Recommender Systems für Webbasierte Argumentationssysteme

Bachelorarbeit

von

Stefan Genenger

aus

Viersen

vorgelegt am

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

April 2017

Betreuer:
Tobias Krauthoff M. Sc.

Zusammenfassung

Im Lehrstuhl für Rechnernetze wird eine dialogbasierte Argumentationssoftware entwickelt. Diese soll die Beitragsstruktur, die durch Internetforen bekannt ist durch eine Dialogstruktur ersetzen. An bestimmten Stellen des Programms wird der Nutzer allerdings nach zu vielen Dialogeinträgen anderen Nutzer mit zu vielen Auswahlmöglichkeiten konfrontiert. Hierfür wurde als Proof of Concept ein Recommender System in Form eines Microservices auf Basis des Collaborative Filterings entwickelt, welches unter den Auswahlmöglichkeiten, die zur Verfügung stehen jeweils fünf Möglichkeiten vorschlägt, die den aktuellen Nutzer am ehesten interessieren könnten. Dabei arbeitet der Filteralgorithmus mit dem Vergleich des früheren Verhaltens des aktuellen Nutzers mit dem früheren Verhalten aller anderen Nutzer. Ein Aussortierungsverfahren schränkt die Auswahl ein, bis fünf Möglichkeiten übrig bleiben und vorgeschlagen werden. Die Evaluation, die die verschiedenen Facetten des Vergleiches der Nutzer untereinander getestet hat, hat gezeigt, dass das Konzept eines Recommender Systems für eine Argumentationssoftware aufgeht. Welche Art Recommender System in diesem Fall optimal wäre, bleibt Gegenstand weiterer Forschung.

Danksagung

Ich danke meinem Betreuer Tobias Krauthoff für die angenehme und lockere Zusammenarbeit und auch Christian Meter, der mir bei einigen technischen Problemen zur Seite stand.

Ganz besonders bedanke ich mich bei meinen Eltern, die mich über meine sehr lange Studienzeit hinweg mit viel Geduld und Hilfsbereitschaft unterstützt haben.

Außerdem bedanke ich mich bei allen Korrekoren dieser Arbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Die webbasierte Argumentations-Software "Dialog-Based Argumentation System- [KBBM16] (kurz D-BAS), die am Lehrstuhl für Rechnernetze an der Heinrich Heine Universität entwickelt wird, dient zur Vereinfachung der Kommunikation zwischen Gesprächsteilnehmern im Internet. Dabei wird die bisher übliche Beitragsstruktur, die aus Online-Foren bekannt ist, durch eine geführte Dialogsstruktur ersetzt. Nach Auswahl eines Themas kann der Nutzer seine Meinung zu bisher eingetragenen Positionen erläutern. Das System fragt den Nutzer dabei nach Prämissen für seine Aussagen und versucht diesen daraufhin mit anderen Argumenten anzugreifen. Darauf reagiert der Nutzer entsprechend wieder mit seiner Meinung und einer Prämisse, bis das System keine weiteren Angriffe mehr kennt. Es steht dem Nutzer außerdem immer frei, nicht nur bereits eingetragene Positionen oder Argumente auszuwählen, sondern auch seine eigenen zu formulieren. All diese Argumente werden wiederum anderen Nutzern an passender Stelle zur Auswahl angezeigt. Im ersten Schritt soll jedoch basierend auf der Millerschen Zahl eine Vorauswahl getroffen werden. Nach Miller [Mil56] ist der Mensch nicht in der Lage mehr als 7 ± 2 Objekte im Kurzzeitgedädchtnis zu behalten. Da die Antwortmöglichkeiten in D-BAS textbasiert und damit komplexer als in Millers Experimenten sind, sollen maximal nur fünf Antwortmöglichkeiten in D-BAS zur Wahl stehen. Ohne Recommender System würde D-BAS alle vorhandenen Einträge anzeigen.

1.1 Begriffsklärung

D-BAS unterschiedet in der aktuellen Version nicht zwischen Interesse und Meinung. Ein Nutzer kann aus Interesse eine Aussage tätigen und begründen, ohne dass dies seiner Meinung entspricht, um zu sehen, womit das System ihn konfrontiert. In dieser Arbeit wird der Begriff Meinung verwendet, um das aktuelle Verhalten des Nutzers wiederzuspiegeln. Es wird davon ausgegangen, dass der Nutzer nichts anderes als seine Meinung angibt.

1.2 Ziel der Arbeit

Im Rahmen dieser Bachelorarbeit soll als Proof of Concept überprüft werden, ob es möglich ist, die Anzeige aller Einträge durch eine durch ein Recommender System unterstützte Anzeige vorgeschlageneder Einträge zu ersetzen. Es sollen möglichst diejenigen Antwortmöglichkeiten präsentiert werden, die dem aktuellen Nutzer[1] am ehesten zusagen. Dabei sind zu Anfang der Arbeit vier Bereiche in D-BAS definiert worden, in denen eine Auswahl getroffen werden soll.

1. Auswahl von fünf Positionen zu einem Thema
2. Auswahl von fünf Prämissen für die Nutzermeinung zu einer Aussage oder Position
3. Auswahl von fünf Prämissen für den Angriff auf ein vom System vorgestelltes Argument
4. Auswahl eines Angriffs auf das Argument eines Nutzers

Während der Arbeiten an der Bachelorarbeit hat zeitgleich ein Experiment am Lehrstuhl mit einer Beispieldiskussion stattgefunden. Bei der Auswertung wurde klar, dass das Zufallsprinzip bei Punkt 4 gut funktioniert, und es wurde beschlossen, diesen Punkt aus der Zielsetzung der Bachelorarbeit zu entfernen. Damit bleiben die ersten drei Aufgaben. Das Recommender System soll außerdem getrennt von D-BAS als Microservice implementiert werden.

1.3 Gliederung

Nach diesem einleitenden Kapitel, dient Kapteil 2 zur Orientierung des Lesers. Es werden die Themen *Recommender Systems* und *Microservices* im Allgemeinen erläutert und somit eine theoretische Grundlage für die Arbeit gelegt. Danach wird in Kapitel 3 auf die Implementierung eingegangen, welche dann durch Kapitel 4 evaluiert wird. In Kapitel 5 wird die Arbeit noch einmal kurz zusammengefasst und es wird ein Ausblick auf zukünftige Möglichkeiten gegeben.

[1] Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet. Sämtlichen Personenbezeichnungen gelten gleichermaßen für beiderlei Geschlechter

Kapitel 2

Grundlagen

Dieses Kapitel soll eine Einführung in theoretischen Grundlagen der Bachelorarbeit geben. Hauptsächlich geht es um die Funktionsweise, Entiwicklung und mögliche Filteralgorithmen von Recommender Systems. Zusätzlich werden Microservices und ihre Vor- und Nachteile besprochen. Abschließend wird erläutert in welchem Kontext beide Themen zur Anwendung kommen.

2.1 Recommender Systems

2.1.1 Definition

Ein Recommender System ist ein Softwareprogramm, dessen Ziel es ist einer Auswahl an Nutzern Vorschläge von Objekten zu machen, die für sie von Interesse sein könnten. Als mögliche Objekte können zum Beispiel Produkte wie Bücher beim Online-Shop Amazon oder Filme und Serien auf dem Streaming-Portal Netflix genannt werden. Aufgrund welcher Mechanismen das Recommender System arbeitet, hängt vor allem von der Art der Objekte und den Daten ab, die zu diesen Objekten zu Verfügung stehen. [SW10]

2.1.2 Entwicklung

Recommender Systems und ihre Möglichkeiten haben sich parallel zum Internet entwickelt. Von statischen Online-Shops über Social Media bis hin zu RFID-Technologien wurde es möglich, immer mehr Informationen und Daten über die Nutzer zu beziehen und zu bearbeiten. Im Folgenden wird die

Entwicklung des Internets erläutert und ein Bogen zu den Möglichkeiten für Recommender Systems geschlagen [BOHG13, vlg.] [EWe10, vgl.].

Web 1.0 - Web der Dokumente
Als das Internet in den frühen 90er Jahren zur weltweiten Nutzung freigegeben wurde, begann es als Web 1.0 mit einer statischen Sammlung an Webseiten, die durch Hyperlinks miteinander verbunden werden konnten. Für Recommender Systems ergab sich hier ein erster Ansatz. Vor allem im Bereich des Onlineshoppings sind die Möglichkeiten offensichtlich. Nutzerkonten mit persönlichen Daten konnten erstellt und daraufhin weitere Daten zum Verhalten dieser Konten gesammelt werden.

Web 2.0 - Web der Services
Mit der Möglichkeit für den Nutzer eigene Inhalte auf Webseiten zu veröffentlichen, entwickelte sich durch das Web 2.0 eine neue Welt. Seiten wie Facebook und Twitter sind heute für die meisten Menschen alltäglich. Nutzer präsentieren und konsumieren Inhalte im gegenseitigen Austausch. Diese Inhalte umfassen auch den Nutzer selbst und hier finden Recommender Systems neue Möglichkeiten, Daten zu sammeln und zu verarbeiten: Friends, Follower, Posts, Tags... All diese Dinge und der Umgang mit ihnen können gespeichert und ausgewertet werden.

Web 3.0 - Sozial Semantisches Web
Das Web 3.0 oder auch Internet der Dinge beschreibt ein Internet, das Maschinen lesen können. Die Inhalte von Nutzern sollen automatisch verarbeitet und im richtigen Kontext verstanden werden können. In der Vision diesesn Internets werden Recommender Systems alltäglich. Es werden nicht nur Objekte vorgeschlagen, sondern auch Verhalten. Ein bereits existierendes Beispiel ist ein Navigationssystem. Es schlägt aufgrund seiner Informationen Routen und ein Verhalten bei Geschwindigkeitsüberschreitung vor. Die Vision des Webs 3.0 geht aber noch weiter. So könnte aufgrund von bekannten Gesundheitsparametern ein System vorschlagen, einen Arzt aufzusuchen oder das eigene Haus registriert, dass der Nutzer von der Arbeit nach Hause fährt, regelt darauf die Atmosphäre im Haus (Temperatur, Licht, etc.), wie es dem Nutzer gefallen könnte [Web14]. Eine Automatisierung dieser Dinge wäre unter anderen durch Recommender Systems möglich.

