Entwicklung eines Social Collaboration Analytics Dashboard-Prototyps für Beiträge von UniConnect

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science
im Studiengang Informatik

Vorgelegt von

Stefan Hermann Strüder

Immatrikulationsnummer: 214200670
E-Mail: stefanstrueder@uni-koblenz.de

Fachbereich 4: Informatik

Institut für Wirtschafts- und Verwaltungsinformatik

Universität Koblenz-Landau

Betreuer:

Prof. Dr. Petra Schubert

Florian Schwade

Koblenz, September 2018

Erklärung

Ich versichere,

dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Stefan Hermann Strüder

Koblenz, September 2018

Abstract

Seit der vergangenen Dekade steigt die Nutzung von sogenannten Enterprise Collaboration Systems (ECS) in Unternehmen. Diese versprechen sich mit der Einführung eines solchen zur Gattung der Social Software gehörenden Kollaborationssystems, die menschliche Kommunikation und Kooperation der eigenen Mitarbeiter zu verbessern. Durch die Integration von Funktionen, wie sie aus Social Media bekannt sind, entstehen große Mengen an Daten. Darunter befinden sich zu einem erheblichen Teil textuelle Daten, die beispielsweise mit Funktionen wie Blogs, Foren, Statusaktualisierungen oder Wikis erstellt wurden. Diese in unstrukturierter Form vorliegenden Daten bieten ein großes Potenzial zur Analyse und Auswertung mittels Methoden des Text Mining. Die Forschung belegt dazu jedoch, dass Umsetzungen dieser Art momentan nicht gebräuchlich sind. Aus diesem Grund widmet sich die vorliegende Arbeit diesem Mangel. Ziel ist die Erstellung eines Dashboard-Prototyps, der sich im Rahmen von Social Collaboration Analytics (SCA) mit der Auswertung von textuellen Daten befasst. Analyseziel ist die Identifikation von populären Themen, die innerhalb von Communities oder communityübergreifend von den Plattformnutzern in den von ihnen erstellten Beiträgen aufgegriffen werden. Als Datenquelle wurde das auf IBM Connections aufbauende ECS UniConnect ausgewählt. Dieses wird vom University Competence Center for Collaboration Technologies (UCT) an der Universität Koblenz-Landau betrieben. Grundlegend für die korrekte Funktionsweise des Dashboards sind mehrere Java-Klassen, deren Umsetzungen auf verschiedenen Methoden des Text Mining basieren. Vermittelt werden die Analyseergebnisse im Dashboard durch verschiedene Diagrammarten, Wordclouds und Tabellen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhangsverzeichnis

Abkürzungsverzeichnis

BAS Betriebliche Anwendungssysteme

CSCW Computer Supported Cooperative Work

DSR Design Science Research

ECS Enterprise Collaboration System

ERP Enterprise Resource Planning

ESN Enterprise Social Network

ESS Enterprise Social Software

IDF Inverse Document Frequency

IE Information Extraction

JSF JavaServer Faces

LDA Latent Dirichlet Allocation

ML Machine Learning

NLP Natural Language Processing

RO Forschungsziel (Research Objective)

RQ Forschungsfrage (Research Question)

SCA Social Collaboration Analytics

SQL Structured Query Language

TF Term Frequency

UCT University Competence Center for Collaboration Technologies

UUID Universally Unique Identifier

1 Einleitung

Die Verwendung von Social Media und insbesondere deren sozialen Features stieg in den vergangenen Jahren im privaten Umfeld rapide an und führte dazu, dass die auf Web 2.0 Technologien basierenden und im Bereich der Social Software einzuordnenden Kommunikationsplattformen die Aufmerksamkeit der Entwickler von Unternehmenssoftware erregten (Schubert & Glitsch, 2015; Schwade & Schubert, 2017). Social Software bezeichnet Softwaresysteme, welche die menschliche Kommunikation und Kollaboration unterstützen, also jene Eckpfeiler des Konzepts der Computer Supported Cooperative Work (CSCW), für die ein besonders hohes Bedürfnis zur Verbesserung innerhalb von Unternehmen besteht (Bächle, 2006; Wehner, Falk, & Leist, 2017).

Dieses gesteigerte Verlangen nach einer für Unternehmen optimierten Lösung führte zur Entwicklung von Enterprise Social Software (ESS) (Schwade & Schubert, 2017). Diese bietet unter anderem die Möglichkeit zur Erstellung von Blogs, Foren und Wikis sowie die Bereitstellung von Funktionen zum Nachrichten- und Datenaustausch. Gemeinsam arbeiten die Mitarbeiter der Unternehmen in ESS selbstorganisiert und räumlich verteilt in virtuellen Gemeinschaften, sogenannte Communities (Bächle, 2006). Hinzu kommt die Verbreitung von sogenannten Enterprise Social Networks (ESN), also unternehmensinternen sozialen Netzwerken, die die Möglichkeit zur Erstellung von sozialen Profilen und den Aufbau von Verbindungen zwischen Nutzern unterstützen (Viol & Hess, 2016).

Eine Verknüpfung der Funktionen von ESS und ESN bilden Enterprise Collaboration Systems (ECS). Zusätzlich beinhalten diese Funktionen, die aus dem Bereich der klassischen Groupware stammen (Schubert & Williams, 2013). Darunter fallen beispielsweise Gruppenkalender. Eine solche integrierte Lösung (siehe Kapitel 2) bietet unter anderem IBM, der Marktführer im Segment der ECS, mit IBM Connections an (Thompson, 2015). Unternehmen, die sich für die Einführung von ECS entscheiden, erhoffen sich unter anderem die folgenden sechs Verbesserungen wahrnehmen zu können (übersetzt nach Richter, Stocker, Müller, & Avram, 2011):

- Effiziente und zielorienterte Kommunikation zwischen Mitarbeitern und Vermeidung von Informationsüberflutung
- Effizienter Wissensaustausch
- Einrichtung von Expertennetzwerken
- Beteiligung von Mitarbeitern und Schaffung einer offenen Unternehmenskultur
- Erhöhte Awareness und Transparenz
- Förderung des Innovationspotienzials und Sicherung der Zukunftsfähigkeit des Unternehmens

Diese sechs Beispiele stellen nur einen kleinen stellvertretenden Teil der mit der Einführung eines ECS einhergehenden Verbesserungen dar.

Anzumerken ist, dass die Verwendung von ECS in vielen Unternehmen nicht vorgeschrieben ist und in vielen Fällen keine genauen Vorgaben und Anweisungen zur einheitlichen Verwendung dieser existieren (Diehl, Kuettner, & Schubert, 2013; Williams & Schubert, 2015). Dies unterscheidet ECS von ERP Systemen (Enterprise Resource Planning), bei welchen die Verfahrensweisen für die Nutzer klar definiert und fest vorgegeben sind (Williams, 2013). Die Mitarbeiter handeln bei der Verwendung von ECS somit gemäß ihrer eigenen interpretativen Flexibilität, die mehrere Möglichkeiten zur Verwendung der Systeme zulässt.

Studien belegen, dass das Marktwachstum von Kollaborationssystemen stetig steigt (Thompson, 2015). Während im Jahr 2010 der Umsatz etwa 570 Millionen US-Dollar betrug, stieg er im Jahr 2015 bereits auf rund 1.7 Milliarden US-Dollar an (Thompson, 2015). Für das Jahr 2019 wird ein Umsatz von 3.5 Milliarden US-Dollar erwartet (Thompson, 2015). Diese Zahlen sprechen analog für eine Zunahme der Verwendung von ECS im unternehmerischen Umfeld und belegen somit deren Relevanz.

Aus der Verwendung von ECS in einem Unternehmen folgt zudem ein erhöhtes Aufkommen von Daten. Dies umfasst neben Transaktionsdaten, wie beispielsweise Loggingeinträge der Nutzeraktivitäten auf der Plattform, auch, und vor allem im großen Umfang, textuelle Daten in Form der mit den Funktionen zum Wissens- und Informationsaustausch erzeugten Beiträge der Plattformnutzer. Gespeichert werden diese Daten in Datenbänken und bieten ein großes Potenzial zur Analyse und Auswertung (Schwade & Schubert, 2017). Durchgeführt wird dies im Rahmen von Data Analytics. Schwade und Schubert prägten im Bezug der Betrachtung von ECS den Begriff Social Collaboration Analytics (SCA) für diese Aufgabe (Schwade & Schubert, 2018). Gleichzeitig merken sie an, dass sich SCA vorwiegend der Analyse der Benutzung von Kollaborationsplattformen widmet (Schwade & Schubert, 2018). Dabei dienen die Transaktionsdaten als Datengrundlage. Eine Analyse der textuellen Daten findet selten statt, obwohl die große Menge an verfügbaren Daten eine aussagekräftige Auswertung zulassen würde.

