll",
        "synonyms": ["abrechnung", "verdienst", "lohnabrechnung"]
      }
    ]
  }
]
```

Weiterhin musste beachtet werden, dass nicht alle Wortformen der Kompetenzen abgebildet werden können. Beispielsweise hat eine Person aus der Abteilung Buchhaltung die Kompetenz „Lohnabrechnung". Bei der Anfrage „Ich benötige Hilfe zu meiner Lohnabrechnung." würde die richtige Person zugeordnet werden können. Wäre die Frage jedoch „Welche Person bearbeitet die Lohnabrechnungen?", also unter der Benutzung der Pluralform, könnte die Kompetenz nicht zugeordnet werden.

Dieses Problem wurde mit einen „Snowball Stemmer" des Python Natural Language Toolkits (NLTK) behoben[6]. Sowohl die gesuchte Kompetenz der Anfrage, als auch alle Kompetenzen des Datenstamms, werden vor dem Vergleich zu ihrer Grundform reduziert, in dem genannten Beispiel zu „Lohnabrechnung". Auf diese Weise können neben Pluralformen auch gebäugte Formen verarbeitet werden.

[6] http://www.nltk.org/howto/stem.html

Folgend der Code für die rekursive Suche nach einer Kompetenz in der Hierarchie:

```
from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer("german")

def searchCompetence(search, dictionary):
  """
  Returns path for a competence from general to specialized
  """
  for competence in dictionary:
    if (
      (
        'competence' in competence and
        stemmer.stem(competence['competence'])
        == stemmer.stem(search.lower())
      )
      or
      (
        'synonyms' in competence and
        stemmer.stem(search.lower()) in [
          stemmer.stem(syn) for syn in competence['synonyms']
        ]
      )
    ):
      return [competence['competence']]
    else:
      try:
        if 'subcategories' in competence:
          return (
            searchCompetence(search, competence['subcategories']) +
            [competence['competence']])
      except ValueError:
        pass
  raise ValueError("Not found")
```

Der Aufruf von searchCompetence("computer", companyData) nach dem Einlesen des Organigramms gibt beispielsweise das folgende geordnete Array zurück:

```
[u'pc', u'client', u'it']
```

Auf diese Weise lassen sich nicht nur Mitarbeiter mit der speziell geforderten Kompetenz finden, sondern auch Angestellte mit generalisierten Kompetenzen, sofern nötig.

# 7 Zusammenfassung & Ausblick

Dieses Kapitel zeigt eine Zusammenfassung der Erkenntnisse dieser Bachelorarbeit und einen Ausblick der Entwicklung von Chatbots allgemein sowie im unternehmensinternen Kontext.

## 7.1 Zusammenfassung

Durch den unternehmensinternen Einsatz von Chatbots können zeitintensive Prozesse von Unternehmen automatisiert unterstützt werden. Bisherige Ansätze der Verwendung von Chatbots für Unternehmen gingen zumeist auf die Assistenz des Kunden ein, beispielsweise mit automatisierten Kundendienstprozessen, doch diese Bachelorarbeit zeigt auf, dass auch rein interne Abläufe durchaus unterstützt werden können.

Wie in der Einleitung bereits erläutert können Nutzerassistenzsysteme leichte Aufgaben von Angestellten schneller, mit einem qualitativ besseren Ergebnis und angenehmer für den Nutzer ausführen (Morana u. a. 2017). In den beiden umgesetzten Use-Cases dieser Bachelorarbeit wurde beispielsweise bei der Suche nach einem freien Termin unter mehreren Angestellten das „Ping-Pong-Spiel" vermieden. Außerdem konnten Mitarbeiterkompetenzen besser dargestellt und nach den tatsächlichen Anforderungen durchsuchbar gemacht werden.

## 7.2 Ausblick

Die großen Herausforderungen von Chatbots sind aktuell die vordefinierten Antwortsätze, die starre Kommunikation mit festen Dialogen und die begrenzte Empathiefähigkeit aufgrund fehlender Sensoren für Gefühle. Dies könnte in den nächsten Jahren durch leistungsfähigere Texterkennung, effizientere Verwendung von Machine Learning Funktionen, kontextsensitivem Dialogmanagement sowie der Integration von Sensoren und neuen Technologien,

wie beispielsweise Emotionserkennung via Webcam oder der Verarbeitung von Augenbewegungen in Echtzeit, verbessert werden (ebd.).