2.1.3 Filteralgorithmen

Recommender Systems arbeiten auf unterschiedliche Art und Weise je nach vorzuschlagenden Objekten und den zu Verfügung stehenden Daten. Im folgenden werden die vier klassischen Filteralgorithmen vorgestellt, die über den Entscheidungsprozess bestimmen. Jedes Recommender System hat dabei seinen eigenen Filteralgorithmus, der für das jeweils zu lösenden Problem optimiert sein sollte.

Jedoch bildet sich jeder dieser Algorithmen stets aus den folgenden vier Basisalgorithmen:

Demographic Filtering [BOHG13]
Das Demographic Filtering ist begründet durch die Annahme, dass Menschen mit demographischer Ähnlichkeit mit erhöhter Wahrscheinlichkeit gemeinsame Interessen haben. Diese Daten erstrecken sich von Geschlecht und Alter bis hin zu Nationalität. Zum Beispiel wäre es wahrscheinlicher einem weiblichen Nutzer beim Onlineshopping verschiedene Lippenstifte vorzuschlagen, als dies bei einem männlichen Nutzer zu tun.

Content-Based Filtering [BOHG13]]
Das Content-Based Filtering ist definiert durch die Vorgehensweise, dem Nutzer Objekte vorzuschlagen, die denen ähneln, für die sich der Nutzer in der Vergangenheit entschieden hat. Dabei können die Objekte in einem vorherigen Schritt analysiert werden und so durch verschiedene Attribute miteinander in Verbindung gebracht werden. Zum Beispiel wäre es sinnvoll einem Nutzer, der früher Science-Fiction Filme gekauft hat, weitere Filme dieses Genres oder ähnliche Produkte, die mit Science-Fiction zusammenhängen, vorzuschlagen. Texte ohne weitere Beschreibung könnten analysiert und miteinander verglichen werden.

Collaborative Filtering [BOHG13]]
Das Collaborative Filtering basiert auf Ratings, die Nutzer zu den verschiedenen Objekten vergeben. Dabei werden diese Ratings miteinander verglichen und Ähnlichkeiten im Geschmack und Interesse der Nutzer berechnet. Es werden dann dem Nutzer Objekte vorgeschlagen, für die sich andere Nutzer mit ähnlichem Verhalten in der Vergangenheit entschieden haben. Als Beispiel können hier die Ratings von Nutzern auf Amazon in Form von 1-5 Sternen angebracht werden. Zur Illustration der Komplexität des Collaborative Filterings soll hier kurz eine fiktive Situation erläutert werden. Wenn die Ratings von Nutzer A weitestgehend mit denen von Nutzer B übereinstimmen, ist es sinnvoll, Nutzer A Objekt 1 vorzuschlagen, welches Nutzer B hoch bewertet hat, aber von Nutzer A noch nicht bewertet oder gar angesehen wurden. Sollte Nutzer A jedoch kaum Ähnlichkeit mit Nutzer C haben und Nutzer C Objekt 1 auch hoch bewertet haben, sollte Nutzer A eventuell ein anderes Objekt vorgeschlagen werden. Solche Situationen und andere Probleme machen das Collaborative Filtering zu einem weitreichenden Forschungsgebiet. Die gängigste Methode ist der k Nearest Neighbors(kNN) Algorithmus.

Hybrid Filtering [BOHG13]
Das Hybrid Filtering ist eine Kombination aus den oben erwähnten Algorithmen. Dabei bestehen übliche Kombinationen aus dem Collaborative Filtering und dem Demographic Filtering oder aus dem Collaborative Filtering und dem Content-Based Filtering. Hierbei sollen Vorteile der jeweiligen Algorithmen kombiniert werden und im besten Fall eine Synergie erzeugen. Zum Beispiel könnte der

Collaborative Filtering Algorithmus für einen männlichen Kunden einen Lippenstift vorschlagen, welcher jedoch vom Demographic Filtering Algorithmus wieder ausgesiebt wird, da es unwahrscheinlich ist, dass hier ein Interesse besteht. Typische Machine Learning Algorithmen wie Neural Networks oder Clustering kommen beim Hybrid Filtering zum Einsatz.

2.1.4 Anwendung

Recommender Systems kommen in den verschiedensten Bereichen zum Einsatz. Im Folgenden wird eine Auflistung von Bereichen und Beispielen nach [LWM+14] zusammengefasst, um einen Überblick über die verschiedenen Anwendungsfelder für Recommender Systems zu geben. Außerdem dient diese Auflistung dazu, ein Bewusstsein zu schaffen, wie viele Teile der digitalen Welt durch Recommender Systems bestimmt sind.

E-Government
E-Government beschreibt das Vorgehen von Regierungen, Bürgern und Firmen, Informationen über das Internet und andere Technologien zukommen zu lassen. Der ständige Wachstum des E-Governments führt zu einer Überflutung an Informationen. An dieser Stelle können Recommender Systems helfen, passende Informationen herauszufiltern.

E-Business
E-Business umfasst technologische Produkte und Dienste von Firmen für Firmen (B2B). Wenn eine Firma eine Entscheidung zwischen verschiedenen Optionen treffen muss, kann zum Beispiel ein Recommender System, das das Verhalten anderer Firmen bezüglich dieses Themas vergleicht, hilfreich sein.

E-Commerce/E-Shopping
Im Gegensatz zum E-Business beschreibt E-Commerce/E-Shopping den Handel mittels technologischer Hilfsmittel zwischen Firmen und Konsumenten (B2C). Klassische Beispiele hierfür sind der Onlineshop Amazon [Ama] oder die Webseite eBay [eBa]. Recommender Systems helfen hier dabei, dem Nutzer aus der Flut an Produkten interessante Vorschläge zu unterbreiten.

E-Library
Digitale Bibliotheken sammeln und verwalten digitale Objekte und bieten dem Nutzer die entsprechenden Dienste an. Recommender Systems können hier zum schnellen Finden von wertvollen Informationen Beitragen, so wie bei einer Google-Suche [Goo] Webseiten nach diversen Kriterien geordnet aufgelistet werden, können in einer digitalen Bibliothek Objekte in einer nach verschiedenen Kriterien gewichteten Ordnung angezeigt werden.

E-Learning

E-Learning umfasst alle digitalen Dienste einer Bildungseinrichtung, die dem Nutzer beim Lernen unterstützen. Sowohl Lerninhalte als auch organisatorische Dinge wie digitale Stundenpläne oder Kursanmeldungen gehören dazu. Auch hier können Recommender Systems aus der Flut von Objekten interessante Vorschläge auswählen: Die Auswahl bezüglich genereller Themen, spezieller Kurse, Materialien zum Lernen, aber auch des Lernverhaltens soll hier unterstützt werden.

E-Tourism

Heutzutage ist es üblich, sich direkt im Internet über Reisemöglichkeiten zu informieren. Länder, Hotels, Restaurants und Sehenswürdigkeiten können durch verschiedene Kriterien wie Preise, aber auch durch das Verhalten anderer Nutzer und eigenes früheres Reiseverhalten vorgeschlagen werden.

E-Resource Service

Der Begriff E-Resource Service beschreibt den Dienst digitale Inhalte, die von Nutzern selbst hochgeladen wurden, zur Verfügung zu stellen. Das umfasst Texte, Dokumente, Videos, Musik, etc. Ein großer E-Resource Service ist die Webseite Youtube [You]. Auf Youtube können Nutzer eigene Videos hochladen. Auf der Startseite werden sofort Videos zum Anklicken vorgeschlagen. Dadurch wird dem Nutzer das Suchen nach interessanten Videos ein Stück weit abgenommen.

E-Group Activtiy

Sogenannte Group Recommender Systems schlagen nicht einzelnen Nutzern Objekte vor sondern Gruppen von Nutzern, die durch eine gemeinsame Aktivität in Verbindung stehen. Zum Beispiel kann ein Group Recommender System Musikvorschläge für ein Fitnessstudio machen, indem es die Musikinteressen der einzelnen Nutzer speichert und durch diverse Kriterien eine Auswahl trifft.

2.2 Microservices

"Microservices, Microservices Everywhere!" [SPA]

Dieses Zitat beschreibt den Hype um Microservices als Softwarearchitektur, der 2015 um das Thema entstanden ist. Das Recommender System für D-BAS soll als Microservice implementiert werden. Was ein Microservice ist, welche Vorteile diese Softwarearchitektur bietet und wie sie richtig angewendet wird, soll im Folgenden besprochen werden. Die Informationen zu diesem Kapitel entstammen der technischen Dokumentation *Microservices from Theory to Practice* [DDE+15].

2.2.1 Definition

'Microservices' ist ein Softwarearchitekturstil, der sich dadurch auszeichnet, dass eine große Softwareanwendung in kleinere Dienste unterteilt wird. Ein Mikroservice steht dabei für sich, ist weitestgehend von den anderen Diensten abgekoppelt und dient dazu, eine bestimmte Aufgabe, und nur diese Aufgabe, zu erfüllen.

2.2.2 Eigenschaften

Im Folgenden sollen die wichtigsten Eigenschaften eines Microservices näher erläutert werden.

Klein
Der Microservice sollte möglichst klein sein und muss sich auf eine Aufgabe fokussieren. Dabei gibt es keine genaue Richtlinie wie klein der Dienst sein muss. Der Fokus sollte dabei auf der Schnittstelle liegen, da eine kleine Schnittstelle meist mit einer kleinen Implementierung einhergeht. Es sollte allerdings auch beachtet werden, dass bei zu hoher Abhängigkeit von äußeren Daten beziehungsweise anderen Microservices es zu erhöhter Latenz und damit zu schlechter Performance der Gesamtanwendung kommen kann.