Eine solche Analyse ist somit lukrativ, jedoch auch mit enormen Aufwand verbunden, der allein durch menschliche Kraft nicht aufwendbar ist (Hippner & Rentzmann, 2006). In diesem Fall ist eine rechnergestützte automatische Analyse notwendig, die mithilfe von Prozessen des Text Mining ausgeführt werden kann (Debortoli et al., 2016). Text Mining ist eine Form des Data Mining mit der Spezialisierung auf die Analyse natürlicher Sprache und erlaubt es, automatisch implizites, vorher unbekanntes und möglicherweise nützliches Wissen aus großen Mengen an unstrukturierten Daten zu extrahieren (Debortoli et al., 2016). Im Fall von ECS wäre es mit der Hilfe von Methoden des Text Mining möglich, den Inhalt der Beiträge der Nutzer zu analysieren und beispielsweise aufzuzeigen, welche Begriffe und Themen besonders häufig auf der Platform Erwähnung finden. Im Umfeld des Text Mining existieren diverse Methoden, mit denen eine solche Analyse durchführbar ist.

Durch die hohe Anzahl an textuellen Daten und der steigenden Verwendung von ECS in Unternehmen, stellen ECS einen besonders relevanten Anknüpfungspunkt zur textuellen Analyse mittels Text Mining Methoden dar. Die Extraktion zuvor unbekannten Wissens kann eine wichtige Informationsquelle für Unternehmen werden.

1.1 Forschungsziele und Forschungsfragen

Übergeordnetes Ziel dieser Arbeit ist es, dem angesprochenen Mangel an Analysemöglichkeiten der textuellen Daten im Rahmen der SCA entgegenzukommen. Dazu soll exemplarisch ein Dashboard-Prototyp für Statistiken über jene Daten der Kollaborationsplattform UniConnect erstellt werden. Dieses ECS ist eine Entwicklung des University Competence Center for Collaboration Technologies (UCT) und basiert auf dem bereits in der Motivation erwähnten IBM Connections. Das UCT ist unterdessen ein gemeinsames Projekt der Universität Koblenz-Landau, der IBM Deutschland GmbH und der GIS GmbH. Wichtiges Arbeitsmittel im Verlauf der Entwicklung des Dashboards ist die Erstellung von Java-Klassen, welche Text Mining Methoden implementieren. Visualisiert werden die Analyseergebnisse durch gängige Darstellungsformen wie Diagramme, Tabellen und Wordclouds.

Grundlegend für die Entwicklung der Java-Klassen sind die Beschaffung und die Aufbereitung der vorhandenen Daten aus den UniConnect-Datenbanken. Hieraus ergibt sich folglich das erste Forschungsziel:

> **RO1:** Entwickeln einer Methode zum Abruf und zur Aufbereitung der Daten aus den Datenbanken.

Daran anknüpfend leitet sich die zugehörige Forschungsfrage ab:

> **RQ1:** Mit welchen Mitteln können die aus den Datenbanken abgerufenen Daten möglichst weit aufbereitet werden?

Die Beantwortung dieser Frage wird auf einen wichtigen Schritt im Rahmen des Text Mining verweisen, dem sogenannten Preprocessing (siehe Kapitel 3.2). Die daraus resultierenden aufbereiteten Daten sind dann bereit zur Verwendung. Dies führt zum zweiten Forschungsziel:

> **RO2:** Erstellung von Java-Klassen zur Analyse, Auswertung und Visualisierung der aufbereiteten Daten unter Verwendung von Text Mining Methoden.

Aus den Vorüberlegungen zum Erreichen des zweiten Forschungsziels ergeben sich die folgenden zugehörigen Forschungsfragen:

> **RQ2a:** Welche Algorithmen kommen für die Analyse der Daten in Frage?
>
> **RQ2b:** Welche Algorithmen eigenen sich am besten, um Statistiken über die Daten und deren Visualisierung zu erstellen?

Mit dem Abschluss des zweiten Forschungsziels folgt die Erstellung einer grafischen Benutzeroberfläche (GUI) in Form eines Dashboards, dessen Entwicklung das dritte Forschungsziel darstellt:

> **RO3:** Modellierung und prototypische Implementierung eines Dashboards mit anschließender Evaluation.

1.2 Forschungsdesign

Das in dieser Arbeit angewendete Forschungsdesign basiert auf der Forschungsmethode Design Science Research (DSR) nach Vaishnavi und Kuechler (2012). Darauf aufsetzend wird zudem das Prozessmodell CRISP-DM (Cross-Industry Standard Process for Data Mining, Chapman et al., 2000) als Vorlage für die weiteren Arbeitsphasen dieser Arbeit verwendet. Da der Großteil dieser Arbeit zum Erreichen der Forschungsziele durch Programmierung erfolgt, bietet das CRISP-DM Prozessmodell ein passendes Vorgehen, da es speziell für die Erarbeitung von Text Mining Projekten (als Teilmenge des Data Mining) entwickelt wurde. Im Folgenden werden die einzelnen Arbeitsphasen näher vorgestellt und ein Bezug zu den Schritten des DSR hergestellt. Eine Visualisierung dessen ist in Abbildung 1.1 dargestellt.

Die ersten beiden Arbeitsphasen sind Business Understanding und Data Understanding. Ziel beider Phasen ist die Einarbeitung in das Thema der Arbeit durch eine strukturierte Literaturanalyse. Weiterhin stehen die Formulierung der Forschungsziele und Forschungsfragen sowie im Rahmen des Data Understanding das Verschaffen eines Überblicks über die Datenbanken der UniConnect Plattform im Mittelpunkt. Dies umfasst auch die Entwicklung der SQL-Abfragen, die im späteren Verlauf benötigt werden. Mit der Vollendigung dieser Aufgaben, kann das erste Forschungsziel abgeschlossen werden. Der zu diesen Phasen ähnliche Forschungsschritt des DSR ist Awareness of Problem. Die darauffolgende Phase Data Preparation sieht den Abruf und die für die weitere Verwendung wichtige Aufbereitung der Daten vor. Die Erstellung der Text Mining Java-Klassen sowie die Modellierung des Dashboard-Prototyps umfasst die dritte Phase Modeling, welche analog im DSR Development genannt wird. Ein weiteres Ziel dieser Phase ist das Erreichen des zweiten Forschungsziels sowie die teilweise Erarbeitung des dritten Forschungsziels. Die vierte Arbeitsphase umfasst die Evaluation des Dashboards, deren Ziel es ist, eventuell auftretende Fehler bei der Verwendung zu erkennen und zu beheben. Damit einher geht der Abschluss des dritten Forschungsziels. Sowohl das CRISP-DM Prozessmodell als auch der DSR sehen diesen Schritt vor. Die abschließende Arbeitsphase Deployment dient der Nachbereitung der Arbeit und umfasst die Inbetriebnahme des Prototyps. Analog lautet der zugehörige Forschungsschritt des DSR Conclusion.

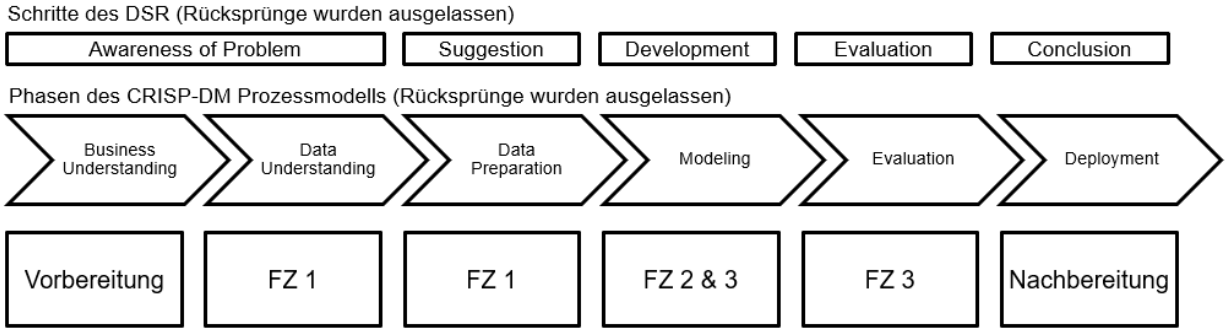

Abb. 1.1: Gegenüberstellung der Forschungsmethode DSR und des Prozessmodells CRISP-DM mit Zuordnung zu den Forschungszielen (Chapman et al., 2000; Vaishnavi & Kuechler, 2012)

1.3 Aufbau der Arbeit

Die Arbeit ist in sechs Kapitel gegliedert, welche an die aus dem CRISP-DM Prozessmodell stammenden Arbeitsphasen angeknüpft sind.