Weiterhin wäre bereits jetzt denkbar, Chatbots so zu erweitern, dass der Bedarf nach Assistenz vor der Artikulation des Nutzers festgestellt werden kann, basierend auf der Erfassung des aktuellen Kontextes und des Nutzerzustandes. So könnte der Chatbot im Kalenderbeispiel bereits während der Unterhaltung zwischen den beteiligten Kollegen die nötigen Daten für einen Termin sammeln und den Nutzer schließlich nur den vorgeschlagenen Termin bestätigen lassen. Dies hätte jedoch noch größere Auswirkungen auf die datenschutzrechtlichen Sicherheitsaspekte.

Darüber hinaus entwickeln sich Messaging-Dienste kontunierlich weiter. Aktuell lassen sich bei Facebook neben Text auch Buttons darstellen, sodass dies die Aktions-Auswahl des Chatbots vereinfacht. In Zukunft ist denkbar, dass im Unternehmen keine traditionellen Chatbots Einsatz finden, sondern starre Prozesse bereits ohne Nutzerinteraktion bzw. Rückfragen durch Assistenzsysteme erledigt werden können. Oftmals findet man zu Chatbot-Themen dazu den Begriff „Virtual Assistant", welcher im Hintergrund arbeitet und nicht aktiv mit dem Nutzer kommuniziert (Zamora 2017).

# Literatur


Back, Andrea (2016). "Enterprise 2.0 - Digitale Transformation durch soziale Technologien". In: *Business Innovation: Das St. Galler Modell*. Hrsg. von Christian Pieter Hoffmann, Silke Lennerts, Christian Schmitz, Wolfgang Stölzle und Falk Uebernickel. Bd. 1. Aufl. Business Innovation Universität St. Gallen, Profilbereich Business Innovation. Wiesbaden: Springer Fachmedien Wiesbaden, S. 123–138. URL: https://www.alexandria.unisg.ch/246905/.

Beißwenger, Michael und Angelika Storrer, Hrsg. (2005). *Chat-Kommunikation in Beruf, Bildung und Medien : Konzepte, Werkzeuge, Anwendungsfelder*. Stuttgart: ibidem-Verl. URL: http://swbplus.bsz-bw.de/bsz111089468inh.htm.

Bocklisch, T., J. Faulkner, N. Pawlowski und A. Nichol (2017). "Rasa: Open Source Language Understanding and Dialogue Management". In: *ArXiv e-prints*. arXiv: 1712.05181 [cs.CL].

Boys, Danah und Kate Crawford (2012). "Critical Questions for Big Data". In: *Information, Communication & Society* 15.5, S. 662–679. DOI: 10.1080/1369118X.2012.678878. eprint: https://doi.org/10.1080/1369118X.2012.678878. URL: https://doi.org/10.1080/1369118X.2012.678878.

Brants, Thorsten (2004). "Natural Language Processing in Information Retrieval". In: *In Proceedings of the 14th Meeting of Computational Linguistics in the Netherlands*, S. 1–13.

Braun, Daniel, Adrian Hernandez-Mendez, Prof. Dr. Florian Matthes und Dr. Manfred Langen (2017). *Evaluating Natural Language Understanding Services for Conversational Question Answering Systems*. SIGdial. URL: http://www.sigdial.org/workshops/conference18/proceedings/pdf/SIGDIAL22.pdf.

Cahn, Jack (2017). “CHATBOT: Architecture, design, and development”. In: *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science*.

Chowdhury, Gobinda G. (2003). “Natural language processing”. In: *Annual Review of Information Science and Technology* 37.1, S. 51–89. DOI: 10.1002/aris.1440370103. URL: http://dx.doi.org/10.1002/aris.1440370103.