Abgekoppelt
Mikroservices müssen von anderen Diensten abgekoppelt sein und für sich stehen können. Der Dienst ist ein eigenständiges Softwareprogramm und wird auch als solches behandelt. Das bedeutet, dass jeder Dienst für sich fertiggestellt, ausgeliefert, gewartet und bearbeitet werden kann.

Sprachenunabhängig
Die Abkopplung von anderen Diensten ermöglichst es, den Microservice in jeder beliebigen Programmiersprache zu schreiben. Sprachenneutrale Schnittstellen ermöglichen hierbei die Kommunikation der Dienste untereinander. So kann für die jeweilige Aufgabe die optimale Programmiersprache gewählt werden beziehungsweise die unterschiedlichen Fähigkeiten der Programmierer bezüglich der Sprachen genutzt werden.

Beschränkt im Kontext
Gemeint mit dieser Eigenschaft ist, dass der Dienst nichts über die Implementierung der anderen Dienste, die ihn umgeben, weiß. Sollte der Dienst diese Information aus welchem Grund auch immer brauchen, ist diese Eigenschaft verletzt.

2.2.3 Vorteile

Viele Vorteile von Microservices sind bereits durch die Eigenschaften beschrieben: Sie sind klein, leicht und schnell zu bearbeiten, übersichtlich, sprachenunabhängig und wissen nicht mehr als sie wissen müssen. Im Folgenden werden weitere, aus diesen Eigenschaften resultierende Vorteile aufgelistet. Hierbei soll angemerkt sein, dass andere Quellen möglicherweise bei diesen Punkten anderer Auffassung sein können.

- Eine große Codebasis wird verhindert, was den Einbau neuer Features und die Wartung vor allem für neue Mitarbeiter erleichtert.
- Microservices können schneller herausgegeben werden als monolithische Programme und die Performance der Programmierer IDE wird durch den kleineren Code deutlich verbessert.
- Die Fehlersuche wird vereinfacht.
- Teams können voneinander unabhängig arbeiten.
- Abhängigkeiten im Code sind einfacher zu verfolgen vor allem für neue Mitarbeiter leichter zu erkennen.
- Über den gesamten Entstehungszyklus hinweg kann ein einzelnes Team an einem Dienst arbeiten. Der Dienst gehört sozusagen nur diesem einen Team.
- Engpässe in der Leistung sind leichter zu bearbeiten.

- Der gesamte Prozess des Testens wird beschleunigt und vereinfacht.

- Produkte können schneller an Kunden ausgeliefert werden.

2.2.4 Nachteile und Anwendung

Es ist in 2015 ein Hype um Microservices entstanden. Diese Softwarearchitektur liegt im Trend. Doch wie jeden Trend sollte auch dieser Trend hinterfragt werden. Die Verwendung von Microservices ist kein Allheilmittel. Es werden im Folgenden die wichtigsten Hinweise erläutert, die vor der Entscheidung, Microservices zu verwenden, beachtet werden sollten. Diese Hinweise zeigen außerdem die wichtigsten Nachteile von Microservices auf, sowohl jene, die immanent in der Architektur gegeben sind, als auch solche, die durch falsche Nutzung entstehen.

Start mit Microservices
Microservices haben einen Sinn. Große Probleme werden in kleine Probleme aufgeteilt. Dafür muss allerdings erst ein großes Problem vorhanden sein. Am Anfang ist eine Softwareanwendung klein genug, um sie als ganzes beizubehalten. Erst bei Problemen der Übersicht oder Datenflut sollte über Microservices nachgedacht werden.

Zu viele Dienste
Da jeder neue Dienst Ressourcen verbraucht, sollte unbedingt geprüft werden, ob die Anhäufung dieses Verbrauchs nicht die Vorteile der Softwarearchitektur negiert. Die Komplexität, die durch Microservices aufgehoben wird, kann sich bei zu vielen Diensten in der Integration dieser Dienste wiederfinden.

Das Latenzproblem
Bei zu vielen untereinander vernetzten Diensten kann es erhöhter Latenz in der Netzwerkkommunikation kommen. Daher sollten Zeitmessungen bei der Kommunikation zwischen den Diensten erfolgen, um Engpässe zu finden und zu beheben.

Kapitel 3

Implementierung

3.1 Aufbau

3.1.1 Schnittstelle zum Hauptprogramm

Zum Verständnis der Arbeitsweise und des generellen Aufbaus, wird im Folgenden die Schnittstelle zu D-BAS dargestellt. Da Microservices über HTTP Anfragen angesprochen werden [DDE+15], ergibt sich die Möglichkeit über entsprechende Pfade die Parameter dieser Anfrage festzulegen:

1. /positions/:userID/:issueID
2. /premisesforstatement/:userID/:issueID/:statementID/:attitude/:isPosition
3. /putstatementargument/:userID/:issueID/:statementID/:attitude/:isPosition/:argumentID
4. /premisesforattack/:userID/:issueID/:targetArgumentID/:attackType
5. /putattackargument/:userID/:issueID/:targetArgumentID/:attackType/:argumentID

Hierbei sind die Paramter der Form ':x' variabel und enthalten die für die Bearbeitung der Aufgabe nötigen Werte. Der Pfad /positions/1/5 bedeutet somit, dass Positionen für den Nutzer mit der ID 1 und zu dem Thema mit der ID 5 angefragt werden. Realisiert wird diese Schnittstelle im Programm mit dem Microframework *Spark* [SPA] für Java 8 [Jav].

3.1.2 Programmstruktur

Bei jedem neu zu schreibenden Programm stellt sich die Frage nach der grundlegenden Struktur des Programms. Die Aufgabe des Microservices lässt sich bei fast allen Pfaden in zwei Teilaufgaben unterteilen:

1. Vergleiche den Nutzer mit allen anderen Nutzern

2. Entferne auf Basis dieses Vergleichs Schritt für Schritt Objekte, bis die Top 5 zurückgegeben werden können

Da das Programm weder eine Benutzeroberfläche hat noch irgendwelche Eingaben (bis auf die Anfragen) getätigt werden, können diese Aufgaben durch eine einfache Klassenverschachtelung abgearbeitet werden. Nach der Auswahl der Aufgabe durch den Pfad und die weitergabe der Parameter wird die entsprechende Methode der Top 5 Objekte aufgerufen, welche wiederum die jeweils passende Klasse und deren Methode zum Vergleich der Nachbarn aufruft. Es ergibt sich eine Programmstruktur, die ohne weitere Eingaben ähnlich einer einfachen Methode mithilfe einiger Hilfsmethoden auf gegebene Parameter ein Resultat zurückgibt.

3.1.3 Datenbank und Pfade

Da jeder Microservice seine eigenen Datenbank haben sollte [DDE+15], wurde dies durch eine eigene SQL Datenbank verwirklicht. Hierbei wurde für jeden Pfad eine eigene Tabelle angelegt:

positions_requests
Diese Tabelle entspricht dem Pfad 1 der Schnittstelle und dient zur Speicherung aller Anfragen bezüglich der Positonen zu einem bestimmten Thema.
Spalten: *id, user_id, issue_id, count, last_request*

premises_for_statement_requests
Diese Tabelle entspricht dem Pfad 2 der Schnittstelle und dient zur Speicherung aller Anfragen bezüglich der Prämissen für eine Meinung zu einer bestimmten Aussage. Desweiteren befinden sich in dieser Tabelle alle Positionen zu einem Thema, so dass der Microservice diese einfach abfragen kann.
Spalten: *id, user_id, issue_id, statement_id, attitude, is_position, count, is_valid, last_request*

statement_arguments
Diese Tabelle entspricht dem Pfad 3 der Schnittstelle und dient zur Speicherung aller gebildeten Argumente (Konklusion + Prämisse) bezüglich der Meinung zu einer bestimmten Aussage.
Spalten: *id, user_id, issue_id, statement_id, attitude, is_position, argument_id, count, is_valid, last_request*

premises_for_attack_requests
Diese Tabelle entspricht dem Pfad 4 der Schnittstelle und dient zur Speicherung aller Anfragen bezüglich der Prämissen für einen Angriff/Angriffstyp auf ein bestimmtes Zielargument.
Spalten: *id, user_id, issue_id, target_argument_id, attack_type, count, is_valid, last_request*

attack_arguments
Diese Tabelle entspricht dem Pfad 5 der Schnittstelle und dient zur Speicherung aller gebildeten Argumente (Konklusion + Prämisse) bezüglich eines Angriffs/Angrifftyps auf ein bestimmtes Zielargument.
Spalten: *id, user_id, issue_id, target_argument_id, attack_type, argument_id, count, is_valid, last_request*

Diese fünf Tabellen bilden die Datenbank für den Microservice, wobei jede wie erläutert ihrem eigenen Zweck dient. Es stellen sich hierbei jedoch einige Fragen, auf die im Folgenden eingegangen wird.

Wozu die Spalten count, is_valid und last_request?
Die Spalte 'count' dient als einfacher Integerwert dazu, zu zählen, wie oft eine identische Anfrage an den Microservice geschickt wurde. Die Spalte 'is_valid' dient als boolescher Wert dazu, anzuzeigen, ob diese Meinung des Nutzers aktuell gültig ist. Nutzer können natürlich in D-BAS immer ihre Meinung ändern. Dies wird hierdurch berücksichtigt. Die Spalte 'last_request' dient als boolescher Wert zum Schutz vor dem, was intern als 'F5-Problem' bezeichnet wurde: Der Nutzer könnte die URL an einer Stelle der Diskussion wiederholt aufrufen und somit den count-Wert künstlich erhöhen. Durch eine einfache Überprüfung dieses Wertes kann entschieden werden, ob der Eintrag aktualisiert wird oder nicht.