Das erste Kapitel beginnt mit einer einleitenden Motivation und umfasst zudem die Erklärung der Forschungsziele, der zugehörigen Forschungsfragen sowie eine Vorstellung des Forschungsdesigns. Im zweiten Kapitel werden die für den weiteren Verlauf dieser Arbeit wichtigen Inhalte im Bezug zu Enterprise Collaboration Systems eingeführt. Dazu zählt unter anderem die nähere Betrachtung der Datengewinnung und -speicherung der für diese Arbeit ausgewählten Plattform UniConnect. Das darauffolgende dritte Kapitel vermittelt die Grundlagen des Text Mining und die Anwendungsgebiete dessen. Ein Bezug zur Phase Data Preparation wird durch die Erläuterung der grundlegenden Schritte des Preprocessings aufgebaut. Kapitel 4 und 5 dienen der Erläuterung der für diese Arbeit geleisteten Programmierarbeit. Hierzu werden zunächst die auf Methoden des Text Mining basierenden Java-Klassen in Kapitel 4 näher beschrieben. Das auf diesen Klassen basierende Dashboard wird dann in Kapitel 5, inklusive der Erläuterung der zugrundeliegenden Technologie, erläutert. Das abschließende sechste Kapitel beinhaltet die Evaluation des Dashboards, eine Zusammenfassung und Erläuterung der Erkenntnisse der Arbeit sowie einen Ausblick auf eventuelle weitere Arbeitsschritte.

Eine Zuordnung der Arbeitsphasen des CRISP-DM Prozessmodells zu den jeweiligen Kapiteln der Arbeit ist in Abbildung 1.2 dargestellt.

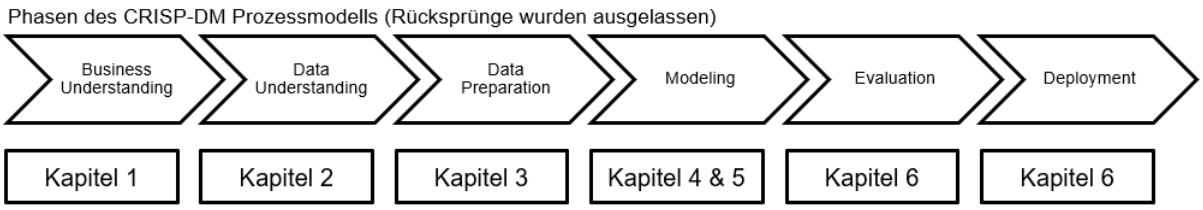

Abb. 1.2: Zuordnung der Kapitel zu den Arbeitsphasen (Chapman et al., 2000)

2 Grundlagen

Dieses Kapitel dient dem Aneignen von Grundwissen, das für das Verständnis des weiteren Verlaufs der Arbeit benötigt wird. Begonnen wird im ersten Kapitel 2.1 mit einer Einführung in Enterprise Collaboration Systems. Kapitel 2.2 gibt einen Einblick in das ECS UniConnect. Das Thema Social Collaboration Analytics wird anschließend in Kapitel 2.3 näher erläutert. Ein Bezug zur Arbeitsphase Data Understanding wird durch die Betrachtung der Datenbanken und der Datengewinnung und Datenspeicherung im Zusammenhang dieser aufgebaut.

2.1 Enterprise Collaboration Systems

Ziel dieses Kapitels ist es, den Leser mit den Grundlagen der auf Enterprise Collaboration Management spezialisierten ECS vertraut zu machen. Dazu werden zunächst Begriffsabgrenzungen zu Softwares mit ähnlicher Konzeption beschrieben, ehe im Anschluss ECS allgemein definiert werden.

2.1.1 Begriffsabgrenzungen

Enterprise Collaboration Systems gehören, entgegen gängiger Bezeichnungen im allgemeinen Sprachgebrauch, nicht zur Gattung der Social Media – beide stehen jedoch über die Gattung Social Software in Verbindung (Schubert & Williams, 2013). Social Software beschreibt indes Softwaresysteme, welche zur Unterstützung der menschlichen Kommunikation, Interaktion und Zusammenarbeit dienen (Sixtus, 2005). Eine alternative Definition beschreibt Social Software als Software, die einen Mehrwert aus menschlichem Sozialverhalten unterstützt, erweitert oder ableitet (Richter & Koch, 2007). Auszeichnende Funktionen sind beispielsweise das Anlegen von Nutzerprofilen und die Möglichkeiten zum Erstellen, Folgen und „Liken“ von textuellen Inhalten (Schwade & Schubert, 2017).

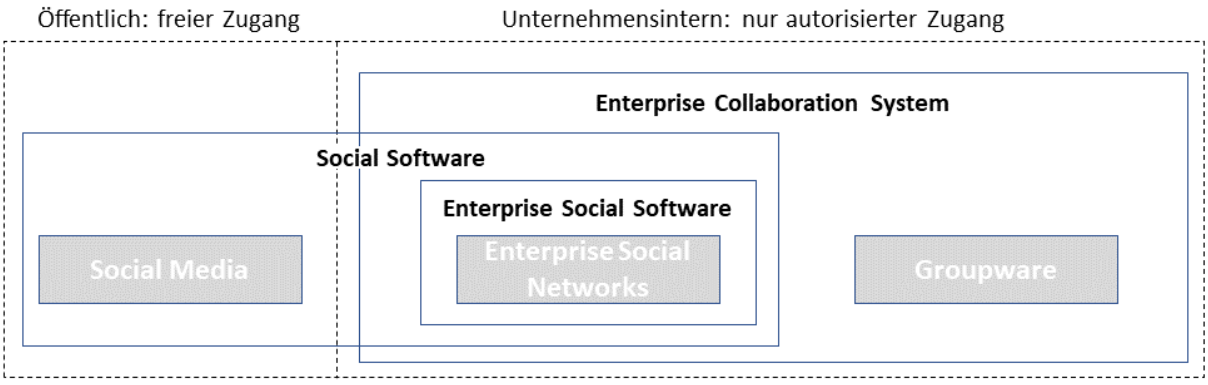

Abb. 2.1: Begriffsabgrenzungen im gattungsspezifischen Umfeld von ECS (Schwade & Schubert, 2017)

Wie Abbildung 2.1 verdeutlicht, unterscheiden sich Social Media und ECS zudem durch ihre Art des Zugangs. Social Media Plattformen, zu denen Facebook und Twitter zählen, sind öffentliche Plattformen, zu denen grundsätzlich jede Person Zugang hat. ECS sind hingegen geschlossene Plattformen und weisen nur einen beschränkten Zugang auf, der allein den Mitarbeitern des Betreiberunternehmens vorbehalten ist (Schubert & Williams, 2013).

Eine genaue Beschreibung der in Abbildung 2.1 erkennbaren Bestandteile von ECS folgt im nächsten Kapitel.

2.1.2 Definition

ECS gehören zur Gattung der betrieblichen Anwendungssysteme (BAS, siehe Abbildung 2.2) und haben das Ziel, als sozio-technisches System die Zusammenarbeit in Unternehmen zu unterstützen. Dieses System umfasst dabei nicht nur die Soft- und Hardware, sondern auch die diesem nahestehenden Personen und Prozesse sowie die betriebliche Organisation (Williams & Schubert, 2015).

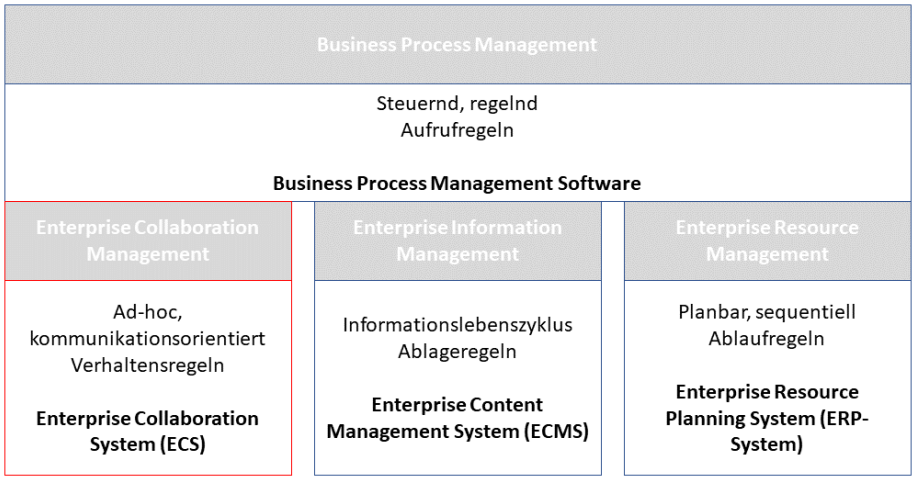

Abb. 2.2: BAS-Anwendungsbereiche mit ihren Softwaregattungen (Schubert & Winkelmann, 2016)