Day, Jaycee (2017). *How we got a 98% success rate for our bot for Zalanda — A UX case study*. https://chatbotsmagazine.com/how-we-got-a-98-success-rate-for-our-bot-for-zalando-a-ux-case-study-fcdc0e70469d. Blog.

De Capitani di Vimercati, Sabrina, Stefano Paraboschi und Pierangela Samarati (2003). “Access control: principles and solutions”. In: *Software: Practice and Experience* 33.5, S. 397–421. DOI: 10.1002/spe.513. URL: http://dx.doi.org/10.1002/spe.513.

Destatis (2017). *Nutzung von Informations- und Kommunikationstechnologien in Unternehmen*. Statistisches Bundesamt. URL: https://www.destatis.de/DE/Publikationen/Thematisch/UnternehmenHandwerk/Unternehmen/InformationstechnologieUnternehmen.html.

Eckmüller, Birgit (2016). *Digitale Überforderung im Arbeitsalltag*. Soprasteria. URL: https://www.soprasteria.de/docs/librariesprovider33/Studien/digitale-ueberforderung-im-arbeitsalltag.pdf?sfvrsn=2.

Gregori, Eric (2017). *Evaluation of Modern Tools for an OMSCS Advisor Chatbot*. Georgia Institute of Technology. URL: https://smartech.gatech.edu/handle/1853/58516.

Hevner, Alan R., Salvatore T. March, Jinsoo Park und Sudha Ram (2004). “Design Science in Information Systems Research”. In: *MIS Q.* 28.1, S. 75–105. URL: http://dl.acm.org/citation.cfm?id=2017212.2017217.

Jackson, Thomas, Ray Dawson und Darren Wilson (2002). "Case study: evaluating the effect of email interruptions within the workplace". In: *Conference on Empirical Assessment in Software Engineering*. URL: https://dspace.lboro.ac.uk/2134/489.

Joulin, A., E. Grave, P. Bojanowski und T. Mikolov (2016). "Bag of Tricks for Efficient Text Classification". In: *ArXiv e-prints*. arXiv: 1607.01759 [cs.CL].

Kemp, Simon (2018). *Digital in 2018*. we are social. URL: https://digitalreport.wearesocial.com/download.

Koch, M. und A. Richter (2009). *Enterprise 2.0: Planung, Einführung und erfolgreicher Einsatz von Social Software in Unternehmen*. Oldenbourg. URL: https://books.google.de/books?id=BNyKs3chIg0C.

Konrat Canan & Weick, Günter (2008). *E-Mail Nutzung in deutschen Unternehmen*. SoftTrust. URL: http://www.softrust.com/docs/SofTrust%20E-Mail-Studie%20V2.pdf.

Kuhn, R. und R. De Mori (1995). "The application of semantic classification trees to natural language understanding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.5, S. 449–460. DOI: 10.1109/34.391397.

Marietto, M. d. G. B., R. Varago de Aguiar, G. de Oliveira Barbosa, W. Tanaka Botelho, E. Pimentel, R. dos Santos França und V. Lúcia da Silva (2013). "Artificial Intelligence MArkup Language: A Brief Tutorial". In: *ArXiv e-prints*. arXiv: 1307.3091 [cs.AI].

Mohit, Behrang (2014). "Named Entity Recognition". In: *Natural Language Processing of Semitic Languages*. Hrsg. von Imed Zitouni. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 221–245. DOI: 10.1007/978-3-642-45358-8_7. URL: https://doi.org/10.1007/978-3-642-45358-8_7.

Morana, Stefan, Celina Friemel, Ulrich Gnewuch, Alexander Maedche und Jella Pfeiffer (2017). "Interaktion mit smarten Systemen – Aktueller Stand und zukünftige Entwicklungen im Bereich der Nutzerassistenz". In: *Wirtschaftsinformatik & Management* 9.5, S. 42–51. DOI: 10.1007/s35764-017-0101-7. URL: https://doi.org/10.1007/s35764-017-0101-7.

Peffers, Ken, Tuure Tuunanen, Marcus A. Rothenberger und Samir Chatterjee (2007). "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3, S. 45–77. URL: http://www.jstor.org/stable/40398896.