Viele gespeicherte Werte wirken redundant

Dieses Problem ist im Verlauf der Arbeit besprochen worden. Am Beispiel der Datenbanktabellen *premises_for_statement_requestst* und *statement_arguments* soll erläutert werden, wie und warum hier Daten redundant gespeichert wurden. Schaut man sich die beiden Tabellen an, sieht man, dass in der Tabelle *statement_arguments* alle Spalten identisch sind und nur eine Spalte hinzugefügt wurde. Man könnte nun argumentieren, dass in der Tabelle *statement_arguments* ein Fremdschlüssel zu bezüglich der Tabelle *premises_for_statement_requests* eingebaut werden sollte, um diese Redundanz zu verhindern. Es gibt jedoch ein praktisches Problem, welches durch diese Redundanz umgangen wird und hier an einem Beispiel erläutert wird: Nutzer A und Nutzer B seien eingeloggt. Nutzer A hat gerade einen Eintrag in die Tabelle *statement_arguments* getätigt, indem er seine Meinung zu einer Position begründet hat. Nutzer B ist neu registriert und hat ein anderes Thema ausgewählt. Nutzer A schickt nun Nutzer B die URL zu der Stelle in der Diskussion, an der sich Nutzer A befindet. Nutzer B klickt auf diesen Link und D-BAS will für Nutzer B nun einen entsprechenden Eintrag in die Tabelle *statement_arguments* machen. Da Nutzer B aber noch nie in der Tabelle *premises_for_statement_requests* aufgetaucht ist, kommt es hier zu einem Fehler in der Datenbank, da kein entsprechender Fremdschlüssel für die Tabelle *statement_arguments* gefunden werden kann. Dieses Problem ist in D-BAS verankert und die Redundanz in der Datenbank könnte und sollte nach dem Lösen dieses Problems aufgehoben werden.

3.2 Grundlegende Algorithmen

Für das Recommender dieser Arbeit wurde das Collaborative Filtering als Filtermethode gewählt, da demographische Angaben in D-BAS freiwillig und damit meist unvollständig sind. Außerdem müsste für ein Content-Based-Filtering eine Textanalyse stattfinden, die den Rahmen dieser Arbeit sprengen würde. Zur Erinnerung wird hier noch einmal angemerkt, dass es sich bei dieser Arbeit um einen Proof of Concept handelt. Beim der gewählten Filtermethode werden die anderen Nutzer mit dem aktiven Nutzer verglichen. Auf dieser Basis werden dem Nutzer Vorschläge ausgewählt. Im Folgenden werden die Algorthimen besprochen, die diesen Auswahlprozess bestimmen.

3.2.1 Nachbarstypen

Wenn man zwei Nutzer miteinander vergleichen möchte, muss zuerst klargestellt werden, welche Attribute dieser Nutzer verglichen werden sollen. Zur Erinnerung: Es sind drei Aufgabe vom Recommender System zu lösen.
Die erste Aufgabe besteht darin, Positionen zu einem Thema vorzuschlagen. Daher macht es Sinn, im ersten Schritt jene Nutzer als nähere Nachbarn anzusehen, die ein ähnlicheres Interesse an Positionen

haben, wie der aktuelle Nutzer. Beispiel: Der aktuelle Nutzer A und auch Nutzer B haben sich für Thema 1 interessiert und dort ihre Meinungen zu den Positionen 1 und 2 (jeweils mehrmals) zum Ausdruck gebracht. Nutzer C interessiert sich für das selbe Thema jedoch eher für die Positionen 3 und 4. Wenn man nun in Thema 2 Positionen für Nutzer A vorschlagen möchte, würde man (bei der Wahl zwischen Nutzer 2 und Nutzer 3) Nutzer 2 vorziehen, da dieser ein ähnlicheres Interesse aufzeigt.
In einem zweiten Schritt kann man nun auch das Interesse an den Themen an sich mit einbeziehen. In unserem Beispiel gäbe es dann zB noch Nutzer D, welcher sich überhaupt nicht für Thema 1 interessiert. Diesem Nutzer würde man hier also Nutzer 2 und Nutzer 3 vorziehen.

Für Problem 1 der Arbeit ergeben sich folgende zwei Nachbarstypen:
Positionsinteresse und **Themeninteresse**

Die Probleme 2 und 3 der Arbeit fragen beide nach Prämissen. Hier liegt es nahe, sich die in der Vergangenheit getätigte Meinung der Nutzeranzuschauen, unabhängig von Themen- oder Positioninteresse. Hierbei ist auch von Interesse, wie groß die Schnittmenge der bewerteten Aussagen und Argumenten ist. Weiterhin kann man auch auch bei diesen Problemen in einem zweiten Schritt anschauen, wie groß das Interesse an den unterschiedlichen Themen ist, da sich Meinungen eher ändern, wenn sie einer Vielzahl anderer Meinungen gegenüberstehen.

Für Problem 2 und Problem 3 der Arbeit ergeben sich folgende zwei Nachbarstypen:
Meinung und **Themeninteresse**

Warum keine Berücksichtigung der Meinung für Problem 1?
Problem 1 befasst sich damit, welche Positionen für einen Nutzer interessant sein könnten. Ob eine Position interessant ist oder nicht, hängt nicht von der Meinung zu ihr ab. Zum Beispiel könnte eine Familie darüber streiten, ob sie einen Hund oder eine Katze haben wollen. Zu diesem Thema gibt es vier mögliche Positionen:

- Ich will einen Hund und eine Katze!
- Ich will nur einen Hund!
- Ich will nur eine Katze!
- Ich will weder einen Hund noch eine Katze!

Jeder dieser Positionen kann man mit Zustimmung oder Ablehnung begegnen. Je nachdem, wie sehr jemand auf seine Meinung beharren möchte und sie ausdrücken will, können alle Positionen von Interesse sein. Andernfalls, kann einem Nutzer das Thema auch egal sein. Er möchte aber zum Beispiel auf keinen Fall zwei Haustiere haben. Hier wäre nur die eine Position für ihn von Interesse. Dieses Bedürfnis wird durch das allgemeine Themeninteresse abgedeckt. Ob andere Nutzer generell ähnlicher Meinung sind, spielt hier keine Rolle.

Warum keine Berücksichtigung des Positionsinteresses für die Probleme 2 und 3?
Die Probleme 2 und 3 befassen sich damit, dass ein Nutzer eine Prämisse für seine Meinung nennen soll. Es spielt dabei keine Rolle, wie oft ein Nutzer seine Meinung zu einer Position ausgedrückt hat. Ob sich ein Nutzer mit dem Thema länger auseinander gesetzt hat oder nicht, wird durch das Themeninteresse berücksichtigt. Außerdem sind Aussagen und Argumente nicht auf nur eine Position innerhalb eines Themas beschränkt, sondern können Diskussionsgegenstand mehrere Positionen sein.

3.2.2 Quantifizierung von Meinung und Interesse

Im Folgenden wird zu jedem Nachbarstyp erklärt, wie genau durch die in der Datenbank gespeicherten Daten ein Zahlenwert für jedes Paar Nutzer-Objekt berechnet wird.

Positions- und Themeninteresse
Das Positionsinteresse eines Nutzers wird durch den 'count'-Wert der entsprechenden Spalte der Tabellen *premises_for_statement_requests* und *statement_arguments* in der Datenbank bestimmt. Zu jeder Position werden die count-Werte beginnend vom Wert 0 aufaddiert.
Die Berechnung des Themeninteresses verläuft identisch. Allerdings werden hier alle Tabellen angeschaut. Das ist der Grund, warum das Thema (issue_id) in jeder Tabelle der Datenbank aufgelistet ist. Im Anschluss wird ein allgemeiner count-Wert über alle Objekte gebildet und für jedes Objekt ein relativer Wert berechnet. Es gilt also für Interesse I:

$$I_{relativ} = \frac{I_{speziell}}{I_{allgemein}} \tag{3.1}$$

Für den im nächsten Kapitel folgenden Vergleichsalgorithmus wird allerdings ein Rating benötigt. Hierfür werden die relativen Werte auf das Intervall [0,1] in Relation zum höchsten Wert normiert. Mit

$$c = \frac{1}{I_{relativ_{max}}} \tag{3.2}$$

gilt also für das Objektrating R:

$$R = c \cdot I_{relativ} \tag{3.3}$$

Meinung

Wie auch beim Positions- beziehungsweise Themeninteresse braucht der Algorithmus bei der Meinung ein Rating. Dieses ist aber im Gegensatz zum Interesse bei einer Meinung direkt gegeben. Der 'attitude'-Wert in den Tabellen *premises_for_statement_requests* und und *statement_arguments* ist beschränkt auf die Werte '-1', '0' und '1'. Der Wert '1' beschreibt, dass dem Objekt zugestimmt wird, während der Wert '-1' ausdrückt, dass man anderer Meinung ist. Der Wert '0' hingegen besagt, dass der Nutzer keine Meinung zu diesem Objekt hat (aktiv so angegeben, der User wurde zu diesem Objekt befragt). Somit ist eine Ratingsskala von '-1' bis '1' gegeben und keine weitere Berechnung ist nötig. Allerdings kann ein Objekt in diesem Fall auch gar nicht bewertet worden sein. Themen oder Positionen, die nie angeklickt wurden, kann man mit einem Nullinteresse bewerten. Aber bei einer aktiven Meinung zu einem Objekt fehlt ein Wert, der besagt, dass das Objekt gar nicht bewertet wurde, weil der Nutzer es nie angeklickt hat. Im Programmiercode wird dieses Objekt mit der Meinung '-2' versehen und durch den Vergleichsalgorithmus entsprechend behandelt.