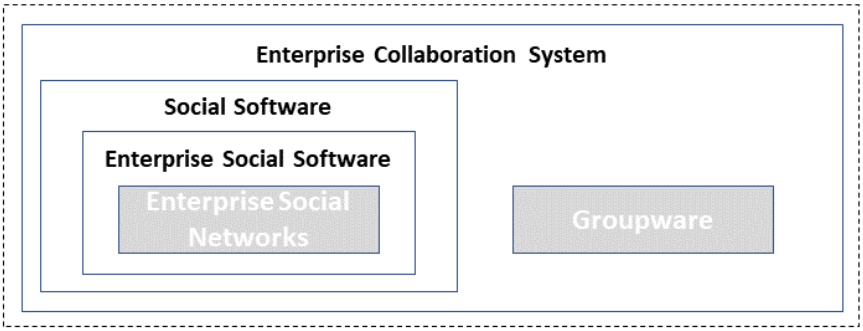

Abb. 2.3: Bestandteile von ECS (Schwade & Schubert, 2017)

Wie in Abbildung 2.3 zu sehen ist, bilden ECS eine Verknüpfung von Komponenten des Typs der Social Software und der klassischen Groupware. Unter den Begriff von Groupware fallen Anwendungen mit dem Ziel, Gruppen von Nutzern bei der Kommunikation, Kollaboration und Koordination ihrer Aktivitäten zu unterstützen (Koch & Gross, 2006). Gängig im Umgang mit Groupware ist das Arbeiten in Organisationsstrukturen in Form von Communities und der Vergabe von Rollen und Verantwortlichkeiten an die Mitglieder (Alarcon, Guerrero, & Pino, 2005). Weitere Artefakte sind beispielsweise Gruppenkalender, Newsgroups und Chats sowie gemeinsam genutzte Dateiablagen (Williams & Schubert, 2015).

Die weiteren Komponenten von ECS, die dem Typ der Social Software entstammen, sind ESS und ESN. ESS implementieren dabei aus Social Media bekannte Features, die für den unternehmerischen Kontext angepasst sind. Dazu zählen soziale Profile, Microblogs, Blogs, Wikis und Methoden zum Datenaustausch (Schwade & Schubert, 2017). ESN implementieren ebenfalls soziale Profile sowie Methoden zur Interaktion und Erstellung von Beziehungen zwischen den Nutzern der Plattformen (Riemer, Stieglitz, & Meske, 2015). Es entsteht somit ein unternehmensinternes soziales Netzwerk (Viol & Hess, 2016).

ECS werden durch zwei verschiedene Ausprägungen unterschieden. Werden unterschiedliche Funktionalitäten als eigenständige Tools von verschiedenen Anbietern verwendet, so handelt es sich bei diesem ECS um ein sogenanntes Portfolio (Williams & Schubert, 2015). Die zweite Ausprägung trifft auf das in dieser Arbeit betrachtete ECS UniConnect zu und wird im zugehörigen Kapitel 2.2.1 erläutert.

2.2 UniConnect

Das ECS UniConnect liefert mit seinen textuellen Daten die Datengrundlage für die im Verlauf dieser Arbeit erstellten Java-Klassen zur Analyse und Auswertung. Eine Einführung des auf IBM Connections basierenden ECS erfolgt in diesem Kapitel. Neben grundlegenden Informationen werden auch die für diese Arbeit wichtigen Funktionen der Plattform sowie die Datenspeicherung und Datengewinnung im Bezug zur Plattform erläutert.

2.2.1 Grundlagen

UniConnect ist ein auf IBM Connections basierendes integriertes ECS. Dies bedeutet, dass es als sogenannte Suite eine Vielzahl an verschiedenen Funktionalitäten und Tools in einem einzigen Produkt vereint (Williams & Schubert, 2015). Entwickelt wurde es vom University Competence Center for Collaboration Technologies (UCT) als akademische Kollaborationsplattform (Schubert & Williams, 2016). Das UCT ist ein Projekt, welches in Zusammenarbeit der Forschungsgruppe Betriebliche Anwendungssysteme (FG BAS, unter Führung von Prof. Petra Schubert) der Universität Koblenz-Landau (Campus Koblenz), der IBM Deutschland GmbH und der GIS AG (Gesellschaft für Informationssysteme) unterhalten wird. Hierbei ist die FG BAS für das Hosten der Plattform verantwortlich. Die IBM Deutschland GmbH dient unterdessen als Unterstützer bei der Weiterentwicklung des Projektes und stellt die nötigen Softwarelizenzen sowie finanzielle Unterstützung und Fachwissen im Bereich von ECS zur Verfü-

gung. Für den technischen Betrieb und die Wartung der Plattform ist die GIS AG verantwortlich (Beschreibung des UCT[1]).

Gegenwärtig wird UniConnect europaweit von mehr als 30 Universitäten und Hochschulen als Forschungs- und Lehrplattform genutzt (Stand 2018). Hinzu kommen Forschungsprojekte namhafter Unternehmen, die die Plattform zum Informations- und Datenaustausch verwenden.

2.2.2 Relevante Funktionen

UniConnect bietet eine Vielzahl an Funktionen, die als Lösung an die vier Elemente / Ziele des inneren Kerns des von Susan P. Williams entwickelten 8C-Modells (Williams, 2011) anknüpfen. Der Fokus dieser Arbeit liegt auf jenen Funktionen, die Nutzereingaben in Form von Text verarbeiten und damit die für die Analyse benötigten textuellen Daten erzeugen. Ausgewählt wurden dazu die Funktionen[2] Blogs, Forum, Statusaktualisierungen und Wikis. Diese stehen den Nutzern in allen Communities zur Verfügung. Eine Community ist hierbei, laut Beschreibung des UCT[3], ein virtueller Arbeitsraum, der einen zentralen Punkt für die Zusammenarbeit von Gruppen von Nutzern der Plattform darstellt. Es folgt eine Erläuterung der ausgewählten Funktionen:

Blogs

Blogs sind längere Textinhalte, die in chronologischer Reihenfolge sortiert sind und einem Onlinejournal ähneln. Ihr Informationsgehalt ist in der Regel hoch einzustufen, da sie zum Informationsaustausch zwischen Mitgliederinstitutionen innerhalb von Communities dienen. Ein einzelner Blogeintrag besteht aus einem Titel, optional angehängten Tags und dem Inhalt, der entweder als Rich Text oder HTML-Quelltext hinzugefügt werden kann. Ferner besteht die Möglichkeit, Blogeinträge zu kommentieren oder zu empfehlen. Zur Analyse werden der Inhalt der Einträge sowie die zugehörigen Kommentare berücksichtigt. Es besteht jedoch auch die Möglichkeit, Kommentare nicht zu berücksichtigen.

Forum

Foren dienen als strukturierter Diskussionsort für Themen. Die Kommunikation verläuft hierbei asynchron, sie findet also zeitversetzt statt. Durch die Sortierung der Beiträge nach ihrem Erstellungsdatum entsteht somit der Eindruck einer fortlaufenden Unterhaltung. Innerhalb eines Forums können sogenannte Themen (auch Diskussionsstränge oder Threads genannt) erstellt werden, in denen bestimmte Themen oder Fragen diskutiert und beantwortet werden können. Ähnlich wie bei Blogs, sind bei der Erstellung eines Themas ein Titel und optional Tags zu vergeben. Hinzu kommt der Inhalt als Text.

[1] http://uct.de/vorstellung

[2] Es werden nur die Community-internen Versionen der Funktionen berücksichtigt.

[3] https://uct.de/uniconnect

Auch im Fall von Foren werden sowohl die Inhalte der erstellten Themen, als auch die zugehörigen Antworten zur Analyse verwendet. Die Möglichkeit Antworten auszufiltern besteht ebenfalls.

Statusaktualisierungen

Statusaktualisierungen sind Kurznachrichten in Form von Microblogs. Mit einem Limit von 1000 Zeichen dienen sie dem schnellen Austausch von Neuigkeiten und besitzen aufgrund ihrer eingeschränkten Länge nur eine geringe Komplexität. Im sogenannten Activity Stream einer Community werden die erstellten Statusaktualisierungen, in chronologischer Abfolge geordnet, aufgelistet. Es besteht ebenfalls die Möglichkeit des Kommentierens. Eine Besonderheit beim Erstellen von Statusaktualisierungen ist die Möglichkeit, inline Tags mittels #Tag zu verwenden. Außerdem können Mitglieder einer Community mittels @Mitgliedsname markiert werden, um die Awareness dieser für den Inhalt des Beitrags zu erhöhen (Gutwin & Greenberg, 2002). Die markierten Mitglieder erhalten dann eine Benachrichtigung über das plattformeigene Benachrichtigungssystem und via E-Mail (sofern aktiviert). Zur Analyse dienen ebenfalls die Inhalte der Beiträge sowie die zugehörigen Kommentare, die sich bei Bedarf auch herausfiltern lassen.