Pfaff, T. (1995). *Dokumentenmanagement - das papierlose Büro?: Konzepte, Technologien, Tips*. EDV-Praxis. VDE-Verlag. URL: https://books.google.de/books?id=PlWbAAAACAAJ.

Reiter, Ehud und Robert Dale (2000). *Building Natural Language Generation Systems*. New York, NY, USA: Cambridge University Press.

Rennebarth, Enrico (2015). *Kundenservice am Wochenende*. Verint. URL: https://callcenter-verband.de/wp-content/uploads/2015/04/Whitepaper-Kundenservice-am-Wochenende-BtoC_CCV_Verint.pdf.

Sawall, Achim (2011). "Atos: IT-Services-Unternehmen verbietet interne E-Mails". In: *Golem*. URL: https://www.golem.de/1111/88078.html (besucht am 30. 12. 2017).

Schank, Roger C. (1972). "Conceptual dependency: A theory of natural language understanding". In: *Cognitive Psychology* 3.4, S. 552–631. DOI: https://doi.org/10.1016/0010-0285(72)90022-9. URL: http://www.sciencedirect.com/science/article/pii/0010028572900229.

Shawar, Bayan Abu und Eric Atwell (2004). "Accessing an Information System by Chatting". In: *Natural Language Processing and Information Systems*. Hrsg. von Farid Meziane und Elisabeth Métais. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 407–412.

— (2007a). "Chatbots: Are they Really Useful?" In: *LDV Forum* 22.1, S. 29–49. URL: http://www.jlcl.org/2007_Heft1/Bayan_Abu-Shawar_and_Eric_Atwell.pdf.

— (2007b). "Different Measurements Metrics to Evaluate a Chatbot System". In: *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in*

*Dialog Technologies*. NAACL-HLT-Dialog '07. Rochester, New York: Association for Computational Linguistics, S. 89–96. URL: http://dl.acm.org/citation.cfm?id=1556328.1556341.

Steingen, Prof. Dr. Ulrich und Nicolay Mausz (2014). *Enterprise Social Network*. WEKA Verlag. URL: http://www.flyingdog.de/downloads/Enterprise_Social_Network.pdf.

Stelzner, Michael A. (2017). *2017 Social Media Marketing Industry Report*. Social Media Examiner. URL: http://www.socialmediaexaminer.com/wp-content/uploads/2017/05/Industry-Report-2017.pdf.

Tromp, Erik und Mykola Pechenizkiy (2011). "Graph-based n-gram language identification on short texts". In: *Proc. 20th Machine Learning conference of Belgium and The Netherlands*, S. 27–34.

Turban, Efraim, Narasimha Bolloju und Ting-Peng Liang (2011). "Enterprise Social Networking: Opportunities, Adoption, and Risk Mitigation". In: *Journal of Organizational Computing and Electronic Commerce* 21.3, S. 202–220. DOI: 10.1080/10919392.2011.590109. URL: https://doi.org/10.1080/10919392.2011.590109.

Zamora, Jennifer (2017). "Rise of the Chatbots: Finding A Place for Artificial Intelligence in India and US". In: *Proceedings of the 22Nd International Conference on Intelligent User Interfaces Companion*. IUI '17 Companion. Limassol, Cyprus: ACM, S. 109–112. DOI: 10.1145/3030024.3040201. URL: http://doi.acm.org/10.1145/3030024.3040201.

Zhang, Jiawei, Philip S. Yu und Yuanhua Lv (2015). "Organizational Chart Inference". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. Sydney, NSW, Australia: ACM, S. 1435–1444. DOI: 10.1145/2783258.2783266. URL: http://doi.acm.org/10.1145/2783258.2783266.

## Eidesstattliche Erklärung

Ich versichere, dass ich meine Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Diese Arbeit wurde von mir bisher keiner Prüfungsbehörde in gleicher oder ähnlicher Form oder auszugsweise vorgelegt.

Darüber hinaus versichere ich, dass die elektronische Version der Bachelorarbeit mit der gedruckten Version übereinstimmt.

Leipzig, den ..... .....

Unterschrift