3.2.3 Der Vergleichsalgorithmus NHSM

Es wurden den einzelnen Nutzern zu jedem Objekt ein persönliches Rating zugeordnet. Nun müssen die Nutzer untereinander verglichen werden, um ihre Nähe als Nachbarn zu quantifizieren. Dabei wird für jeden Nachbarstyp einzeln verglichen und eine eigene Nachbarschaftsmatrix erstellt. Als Algorithmus wurde hierfür das *New Heuristic Similarity Model* (NHSM) gewählt, welches im Paper [LHM+13] detailiert beschrieben wird. Im Folgenden werden die drei Säulen des NHSMs erläutert und welche Probleme durch sie gelöst werden.

Proximity-Significance-Singularity (PSS)

Diese drei Begriffe bilden die erste Säule des Algorithmus.

- Proximity: Es wird berücksichtigt, wie weit die Ratings r für das Objekt p beider Nutzer u und v, die verglichen werden, auseinanderliegt.

$$Proximity(r_{u,p}, r_{v,p}) = 1 - \frac{1}{1 + exp(|r_{u,p} - r_{v,p}|)} \tag{3.4}$$

- Significance: Es wird angenommen, dass ein Rating r signifikanter ist, je weiter es vom Medi-

an der Ratingsskala r_{med} entfernt ist. Das bedeutet für unser Recommender System, dass sehr geringes Interesse und hohes beziehungsweise höchstes Interesse, sowieso Zustimmung bziehungsweise Ablehnung eines Objektes besonders hoch gewertet werden.

$$Significance(r_{u,p}, r_{v,p}) = \frac{1}{1 + exp(|r_{u,p} - r_{med}| \cdot |r_{v,p} - r_{med}|)} \tag{3.5}$$

- Singularity: Hierdurch wird verglichen, wie verschieden die beiden verglichenen Ratings vom mittleren Rating μ des Objektes p sind.

$$Singularity(r_{u,p}, r_{v,p}) = 1 - \frac{1}{1 + exp(|\frac{r_{u,p} + r_{v,p}}{2} - \mu_p|)} \tag{3.6}$$

Die Ähnlichkeit der beiden Nutzer u und v wird nun berechnet durch:

$$sim(u,v)^{PSS} = \sum_{p \in O} PSS(r_{u,s}, r_{v,p}) \tag{3.7}$$

Mit O als Menge aller Objekte und

$$PSS(r_{u,s}, r_{v,p}) = Proximity(r_{u,s}, r_{v,p}) \cdot Significance(r_{u,s}, r_{v,p}) \cdot Singularity(r_{u,s}, r_{v,p}) \tag{3.8}$$

Jaccard

Als zweite Säule dient die Berechnung des Jaccardwertes, welcher berücksichtigt, wie viele gemeinsame Objekte aus der Menge O die beiden verglichenen Nachbarn u und v bewertet haben. Es ergibt sich folgende Jaccard-Ähnlichkeit:

$$sim(u,v)^{Jaccard} = \frac{|O_u \cap O_v|}{|O_u| \cdot |O_v|} \tag{3.9}$$

Anmerkung: Dieser Wert ist bei Positions- und Themeninteresse immer $\frac{1}{|O|}$, da absolutes Desinteresse an einem Thema oder einer Position mit der Nullwertung versehen wird. Beim Nachbarstyp Meinung kommt hier das oben genannte Rating '-2' zum Tragen, welches ein Objekt aus der Schnittmenge ausschließt.

User Rating Preference (URP)

Die dritte und letzte Säule des NHSMs beschreibt das allgemeine Ratingverhalten der verglichenen Nutzer. Manche Nutzer tendieren zu hohen Wertungen, andere Nutzer zu niedrigen Wertungen. Es werden die mittleren Ratings μ sowie die Standardvarianzen σ der beiden Nutzer u und v miteinander

verglichen. Es ergibt sich folgende URP-Ähnlichkeit:

$$sim(u,v)^{URP} = \frac{1}{1+exp(\mu_u - \mu_v| \cdot |\sigma_u - \sigma_v|)} \tag{3.10}$$

Weniger interessant für den Meinungsvergleich, ist diese Säule sehr hilfreich beim Positions- und Themeninteressenvergleich. Niedrige Varianz und ein hohes mittleres Rating deuten zum Beispiel auf ein breites Interessenspektrum hin. Es macht Sinn, zwei Nutzern einen niedrigeren Wert zuzuordnen, wenn Nutzer u sich nur für wenige Objekte interessiert und Nutzer v ein breiteres Spektrum abdeckt.

Durch diese drei Säulen ergibt sich nun im Vergleich der beiden Nutzer u und v endgültige NHSM-Ähnlichkeit:

$$sim(u,v)^{NHSM} = sim(u,v)^{PSS} \cdot sim(u,v)^{Jaccard} \cdot sim(u,v)^{URP} \tag{3.11}$$

3.2.4 Auswahl der Objekte

Die Auswahl der vorgeschlagenden Objekte wird durch ein Ausscheidungsverfahren getroffen. Üblicherweise werden die Top N Nachbarn gebildet und dann durch diese eine Auswahl der Top N Objekte getroffen. Allerdings ist es für eine Diskussionsplattform passender, wenn höchstwahrscheinlich uninteressante Objekte aus der Auswahl entfernt werden, anstatt eine direkte Auswahl für den Nutzer zu treffen. So kann je nach Einstellung des Systems am Ende ein Grad von zufälligen Vorschlägen gewahrt bleiben, ohne eventuell interessante Vorschläge nie in Erwägung zu ziehen.

Anfrage von Positionen

Werden Positionen zu einem Thema angefragt, werden zuerst alle Positionen dieses Themas aufgelistet. Wie in Kapitel 4.2.1 erklärt, wird erst nach Positionsinteresse ausgesiebt. Es wird eine Liste mit allen Vergleichswerten zu den anderen Nutzern erstellt. Dann wird der kleinste Wert (die geringste Übereinstimmung) und der entsprechende Nutzer herausgesucht. Der Algorithmus überprüft nun, ob dieser Nutzer ein exklusives Interesse an einem der aufgelisteten Positionen hat. Ist dies der Fall, wird diese Position aus der Liste rausgenommen. Der Nutzer wird anschließend aus der Liste der Nachbarn entfernt. Dieser Prozess wiederholt sich so lange, bis ein Grenzwert von Nutzern entfernt wurde. Dieser Wert ist im Code variabel. So wird schrittweise die Menge an Positionen verringert. Nachdem der Grenzwert erreicht ist, endet das Aussieben nach Positionsinteresse und mit dem gleichen Verfahren startet das Aussieben nach Themeninteresse. Der Algorithmus endet vorzeitig, wenn durch das Entfernen einer Position die Anzahl auf fünf Positionen schrumpft. Sollte dies zutreffen, werden genau diese Positionen vom Recommender System vorgeschlagen. Sollte der Algorithmus hingegen am Ende mehr als fünf Positionen vorschlagen, fällt das Recommender System auf das alte System der zufälligen Auswahl zurück, bis nur noch fünf Positionen vorhanden sind.

Anfrage von Prämissen

Die Anfrage von Prämissen wird äquivalent zur Anfrage Positionen bearbeitet. Der Unterschied liegt nur darin, dass statt des Positionsinteresses die Meinung der Nutzer im ersten Schritt über das Aussieben entscheidet. Im zweiten Schritt bleibt das Themeninteresse. Auch hier wird am Ende falls nötig auf das Zufallssystem zurückgegriffen.

Kapitel 4

Evaluation

In diesem Kapitel wird geprüft, ob sich die Implementierung des Microservices wie gewünscht verhält. Die Evaluation besteht dabei aus zwei Teilevaluationen: Die erste Evaluation beschäftigt sich mit dem Positionsinteresse, während die zweite Evaluation sich mit den Meinungen auseinandersetzt. Das Themeninteresse wird durch die Evaluation des Positionsinteresses mitbesprochen, da der zugrunde liegende Algorithmus der selbe ist. Für diese Evaluation wurden zwei Javaklassen zum Erfassen der Daten geschrieben. Die Graphen wurden durch ein Pythonskript unter der Verwendung der Bibliotheken Numpy [Num] und Matlibplot [Mat] erstellt.

4.1 Einstellungen

Um die Evaluationen zu verstehen, ist es wichtig zu wissen, wie das Recommender System eingestellt wurde.

Aussiebungsmenge
Es wird die Hälfte der Nachbarschaft zur Aussiebung der Objekte herangezogen. Nach dieser Hälfte wird stoppt der Algorithmus.

Vorschlagsmenge
Da in beiden Szenarien acht Nutzer in zwei Gruppen aufgeteilt werden und acht Objekte vorhanden sind, werden jedem Nutzer vier Objekte vorgeschlagen.

4.2 Positionsinteresse

Das Positionsinteresse wird evaluiert, indem ein künstliches Szenario erzeugt wird, das Nutzer in zwei Gruppen aufteilt und die Restriktionen für diese Gruppen immer weiter lockert. Die gesammelten Daten besagen, wie oft eine Position vom Recommender System vorgeschlagen wurde und werden als Graphen dargestellt. Diese Graphen werden im Anschluss qualitativ ausgewertet und besprochen, um zu beurteilen, ob das gewünschte Verhalten erkennbar ist.