Wikis

Wikis dienen zur strukturierten und zentralisierten Kollektion von Informationen. Ähnlich wie bei der Online-Enzyklopädie Wikipedia, können die Mitglieder einer Community gemeinsam Wikieinträge erstellen, bearbeiten und kommentieren. Wikis dienen in der Regel der langfristigen Speicherung und Zurverfügungstellung von Wissen und werden laufend mit aktualisierten Informationen verbessert. Bei der Erstellung eines Wikis muss ein Titel vergeben werden. Optional können Tags hinzugefügt werden. Auch im Fall von Wikis, werden die gekürzten Einträge (siehe kommendes Kapitel) und die zugehörigen Kommentare analysiert. Es besteht ebenfalls eine Möglichkeit zur Filterung der Kommentare.

Es ist zudem anzumerken, dass UniConnect bilingual verwendet wird. Die Existenz von Beiträgen in Deutsch und Englisch stellt sich im weiteren Verlauf der Arbeit als Hürde heraus. Diese Tatsache benötigt einen speziellen Umgang mit den Daten, welcher in Kapitel 4.2 näher beschrieben wird.

2.2.3 Datenspeicherung und Datengewinnung

Die textuellen Daten von UniConnect liegen auf DB2 Datenbanken. DB2 ist ein von IBM entwickeltes kommerzielles Datenbankmanagementsystem, welches in Instanzen organisiert ist. Die Aufteilung in Instanzen bedeutet, dass mehrere unabhängige Datenbanksystemumgebungen auf einem physischen Server oder einer virtuellen Maschine betrieben werden können (Chong, 2002).

Im hier vorliegenden DB2 Datenbanksystem existieren vier Instanzen, darunter eine Instanz, welche die sogenannte Metrics-Datenbank enthält. Diese umfasst sämtliche Daten, denen Logging-Operationen der Plattform zugrunde liegen. Auch diese Daten können zur Analyse verwendet werden (siehe Kapitel 1.1). Relevant für diese Arbeit sind jedoch nur die Inhaltdatenbanken der vier ausgewählten Features sowie die Datenbank, welche Informationen zu den Communities bereitstellt. Eine

Übersicht der Funktionen mit ihren zugehörigen Datenbanken bietet die Abbildung 2.4. Eine nähere Beschreibung, aufgeteilt nach Funktion, folgt im Anschluss der Abbildung.

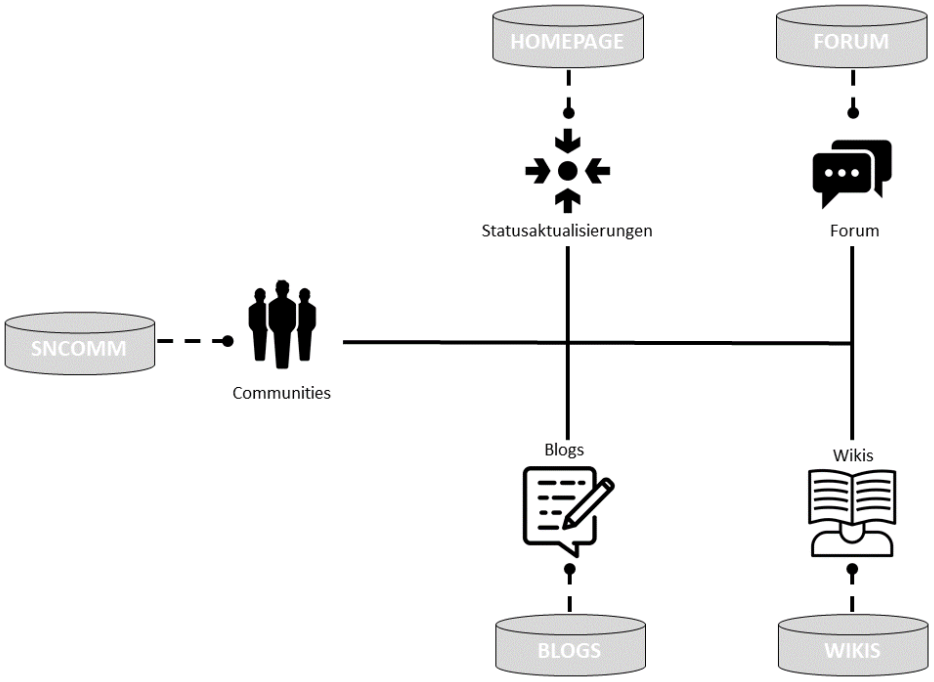

Abb. 2.4: Zuordnung der ausgewählten Funktionen zu ihren Datenbanken (Beschreibung des UCT)

Communities

Jegliche Informationen zu einer Community werden in der Datenbank *sncomm* gespeichert. Wichtige Attribute dieser sind der Communityname und die zugehörige UUID (Universally Unique Unifier) zur eindeutigen Identifizierung. In Form eines Fremdschlüssels ist diese UUID ebenfalls in den Datenbanken der ausgewählten Funktionen wiederzufinden. Dies ermöglicht es, einen Bezug zwischen den verfassten Beiträgen und der zugehörigen Community herzustellen. Die genauen Verbindungen werden jeweils in den folgenden Abschnitten zu den Funktionen näher erläutert. Einen Überblick über die Anzahl der in UniConnect erstellten Communities, aufgeteilt nach der Sichtbarkeit, gibt die nachfolgende Tabelle 2-1.

Tab. 2-1: Anzahl der Communities

Sichtbarkeit	**Anzahl (Stand 19. September 2018)**
- öffentlich	40
- öffentlich mit Zugangskontrolle	55
- privat	1045
- **gesamt**	**1140**
Eine Übersicht der verwendeten SQL-Abfragen zum Erhalt der Zahlen befindet sich in Anhang 1.	

Auch die textuellen Daten der privaten Communites sind auf den Datenbanken frei verfügbar. Es findet dort keine Verschlüsselung oder Ähnliches statt. Eine Analyse ist hier also auch möglich.

Blogs

Sämtliche in der Blogs-Funktion erzeugten Inhalte werden in der gleichnamigen Datenbank gespeichert. Die textuellen Daten, bestehend aus Einträgen und Kommentaren, werden jeweils in einer separaten Tabelle gespeichert. Hierbei befinden sich die Einträge in der Tabelle *weblogentry*. Die benötigten Attribute dieser sind *WEBSITEID* und *TEXT*. Wie der Name bereits vermuten lässt, befinden sich in der Spalte *TEXT* die textuellen Inhalte der Beiträge in Form von Rich Text oder HTML. Jeder Eintrag ist zudem über die Spalte *WEBSITEID* mit der UUID des zugrundeliegenden Blogs verknüpft. Zusätzlich erhält jeder Eintrag eine eigene ID. Diese ID dient in der für die Speicherung der Kommentare vorgesehenen Tabelle *roller_comment* als Fremdschlüssel zur Verknüpfung mit dem zugehörigen Eintrag. Die Inhalte der Kommentare sind dort ebenfalls unter dem Attribut *TEXT* zu finden. Die Assoziation eines Blogs zu seiner Community lässt sich über die Tabelle *websiteassoc* mithilfe seiner UUID über das Attribut *ASSOCID (= UUID der Community)* bewerkstelligen. Mittels Transition ist es dann möglich, eine Verbindung der Community zu den Einträgen und Kommentaren herzustellen.

Eine Übersicht über die Anzahl der in UniConnect erstellten Beiträge in der Funktion Forum, gibt die nachfolgende Tabelle 2-2.

Tab. 2-2: Anzahl der Beiträge der Funktion Blogs

Art	**Anzahl (Stand 19. September 2018)**
- Einträge - Kommentare - **gesamt**	2912 2183 **5095**
Eine Übersicht der verwendeten SQL-Abfragen zum Erhalt der Zahlen befindet sich in Anhang 1.	

Forum

Die mit der Funktion Forum erzeugten Inhalte werden auf der gleichnamigen Datenbank gespeichert. Eine Besonderheit dieser ist, dass die benötigten textuellen Daten, bestehend aus Themen und Antworten, in einer gemeinsamen Tabelle *df_node* gespeichert werden. Der benötigte Inhalt eines Beitrags in Form von Rich Text oder HTML wird unter dem Attribut *DESCRIPTION* abgelegt. Die Unterscheidung, ob es sich bei einem Beitrag um ein Thema oder eine Antwort handelt, geschieht über die Spalte *NODETYPE*. Ein Thema wird hierbei mit der Bezeichnung *„forum/topic“* versehen, wohingegen eine Antwort die Bezeichnung *„forum/reply“* trägt. Beziehungen zwischen Foren und ihren Beiträgen werden ebenfalls mittels UUIDs hergestellt. Jedes Forum, erkennbar durch die Bezeichnung *„application/forum“* in der Spalte *NODETYPE*, erhält eine UUID über das Attribut *FORUMUUID*. Mittels dieser kann über die Tabelle *df_nodecommmap* eine Referenz zur zugehörigen Community (über ihre UUID)

hergestellt werden. Ferner erbt jedes erstellte Thema die FORUMUUID des zugehörigen Forums als *PARENTUUID* und zusätzlich eine eigene UUID über die Spalte *TOPICID*. Diese wiederum werden auf die Antworten des Themas über das gleiche Attribut vererbt. Somit lassen sich zu allen Beiträgen Verbindungen herstellen.

Eine Übersicht über die Anzahl der in UniConnect erstellten Beiträge in der Funktion Forum, gibt die nachfolgende Tabelle 2-3.

Tab. 2-3: Anzahl der Beiträge der Funktion Forum

Art	**Anzahl (Stand 19. September 2018)**
- Foren - Themen - Antworten - **gesamt** *(Themen + Antworten)*	1279 2126 6039 **8165**
Eine Übersicht der verwendeten SQL-Abfragen zum Erhalt der Zahlen befindet sich in Anhang 1.	

Statusaktualisierungen

Statusaktualisierungen und ihre Kommentare werden in der Datenbank *homepage* gespeichert. Dort werden in der Tabelle *board_entries* sowohl die Statusaktualisierungen gespeichert, die in Communities erstellt werden, als auch die, welche über die Profile der Plattformnutzer abgesetzt wurden. Eine Unterscheidung ist über das Attribut *SOURCE* möglich. Der Eintrag *„profiles“* steht dabei für einen über ein Profil erstellten Beitrag. Ein in einer Community geteilter Beitrag ist mit *„communities“* versehen. Wie bereits zuvor erwähnt, werden nur Beiträge der letztgenannten Quelle berücksichtigt. Die textuellen Inhalte werden in der Spalte *CONTENT* abgelegt. Zum Aufbauen von Beziehungen werden jedem Beitrag zwei IDs zugeordnet. Mittels der in der Spalte *CONTAINER_ID* abgelegten ID lässt sich die Assoziation eines Beitrags zu seiner zugehörigen Community über die Tabellen *board* und *person* herstellen. Über das Attribut *ENTRY_ID* erhält jeder Beitrag eine UUID, die als Fremdschlüssel in der für Kommentare vorgesehenen Tabelle *board_comments* dient. Auch hier wird der textuelle Inhalt in der Spalte *CONTENT* abgelegt.

Eine Übersicht über die Anzahl der in UniConnect erstellten Beiträge in der Funktion Statusaktualisierungen, gibt die nachfolgende Tabelle 2-4.

Tab. 2-4: Anzahl der Beiträge der Funktion Statusaktualisierungen

Art	**Anzahl (Stand 19. September 2018)**
- Beiträge - Kommentare - **gesamt**	2681 2400 5081
Eine Übersicht der verwendeten SQL-Abfragen zum Erhalt der Zahlen befindet sich in Anhang 1.	

Wikis

Wikieinträge und ihre Kommentare werden in der Datenbank *wikis* in separaten Tabellen gespeichert. Die textuellen Daten der Einträge befinden sich in der Tabelle *media*. Eine Besonderheit hierbei ist, dass in den Datenbanken nur gekürzte Versionen der Einträge gespeichert werden. Die vollständigen Einträge werden in einem bisher unzugänglichen XML-Format gespeichert. Für die spätere Analyse wird daher auf die verkürzten Versionen, die in der Spalte *SUMMARY* als Rich Text oder HTML Code gespeichert werden, zurückgegriffen. Jeder Eintrag besitzt zudem eine eigene UUID unter dem Attribut *ID*. Diese dient als Fremdschlüssel in der Tabelle *media_comment*, dem Speicherort der zugehörigen Kommentare und erlaubt die Verknüpfung zwischen Einträgen und Kommentaren. Der Inhalt der Kommentare ist in der Spalte *COMMENT* abgelegt. Zur Herstellung einer Verbindung zwischen den Einträgen und Kommentaren der Wikis und den zugehörigen Communities, besitzen beide Tabellen das Attribut *LIBRARY_ID*. Mithilfe dessen kann in der Tabelle *LIBRARY* auf die UUID der Community referenziert werden.

Eine Übersicht über die Anzahl der in UniConnect erstellten Beiträge in der Funktion Wikis, gibt die nachfolgende Tabelle 2-5.

Tab. 2-5: Anzahl der Beiträge der Funktion Wikis

Art	**Anzahl (Stand 19. September 2018)**
- Einträge - Kommentare - **gesamt**	5627 1560 **7187**
Eine Übersicht der verwendeten SQL-Abfragen zum Erhalt der Zahlen befindet sich in Anhang 1.	

Es ist zu erkennen, dass die Speicherung der Daten der Funktionen Blogs, Statusaktualisierungen und Wikis, durch das Trennen der Tabellen für Beiträge und Kommentare, ähnlich verläuft. Ein Grund für die abweichende Art der Speicherung der Beiträge der Funktion Forum ist nicht bekannt.

Wie bei Datenbanken üblich, erfolgt die Datengewinnung mittels SQL. Der grundsätzliche Aufbau einer Abfrage nach dem textuellen Inhalt verläuft dabei bei allen vier Funktionen ähnlich und wird vereinfacht in der nachfolgenden Abbildung 2.5 gezeigt.

```
SELECT text
  FROM einträge/themen
       UNION ALL (SELECT text
                  FROM kommentare/antworten)
```

Abb. 2.5: Grundsätzlicher Aufbau einer SQL Abfrage (vereinfacht)

Eine sprachliche Ausformulierung dessen lautet wie folgt:

Selektiere (FROM) die Tabelle, welche die Einträge beziehungsweise Themen der gewünschten Funktion enthält und wähle (SELECT) daraus die Spalte mit dem textuellen Inhalt. Mache dies ebenfalls für die Tabelle, welche die Kommentare beziehungsweise die Antworten enthält. Vereine (UNION ALL) nun die beiden Spalten horizontal.

Eine Übersicht über die konkret für die Datengewinnung verwendeten SQL-Abfragen, aufgeteilt in die jeweiligen Funktionen, befindet sich in Anhang 2. Diese enthalten zusätzlich eine Verknüpfung (JOIN ... ON) mit den Assoziationstabellen, um eine Referenz zwischen den einzelnen Beiträgen und ihren zugehörigen Communities herstellen zu können.

2.3 Social Collaboration Analytics

SCA ist ein Begriff, der im Jahr 2017 von Schwade und Schubert als Bezeichnung für die Untersuchung von Logfiles und der textuellen Daten zum besseren Verständnis der Nutzung von ECS geprägt wurde (Schwade & Schubert, 2017).

Grundlegendes Prinzip von SCA sind Data Analytics größerer Datenmengen, die in diesem Fall im Rahmen der Verwendung von Kollaborationssystemen anfallen (W. Tan, Blake, Saleh, & Dustdar, 2013). Data Analytics beschreiben dabei die Tätigkeit, große Datenmengen aus unterschiedlichen Quellen zu beschaffen, zu analysieren, aufzubereiten und anschließend sinnvoll zu präsentieren (Litzel, 2016). SCA sind somit ein Teilgebiet der Data Analytics, deren Analysen auf Daten aus Kollaborationssystemen basieren. ECS besitzen für solche Analysen drei Datenquellen, die in der nachfolgenden Abbildung 2.6 gezeigt und in deren Anschluss erläutert werden.

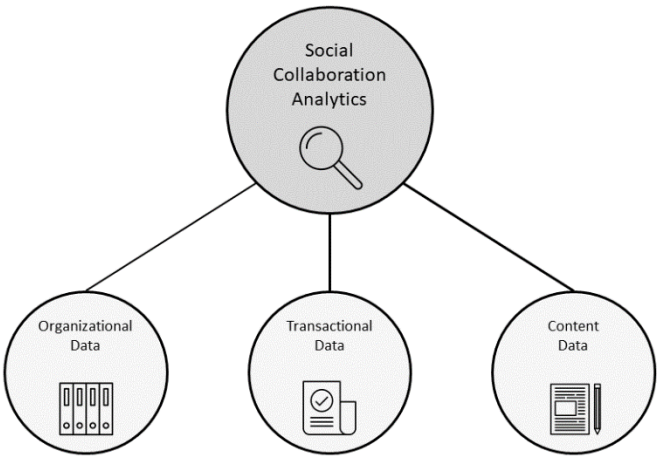

Abb. 2.6: Datenquellen für SCA (nach Schwade & Schubert, 2018)

Erläuterung der Datenquellen nach Schwade & Schubert (2017):

Organizational Data

Die Organizational Data umfasst jene Daten, die mit der Organisation eines ECS in Verbindung stehen, darunter die grundlegende Struktur und das Nutzerverzeichnis mit Nutzerverbindungen und personenspezifischen Zugriffsrechten. Ein Beispiel in Hinblick auf UniConnect bilden in diesem Fall Informationen zu den Nutzern und ihren Verbindungen untereinander (ESN) sowie zu den Communities und den zugehörigen Zugriffsrechten.