4.2.1 Szenario

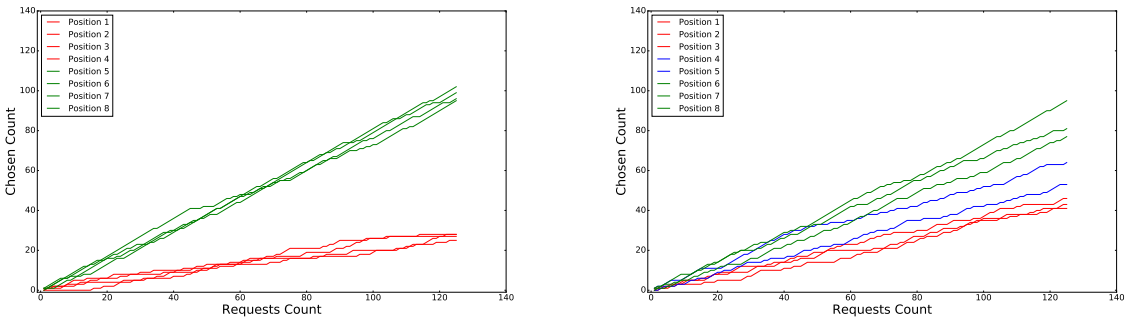
Das Szenario besteht aus acht Nutzern und acht Positionen. Jeder Nutzer hat eine dieser Positionen erstellt. Dabei werden die Nutzer in zwei Gruppen aufgeteilt. Die Gruppen werden durch die Positionen definiert, unter denen sie zufallsbasiert ihr Interesse ausdrücken. Es finden fünf Durchläufe statt, wobei im ersten Durchlauf die beiden Gruppen klar voneinander getrennt sind. Gruppe 1 darf nur zwischen Position 1, 2, 3 und 4 wählen, während Gruppe 2 zwischen den Positionen 5, 6, 7 und 8 wählen muss. Der letzte Durchlauf setzt die beiden Gruppen gleich und den Nutzern werden alle Positionen zur Verfügung gestellt. Die Durchläufe 2, 3 und 4 lockern die Restriktion nach und nach auf. Zum Beispiel dürfen in Durchlauf 2 die Nutzer aus Gruppe 1 auch Position 5 wählen und in Durchlauf 3 kommt Position 6 hinzu. Jeder Durchlauf besteht dabei aus 1000 Wahlaktionen, die von den 8 Nutzern der Reihe nach abgearbeitet werden. Demnach wählt jeder Nutzer pro Durchlauf 125 mal eine Position. Jede Wahl wird in die Datenbanktabelle *premises_for_statement_requests* gespeichert. Danach werden dem Nutzer vier Positionen vorgeschlagen. Für jeden Nutzer wird die Anzahl an Vorschlägen bezüglich jeder Position gespeichert.

4.2.2 Erwartungen

Es wird erwartet, dass in Durchlauf 5 nach einer gewissen Zeit jede Position im Mittel mit der gleichen Häufigkeit vorgeschlagen wird. Wenn von jedem Nutzer jede Position einmal gewählt wurde, kann der Algorithmus keine exklusiven Positionen mehr finden und muss auf ein Zufallsverfahren zurückgreifen. In den anderen Durchläufen sollte der Trend erkennbar sein, dass die für diesen Nutzer nicht erlaubten Positionen signifikant weniger häufig als die exklusiven Positionen vorgeschlagen werden. In Durchlauf 1 wird eine klare Trennung zwischen Gruppe 1 und Gruppe 2 erwartet.

4.2.3 Daten

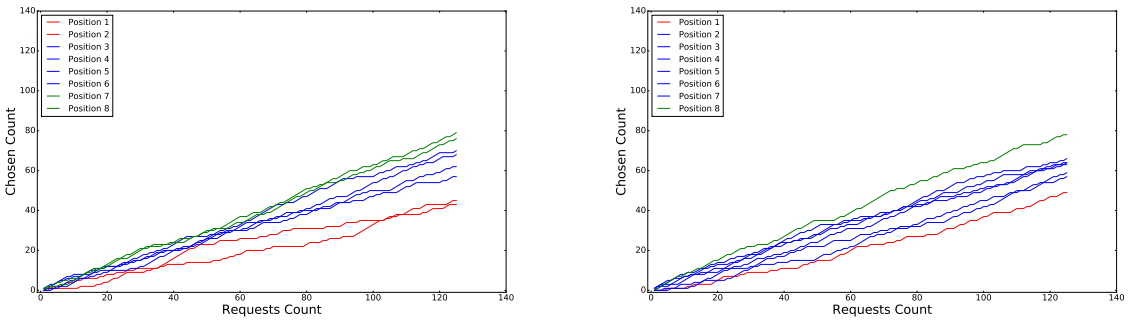
Im Folgenden werden die Daten für die Evaluation des Algorithmus bezüglich des Positionsinteresses in Form von Graphen präsentiert. Diese Daten stammen von Nutzer 8. Dieser Nutzer wurde gewählt, da zu dem Zeitpunkt, an dem Nutzer 8 seine Vorschläge erhält, jeder Nutzer die gleiche Anzahl an Wahlaktionen getätigt hat. Dabei wird die Anzahl der Wahlaktion des Nutzers gegen die Häufigkeit von Vorschlägen bezüglich jeder Position aufgetragen. Zur besseren Illustration wurden bei den Durchläufen 1-4 die Positionen dabei in verschiedene Farbgruppen eingeteilt, wobei den Positionen, die exklusiv der Gruppe des Nutzers zugeordnet wurden, die Farbe grün, den Positionen, die beiden Gruppen offen stehen, die Farbe blau, und den Positionen, die der Nutzer nicht wählen durfte, die Farbe rot gegeben wurde. Für Durchlauf 5 wurde zur besseren Unterscheidung jeder Position eine eigene Farbe gegeben, da es hier keine Restriktionen mehr gibt.



(a) Durchlauf 1

(b) Durchlauf 2

Abbildung 4.1: Verlauf der Vorschläge von Positionen, Durchläufe 1 und 2



(a) Durchlauf 3

(b) Durchlauf 4

Abbildung 4.2: Verlauf der Vorschläge von Positionen, Durchläufe 3 und 4

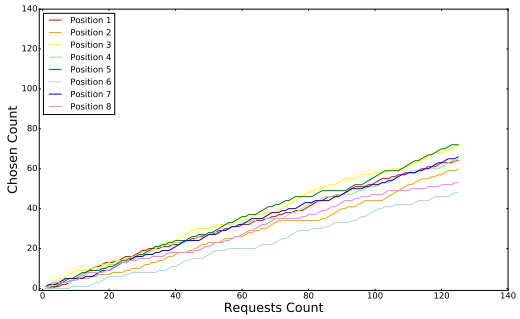


Abbildung 4.3: Verlauf der Vorschläge von Positionen, Durchlauf 5

4.2.4 Analyse und Auswertung

Durchlauf 1

Der Graph zeigt wie erwartet eine klare Schere zwischen den vorgeschlagenen Positionen. Hier arbeitet das Recommender System offenbar wie erwartet. Es ist auch zu sehen, dass nicht nur die gewählten Positionen und damit die Ratings für die Positionen selbst eine Rolle bei der Definition der Nachbarschaftsverhältnisse spielen, sondern die anderen Aspekte des Vergleichsalgorithmus den Nutzern aus Gruppe 1 eine Chance geben, nicht ausgesiebt zu werden.

Durchläufe 2, 3 und 4

Die Entwicklung der Graphen für diese Durchläufe zeigt sich ebenfalls wie erwartet. Die Schere schließt sich mehr und mehr, jedoch werden immernoch die exklusiven Positionen häufiger vorgeschlagen als die verbotenen Positionen, während die überlappenenden Positionen in der Mitte dieser beiden Gruppen auffindbar sind.

Durchlauf 5

Der Graph zeigt auf den ersten Blick eine Bevorzugung gewisser Positionen gegenüber anderen Positionen. Vor allem Position 6 wird sehr viel seltener vorgeschlagen. Sie wird sogar seltener vorgeschlagen als Position 1 in Durchlauf 4. Es ist jedoch auch erkennbar, dass Position 6 vor allem am Anfang kaum vorgeschlagen wird. Dies kann als eine Art Offset angesehen werden, der den Verlauf des Graphen auf der x-Achse verschiebt, da in der Anfangsphase noch nicht jeder Nutzer jede Position gewählt hat und Position 6 vermutlich von den wenigsten Nutzern bis dahin gewählt wurde. Daher ist die Anfangsphase als eine mögliche Ursache für die große Verteilung anzusehen.

4.3 Meinung

Der Algorithmus zur Aussiebung unter Berücksichtigung der Meinung der Nutzer zu einer Aussage oder eines Arguments wird ebenfalls durch ein künstliches Szenario evaluiert. Es werden wieder Nutzer in zwei Gruppen aufgeteilt. Welche Veränderungen sich durch der Unterschiede zwischen den beiden Algorithmen ergeben, wie die Daten aussehen und ob das Programm den Erwartungen entsprechend arbeitet, wird im Folgenden besprochen.

4.3.1 Szenario

Das Szenario besteht aus acht Nutzern, neun Statements und acht Prämissen. Dabei dienen die Statements 1-8 zur Meinungsveranschaulichung der Nutzer, während die acht Prämissen für eine zustimmende Meinung zu einem Statement 9 als Vorschläge dienen. Jeder Nutzer gibt 125 mal seine Meinung der Reihe nach zu den Statements 1-8 ab. So gibt Nutzer 1 anfangs zu Statement 1 seine Meinung ab und wenn alle anderen Nutzer ihre Meinungen zu ihren zugeordneten Statements abgegeben haben, ist Nutzer 1 wieder an der Reihe und gibt seine Meinung zu Statement 2 ab. Dabei werden die Nutzer in zwei Gruppen eingeteilt. Ähnlich wie im Szenario zum Positionsinteresse werden fünf Durchläufe gemacht. In Durchlauf 1 werden Gruppe 1 die Statements 1-4 und Gruppe 2 die Statements 5-8 zugeordnet. In Durchlauf 2 dürfen Gruppe 1 zusätzlich zu Statement 5 und Gruppe 2 zusätzlich zu Statement 4 ihre Meinungen äußern bis in Durchgang 5 beiden Gruppen alle Statements zugeordnet sind. Ein weiteres Unterscheidungsmerkmal der Gruppen ist ihre Meinung. Bis 500 mal die Meinung abgegeben wurde, stimmt Gruppe 1 immer den Statements zu, während Gruppe 2 die Statements ablehnt. Ab 500 Meinungsbekundungen ändern die Nutzer 3, 4 und 8 ihre Meinungen. Nachdem ein Nutzer seine Meinung abgegeben hat, fragt er Vorschläge für Prämissen zu seiner positiven Meinung zu Statement 9 an. Jeder Nutzer hat eine dieser Prämissen erzeugt. Der anfragende Nutzer wählt keine dieser Prämissen aus. Es wird nur die Häufigkeit für jeden Nutzer gespeichert, in der sie vorgeschlagen werden. So bleiben die Prämissen für die einzelnen Nutzer exklusiv und an der Aussiebung der Prämissen kann eindeutig erkannt werden, welche Nutzer sich in der ausgesiebten Hälfte der Nachbarschaft befanden.

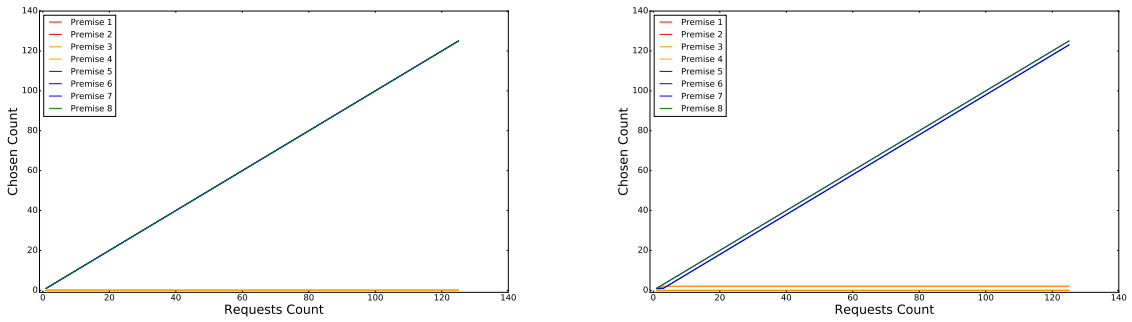
4.3.2 Erwartungen

Es wird Erwartet, dass die Aussiebung sehr viel strenger und weniger vom Zufall abhängt, als beim Positionsinteresse. Durch die Option, dass Nutzer gar kein Rating (ihre Meinung) zu einem Objekt abgeben, kann es zu einer kompletten Ignorierung für Nutzer aus Gruppe 1 bezüglich Nutzer aus

Gruppe 2 kommen. Dies wird vor allem im Durchlauf 1 erwartet, da es hier keine Schnittmenge der bewerteten Statements gibt. Im Gegensatz zum Algorithmus bezüglich des Positionsinteresses ist hier außerdem die Anzahl der abgegeben Meinung irrelevant. Sobald jeder Nutzer einer Gruppe seine Meinung zu jedem ihm zugeordneten Statement abgegeben hat, dürften sich die vorgeschlagenen Prämissen nicht mehr ändern, bis die Nutzer ihre Meinung ändern. Nach der Hälfte eines Durchlaufs wird sich das jedoch ändern, da es nun nur noch drei Nutzer gibt, die ihren Statements zustimmen und fünf Nutzer bei ihren Statements angeben, anderer Meinung zu sein. Zu welcher dieser Gruppen der anfragende Nutzer auch gehört, sollten hier aufgrund von gleichen Ähnlichkeitswerten zufällige Aussiebungen stattfinden.

4.3.3 Daten

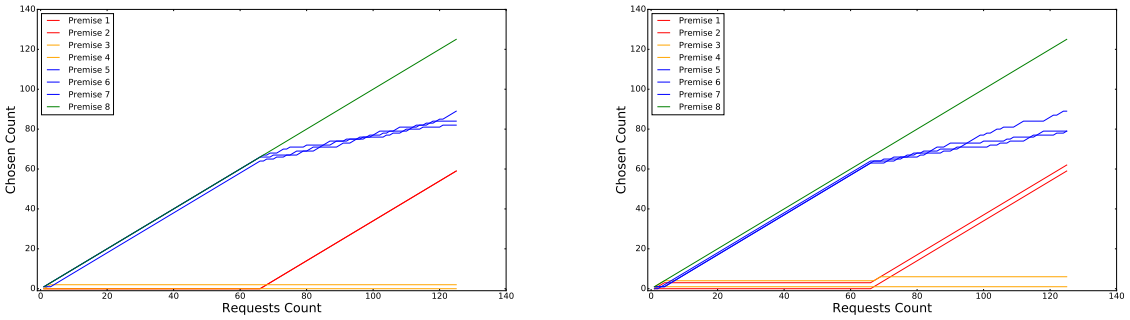
Im Folgenden werden die Daten zur Evaluation des Algorithmus bezüglich der Meinung der Nutzer in Form von Graphen präsentiert. Wie auch im Algorithmus bezüglich des Positionsinteresses wurden die Daten von Nutzer 8 verwendet. Dabei ist die Anzahl der Meinungsbekundungen gegen die Anzahl, wie oft eine Prämisse vorgeschlagen wurde, aufgetragen. Zur besseren Illustration wurden auch hier unterschiedliche Farben verwendet. Die Nutzer 1 und 2 sind aus Gruppe 1 und ändern ihre Meinung nie und werden deshalb mit der Farbe rot gekennzeichnet. Die Nutzer 3 und 4 sind auch aus Gruppe 1, ändern ihre Meinung jedoch und erhalten deswegen die Farbe gelb. Die Nutzer 5, 6 und 7 snid aus Gruppe 2 und ändern ihre Meinung nie und sind deshalb als blau gekennzeichnet, während Nutzer 8 aus Gruppe 2 ist und seine Meinung ändert und die Farbe grün erhält, da ihm die Vorschläge unterbreitet werden.



(a) Durchlauf 1

(b) Durchlauf 2

Abbildung 4.4: Verlauf der Vorschläge von Prämissen, Durchläufe 1 und 2



(a) Durchlauf 3

(b) Durchlauf 4

Abbildung 4.5: Verlauf der Vorschläge von Prämissen, Durchläufe 3 und 4

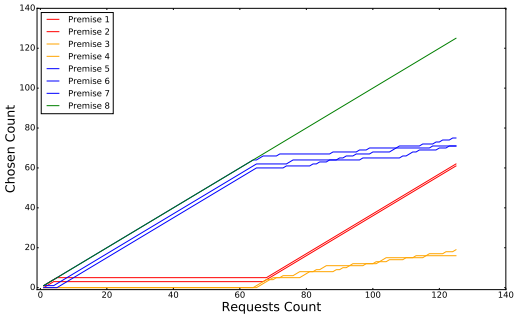


Abbildung 4.6: Verlauf der Vorschläge von Prämissen, Durchlauf 5

4.3.4 Analyse und Auswertung

Durchlauf 1

Der Graph zeigt genau das, was erwartet wurde. Die beiden Gruppen haben keinerlei Schnittmenge an Statements und so werden die Nutzer der jeweils anderen Gruppe automatisch ausgesiebt.

Durchlauf 2

Durchlauf 2 zeigt ein ähnliches Verhalten wie Durchlauf 1. Allerdings wird aufgrund der gemeinsamen Statements 4 und 5 anfangs noch Statement 4 statt Statement 5 vorgeschlagen, bis Nutzer 8 auch zu Statement 5 seine Meinung abgegeben hat.

Durchlauf 3

Aber erst ab Durchlauf 3 zeigen die Meinungsänderungen von Nutzer 1, 2 und 8 ihre Wirkung. Ab

68 Anfragen werden plötzlich die Prämissen von Nutzer 1 und 2 vorgeschlagen. Diese Nutzer sind mit Nutzer 8 die einzigen Nutzer, die ihren zugeordneten Statements noch zustimmen. Da dies nur drei Prämissen sind, wird eine vierte von einem Nutzer aus Gruppe 2 zufällig ausgewählt, leicht am Verhalten der blauen Linien zu erkennen. Diese werden aus Gruppe 2 gewählt, weil Nutzer 8 die größere Schnittmenge an Statements mit dieser Gruppe bildet (er gehört zu dieser Gruppe, hat nur seine Meinung geändert).

Durchlauf 4
Durchlauf 4 zeigt das gleiche Verhalten wie Durchlauf 3, wenn man von der Verschiebung der Prämissen 1 und 2 absieht, welche einfach dadurch erklärbar ist, dass eine Meinungsänderung erst dann sichtbar wird, wenn der Nutzer seine Meinung zu allen Statements abgeben konnte. Insbesondere für Nutzer 8 ist hierbei die Schnittmenge mit den Nutzern 1 und 2 interessant.

Durchlauf 5
Auch Durchlauf 5 zeigt ein erwartetes und leicht erklärbares Verhalten. Hier werden ab der Hälfte auch die Prämissen 3 und 4 vorgeschlagen. Dies geschieht, weil die Schnittmenge von bewerteten Statements von Nutzer 8 mit allen anderen Nutzern identisch ist. Jeder Nutzer bewertet jedes Statement. Daher wird aus den fünf Nutzern, die nun nicht der gleichen Meinung sind wie Nutzer 8 ein Nutzer ausgewählt, dessen Prämisse vorgeschlagen wird.

Außerdem ist erkennbar, dass über alle Durchläufe hinweg, der Vorschlag von Nutzer 8 selbst nie ausgesiebt wurde. Nutzer 8 hat mit sich selbst natürlich die größte Ähnlichkeit, was ein Aussieben seiner Prämisse unmöglich macht. Dieses Verhalten ist so gewünscht.

4.4 Fazit

Das Verhalten des programmierten Recommender Systems entspricht den Erwartungen. Diese Evlaution ist natürlich nur eine qualitative, aber sie reicht aus, um zu erkennen, dass es möglich ist ein Recommender System für die Diskussionsplattform D-BAS zu programmieren. Das programmierte Recommender System löst die Problemstellungen der Bacheloarbeit hinreichend, kann jedoch mit einer groß gefüllten Datenbank, einer quantitativen Auswertung von Nutzermeinungen sowie unter unterschiedlichen Einstellungen noch genauer evaluiert werden. Diese Dinge standen zur Zeit der Arbeit am Programm nicht zur Verfügung und würden auch den Rahmen dieser Bachelorarbeit sprengen.