Transactional Data

Die Transactional Data umfasst die Einträge zu sämtlichen von den Nutzern ausgeführten Aktionen im ECS. Sie erfüllt somit den Zweck des Loggings und speichert zu jeder getätigten Aktion ihren Typ, den Inhalt, den Nutzer und einen Zeitstempel. In UniConnect übernimmt diese Aufgabe die bereits erwähnte Metrics-Datenbank.

Content Data

Die letzte und für diese Arbeit relevante Datenquelle ist die sogenannte Content Data in Form der bereits mehrfach thematisierten textuellen Inhalte. Sie umfasst jegliche vom Nutzer erzeugten Inhalte, darunter, im Fall von UniConnect, die Beiträge der Funktionen Blogs, Forum, Statusaktualisierungen und Wikis. Es ist möglich Verbindungen zwischen Transactional Data und Content Data herzustellen.

Die Ergebnisse von Data Analytics sind vielfältig (siehe Tabelle 2-5). Im Kontext von SCA konnten Schwade und Schubert (2018) in einer Literaturanalyse feststellen, dass die Ergebnisse stets auf einem von acht Schlüsselthemen beruhen:

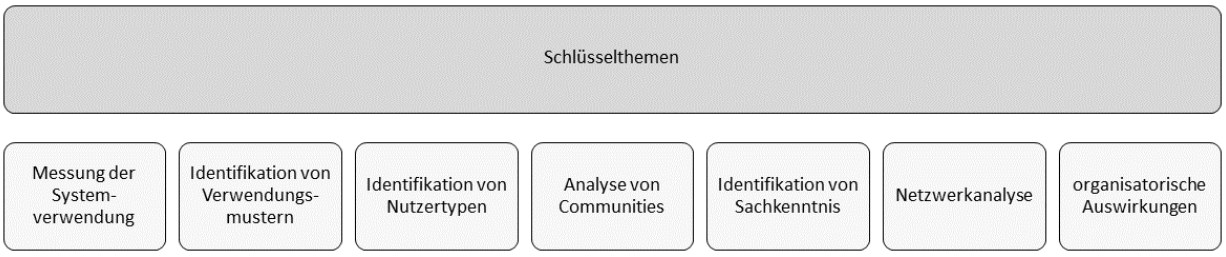

Abb. 2.7: Identifizierte Schlüsselthemen von Social Collaboration Analytics (übersetzt)

In der folgenden Tabelle 2-6 werden die Schlüsselthemen kurz erläutert und mit Beispielen aus der Forschung ergänzt.

Tab. 2-6: Erläuterung der Schlüsselthemen von SCA mit Forschungsbeispielen

Schlüsselthema	Erläuterung	Forschungsbeispiele
Messung der Systemverwendung	Analyse der Systemverwendung auf zwei Ebenen: Ebene 1: Messung der allgemeinen Nutzung der Plattform auf höherem Level Ebene 2: Aktionsspezifische Messung	Analyse der Nutzung der Kommunikationsfunktionen (Behrendt, Richter, & Trier, 2014)
		Messung des Erfolgs der Nutzung eines ESS anhand der Analysen von Metriken (Herzog, Richter, & Steinhüser, 2015)
		Messung der Verwendung von ESS durch Betrachtung von Verwendungsstatistiken, Grad der Vernetzung und Anzahl der Hyperlinks, DMs und Kommentare (Steinhüser, Herzog, Richter, & Hoppe, 2015)
Identifikation von Verwendungsmustern	Identifikation der Use Cases und Collaboration Scenarios (Schubert & Glitsch, 2016): Use Case: Veranschaulichung des geschäftlichen Nutzens, der aus der Verwendung der ECS abgeleitet werden kann Collaboration Scenario: Veranschaulichung, wie die Akteure, Aufgaben und Interaktionen durch Technologie unterstützt werden können	Identifikation von Verwendungsmustern durch Process Mining (Chaves & Córdoba, 2014)
		Identifikation von Verwendungsmustern durch Betrachtung der Intensität und Häufigkeit der Funktionen in verschiedenen Communities (Muller et al., 2012)
		Analyse von Sequenzen aus Aktionen und Interaktionen (Bøving & Simonsen, 2004)
Identifikation von Nutzertypen	Identifikation typischer Charakteristika von verschiedenen Nutzertypen: Ansatz 1: Identifikation basierend auf der Intensität der Systemverwendung Ansatz 2: Identifikation basierend auf der Art der Nutzung	Häufigkeit der Nutzung der Plattform als Maßstab (Appelt, 2001)
		Zuordnung der Nutzer in die Kategorien lurkers („Lauerer“, nur passive Beteiligung), contributors (Mitwirkender) und uploaders (Erschaffer von neuen Inhalten), basierend auf den Tätigkeiten in einem ESS (Muller et al., 2012)
		Zuordnung von Rollen (Helfer, Beteiligter, Suchender) basierend auf den Ausgeführten Handlungen im ESS (Hacker, Bodendorf, & Lorenz, 2017)
Analyse von Communities	Analyse der Geschehen innerhalb von Communities	Metriken zur Messung der Produktivität, Aktivität und Kooperativität einer Community (Jeners & Prinz, 2014)

Identifikation von Sachkenntnis	Erkennen und Auffinden von Sachkenntnis	Nutzer, die häufig Beiträge zu einem Thema erstellen, besitzen eine besonders hohe Kenntnis für ebendieses (Nasirifard & Peristeras, 2009)
Netzwerkanalyse	Identifikation der Charakteristika von Netzwerken Erkennung der verschiedenen Arten von Netzwerken	Analyse der sozialen Interaktionen zwischen Nutzern (Smith, Hansen, & Gleave, 2009)
		Messung der Ausbreitungsgeschwindigkeit von Informationen innerhalb eines Netzwerks (Behrendt & Richter, 2015)
Organisatorische Auswirkungen	Messung der organisatorischen Auswirkungen durch und innerhalb der ESS	Hat die formale Hierarchie einen Einfluss auf die Verhaltensweisen der Nutzer im ESN? (Stieglitz, Riemer, & Meske, 2014)

Auffällig ist, dass für die zuvor genannten Beispiele aus der Forschung hauptsächlich Transactional Data und Organizational Data als Datengrundlagen[4] dienen. Eine Verwendung der Content Data zur Textanalyse ist in ECS bisher nicht gängig. Lediglich im Bereich der Social Media finden solche Analysen bereits statt (Stieglitz, Dang-Xuan, Bruns, & Neuberger, 2014).

Ein Grund für die fehlende Betrachtung der Content Data von ECS liefert die Verwendung von Text Mining Methoden, die mehrere Herausforderungen mit sich bringen. Während die Analyse von Transactional Data und Organizational Data hauptsächlich auf einfach zu verarbeitenden Zahlendaten beruht, umfasst Text Mining eine große sprachliche Komponente, bestehend aus unstrukturierten textuellen Daten (Bhardwaj, 2016; A.-H. Tan, 1999). Die schwerwiegendste Herausforderung ist jedoch die Mehrdeutigkeit der Sprache, die dazu führen kann, dass ein Wort oder ein Satz mehrere Bedeutungen besitzen kann, die mitunter auch gegenteilig sein können (Bhardwaj, 2016). Eine dazu nötige semantische Analyse ist rechenintensiv und für größere Umfänge an Texten nur gering skalierbar (A.-H. Tan, 1999). Ein weiteres Problem ist die Nutzung von umgangssprachlichen oder regionsspezifischen Begriffen, die nicht in Wörterbüchern gelistet sind und somit die Ergebnisse des Text Mining beeinflussen (Bhardwaj, 2016). Hinzu kommen Rechtschreibfehler, die ebenfalls einen Einfluss haben. Weiterhin ist die bereits angesprochene Multilingualität ein Problem. Nur wenige Text Mining Tools beherrscht diese, sodass in der Regel eine aufwändige Verwendung von mehreren Tools nötig ist (Talib, Hanif, Ayesha, & Fatima, 2016).

Ein weiterer, selbst vermuteter Grund, kann durch datenschutzrechtliche Bedenken bei der Verarbeitung von Beiträgen mit vertraulichen Inhalten oder der Analyse von Beiträgen aus privaten Communities begründet sein. Insbesondere bei der Nutzung von SCA Tools, in der Verbindung mit Text Mining Methoden, wie sie im weiteren Verlauf der Arbeit vorgestellt werden, können sensible Daten preisge-

[4] Bei der Identifikation von Sachkenntnis kann die Content Data berücksichtigt werden. Siehe zugehöriges Forschungsbeispiel in Tabelle 2-5.

geben werden, die dann im ungünstigsten Fall unberechtigten Mitarbeitern oder Dritten zugängig sind.