Kapitel 5

Zusammenfassung und Ausblick

5.1 Zusammenfassung

In dieser Arbeit wurde ein Recommender System in Form eines Microservices für die dialogbasierte Diskussionsplattorm D-BAS entwickelt. Wie es für Microservices üblich ist, verwaltet auch dieser seine eigene Datenbank. Das Recommender System löst dabei drei Aufgaben: Es schlägt je nach Anfrage fünf Positionen für ein Thema, fünf Prämissen für die Meinung zu einer Aussage oder fünf Prämissen für einen Angriff auf eine Aussage oder ein Argument vor. Dabei wird das Collaborative Filtering verwendet, bei dem der anfragende Nutzer mit allen anderen Nutzern verglichen wird und solche Objekte vorgeschlagen werden, die von Nutzern verwendet wurden, deren Verhalten dem Verhalten des anfragenden Nutzers am ehesten ähneln. Dieses Verhalten wird in dieser Arbeit unterteilt in drei Kategorien: Die Meinung zu einer Aussage oder einem Argument, das Interesse an einer Position und das Interesse an einem Thema. Letzteres wird für die erste Aufgabe verwendet während ersteres für die anderen beiden Aufgaben eingesetzt wird. Das Interesse an einem Thema dient allen drei Aufgaben als zusätzliche Möglichkeit, Objekte auszusieben, bevor eine zufällige Auswahl getroffen werden muss. Die Evaluation hat gezeigt, dass das entwickelte Programm die Aufgaben hinreichend bewältigen kann. Es zeigt das erwartete und gewünschte Verhalten. Für jedes abweichende Verhalten konnte eine mögliche Ursache innerhalb des Algorithmus, der die Nutzer miteinander vergleicht, gefunden werden.

5.2 Ausblick

Da diese Arbeit nur darauf abzielte, die Frage zu beantworten, ob es möglich ist ein Recommender System für D-BAS zu entwickeln, ist das entwickelte Programm nicht für D-BAS optimiert. Es können viele Dinge verbessert und verändert werden. Andere Vergleichsalgorithmen könnten für D-BAS passender sein. Was passiert, wenn der Microservice ausfällt und seine Datenbank nicht mehr aktualisieren kann? D-BAS hat seine eigene Datenbank. Hier könnte ein Teilservice beim Starten des Recommender System beide Datenbanken überprüfen und die eigene Datenbank aktualisieren. Optimale Einstellungen der Algorithmen im Bezug auf Notwendigkeit und Leistung sollten gefunden werden. Ein Content-Based Filtering basiertes Recommender System oder ein Hybrid aus Content-Based Filtering und Collaborative Filtering wären denkbar. Hierfür müssten die Einträge der Nutzer einer Textanalyse unterzogen werden. Hier könnte man zum Beispiel mit Tags arbeiten. Für den Umfang und das Ziel dieser Bachelorarbeit wurde der schnellste und einfachste Weg gewählt. Die Schnittstelle zwischen Recommender System und D-BAS könnte ebenfalls optimiert werden, vor allem im Bezug auf Sicherheit. Mit der Weiterentwicklung von D-BAS selbst wird auch eine stetige Anpassung des Recommender Systems nötig sein. Eine Unterscheidung zwischen wirklicher Meinung und bloßer Neugier eines Nutzers zum Beispiel ist dringend erforderlich.

Literaturverzeichnis

[Ama] *Amazon*. https://www.amazon.com/, . Zuletzt aufgerufen: 28.03.2017

[BOHG13] BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; GUTIÉRREZ, A.: In: *Knowledge-Based Systems* (2013).

[DDE+15] DAYA, Shahir; DUY, Nguyen V.; EAT, Kameswara; FERREIRA, Carlos M.; GLOZIC, Dejan; GUCER, Vasfi; GUPTA, Manav; JOSHI, Sunil; LAMPKIN, Valerie; MARTINS, Marcelo; NARAIN, Shishir; VENNAM, Ramratan: *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach*. IBM International Technical Support Organization, 2015. Zuletzt aktualisiert: 2016

[eBa] *eBay*. http://www.ebay.com/, . Zuletzt aufgerufen: 28.03.2017

[EWe10] *Entwicklung des Webs*. http://www.informatik.uni-oldenburg.de/~iug10/sn/html/content/ewb.html, 2010. Zuletzt aufgerufen: 28.03.2017

[Goo] *Google*. http://www.google.com/, . Zuletzt aufgerufen: 28.03.2017

[Jav] *Informationen zu Java 8*. https://www.java.com/de/download/faq/java8.xml, . Zuletzt aufgerufen: 28.03.2017

[KBBM16] KRAUTHOFF, Tobias; BETZ, Gregor; BAURMANN, Michael; MAUVE, Martin: Dialog-Based Online Argumentation. In: *Computational Models of Argument*, Potsdam, 2016, S. 33–40.

[LHM+13] LIU, Haifeng; HU, Zheng; MIAN, Ahmad; TIAN, Hui; ZHU, Xuzhen: In: *Knowledge-Based Systems* (2013).

[LWM+14] LU, Jie; WU, Dianshuang; MAO, Mingsong; WANG, Wei; ZHANG, Guangquan: In: *Decision Support Systems* (2014).

[Mat] *Matplotlib*. `http://matplotlib.org/`, . Zuletzt aufgerufen: 28.03.2017

[Mil56] MILLER, George A.: In: *Psychological Review* 63 (1956), 81–97 S.

[Num] *Numpy*. `http://www.numpy.org/`, . Zuletzt aufgerufen: 28.03.2017

[SPA] *Spark Framework - A tiny Java web framework*. `http://sparkjava.com/`, . Zuletzt aufgerufen: 28.03.2017

[SW10] SAMMUT, Claude (Hrsg.); WEBB, Geoffrey I. (Hrsg.): *Encyclopedia of Machine Learning*. Springer US, 2010

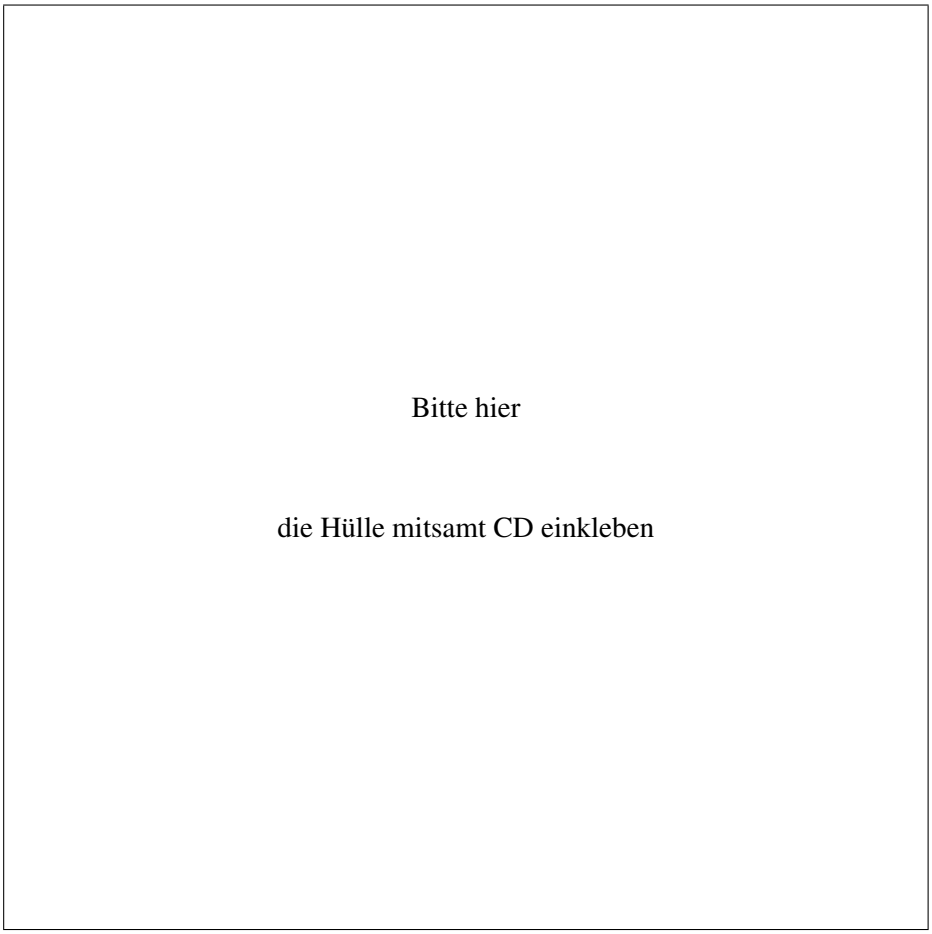
[Web14] *Web 2.0 ist Vergangenheit, jetzt kommt das Web 3.0*. `https://www.upon-onlinemarketing.de/web-2-0-ist-vergangenheit-jetzt-kommt-das-web-3-0/`, 2014. Zuletzt aufgerufen: 28.03.2017

[You] *Youtube*. `http://www.youtube.com/`, . Zuletzt aufgerufen: 28.03.2017

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, 03.April 2017 Stefan Genenger



Diese CD enthält:

- Eine *pdf* Version der Bachelorarbeit
- Alle LaTeXund Grafik Dateien mitsamt dazugehörigen Skripten, die verwendet wurden
- Der Quellcode, der während der Bachelorarbeit erarbeitet wurde
- Alle während der Evaluation angefallenen Messdaten
- Alle referenzierten Webseiten und wissenschaftlichen Arbeiten