Im Rahmen der Regeln und Richtlinien zur Nutzung von UniConnect[5] („Terms of Use“), weist das UCT darauf hin, dass die Plattform dem Zweck der Lehre und Forschung dient und registrierte Nutzer mit ihrer Anmeldung einwilligen, dass ihre Nutzungsdaten für ebendiese Forschung verwendet werden dürfen. Dies legitimiert das im Rahmen dieser Arbeit erstellte Dashboard zur Analyse der Content Data und die gleichzeitige Forschung, wie Text Mining Methoden auf diese Daten angewendet werden können. Zudem ist der Zugriff auf das Dashboard nur den Mitarbeitern des UCT vorbehalten. Sichergestellt wird dies durch eine verpflichtende Authentifizierung mittels Benutzernamen und Passwort.

[5] https://uct.de/terms-of-use

3 Text Mining

Bereits zu Beginn dieser Arbeit wurde in der Einleitung erwähnt, dass es sich bei Text Mining um ein Teilgebiet des Data Mining handelt. Das erste Kapitel 3.1. dient dazu, eine Einführung in das Themengebiet des Text Mining zu erhalten und die Abgrenzung zum Data Mining genauer zu erläutern. Die darauffolgenden Kapitel dienen zur Demonstration der in dieser Arbeit verwendeten Methoden des Text Mining. In Kapitel 3.2 wird dazu zunächst das Konzept des Preprocessings vorgestellt, welches einen wichtigen Arbeitsschritt im Rahmen der Vorarbeit zum Text Mining darstellt und den Bezug zur Arbeitsphase Data Preparation herstellt. Weitere Methoden werden dann in Kapitel 3.3 dargelegt.

3.1 Definition

Text Mining ist ein noch sehr junges Forschungsgebiet, welches als Teilgebiet des Data Mining etwa seit der Jahrtausendwende zunehmend an Interesse gewinnt (Debortoli et al., 2016). Analog existieren die Bezeichnungen Intelligent Text Analysis, Knowledge-Discovery in Text und, in Anlehnung an die Abstammung, Text Data Mining (Gupta & Lehal, 2009). Im Allgemeinen beschreibt Text Mining den rechnergestützen Prozess der Extraktion von interessanten und nicht-trivialen Informationen sowie von Wissen aus großen Mengen unstrukturierter Texte (Debortoli et al., 2016; Gupta & Lehal, 2009). Im Rahmen dessen finden auch Methoden des Information Retrievals, des Data Mining, des Machine Learnings, der Statistik und der Computerlinguistik ihre Anwendung, dabei vor allem im Rahmen des Natural Language Processings (NLP, siehe Kapitel 3.3) (Gupta & Lehal, 2009).

Auffällig häufig werden in der Literatur die Begriffe Text Mining und Data Mining synonym verwendet, sie unterscheiden sich jedoch signifikant in der Datengrundlage (Hippner & Rentzmann, 2006). So dienen dem Data Mining strukturierte Daten als Grundlage, die beispielsweise als erste Normalform (1NF) gemäß der Spezifikation von Datenbanken vorliegen und nicht weiter zerlegbar sind (Hippner & Rentzmann, 2006). Text Mining besitzt hingegen eine Datenbasis aus semi-strukturierten oder unstrukturierten Daten (Vijayarani, Ilamathi, & Nithya, 2015). Unstrukturierte Daten umfassen dabei Informationen, die üblicherweise nicht in einer traditionellen Datenbank mit Tabellen-Aufbau gespeichert werden (Vijayarani et al., 2015). Dies umfasst beispielsweise HTML-Dateien, E-Mails und Textdokumente. Semi-Strukturierte Daten bilden eine Kreuzung der vorangegangenen Begriffe (Vijayarani et al., 2015). Ein passendes Beispiel dazu ist die Content Data des ECS UniConnect. Sie ist weder als reine Textdatei noch als unzerlegbarer atomarer Wert gespeichert. Text Mining kann somit höchstens als Ausprägung des Data Mining angesehen werden.

Der Prozess des Text Mining kann durch mehrere Modelle beschrieben werden. Eines davon ist das bereits in Kapitel 1.3 im Zusammenhang mit dem Forschungsdesign dieser Arbeit genannte CRISP-DM, welches ursprünglich für Data Mining entwickelt wurde, jedoch auch auf Text Mining übertragbar ist. Ein weiteres Modell von Hippner & Rentzmann (2006) wird im Folgenden vorgestellt und die einzelnen Iterationen näher beschrieben.

Abb. 3.1: Iterativer Text Mining Prozess nach Hippner & Rentzmann (2006)

Wie in Abbildung 3.1 zu erkennen ist, ähneln sich die beiden Modelle in ihrem Aufbau.

Der Prozess beginnt mit der Aufgabendefinition mit dem Ziel der Festlegung der Problemstellung und der Ableitung der Text Mining Ziele. Darauf aufbauend gilt es in der Phase der Dokumentselektion die für das Erreichen der Ziele relevanten Dokumente zu identifizieren. Der nachfolgende Schritt, die Dokumentaufbereitung, umfasst das sogenannte Preprocessing. Eine umfassende Beschreibung dazu befindet sich im nächsten Kapitel. Nach Abschluss des Preprocessings folgt das Anwenden der Text Mining Methoden. Dies umfasst beispielsweise das automatische Zuordnen von Texten zu vorgegebenen Kategorien (Klassifikation) oder eine Abhängigkeitsanalyse mit dem Ziel, gemeinsam auftretende Terme zu erkennen. Diese und weitere Methoden werden in Kapitel 3.3 näher betrachtet. Im Rahmen des Schritts Interpretation / Evaluation werden die zuvor erhaltenen Ergebnisse näher analysiert und die handlungsrelevanten Resultate ausgefiltert und bewertet. Im letzten Schritt folgt die Anwendung der Ergebnisse. Beispielhaft dafür ist die Visualisierung der Ergebnisse. Im Fall dieser Arbeit umfasst dieser Schritt die Implementierung der Text Mining Java-Klassen in das Dashboard.

3.2 Preprocessing

Wie bereits im zurückliegenden Kapitel erwähnt wurde, dienen unstrukturierte und semi-strukturierte Informationen als Datengrundlage für das Text Mining. Während beim Data Mining die strukturierten Daten direkt den zu verarbeitenden Analyseeinheiten entsprechen, erfordert das Text Mining eine zusätzliche Bearbeitung der Daten in Form des Preprocessings, um sie für die Kernmethoden (Analyse) verwendbar zu machen (Feldman & Sanger, 2007; Hippner & Rentzmann, 2006). Das Preprocessing spielt somit eine wichtige Rolle im Rahmen des Text Minings Prozesses und umfasst laut Vijayarani et. al. (2015) vier Schritte, die in Abbildung 3.2 dargestellt sind. Diese werden dabei durch Rechtecke dargestellt. Zudem werden Zustände durch Rechtecke mit runden Ecken und Datenquellen, die zum Durchführen eines Schritts benötigt werden, durch ein Plattenspeichersymbol dargestellt. Eine genaue Beschreibung der vier Schritte folgt im Anschluss der Abbildung. Eine beispielhafte Anwendung der ersten drei Preprocessing-Schritte wird in Abbildung 3.3 gezeigt. Verwendet wird dabei ein Zitat von Steve Jobs aus dem Jahr 2003.

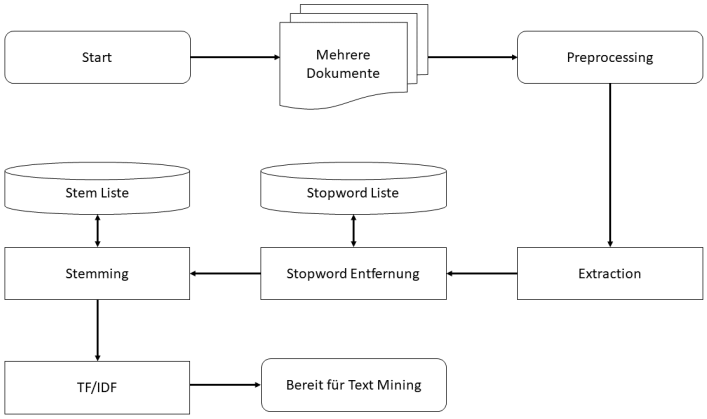

Abb. 3.2: Schritte des Preprocessings (teilweise übersetzt, Vijayarani et al., 2015)

Extraction

Im Rahmen der Extraction findet eine Normalisierung und eine sogenannte Tokenization der textuellen Daten statt. Ziel der Tokenization ist es, den Text in seine einzelnen Bestandteile zu zerlegen. Üblich ist dabei die Zerlegung in einzelne Wörter, die sogenannten Tokens (Feldman & Sanger, 2007). Dabei werden zudem jede im Text vorkommende Zahl und jedes Satzzeichen zu einem separaten Token. In der darauffolgenden Normalisierung werden die Tokens in eine einheitliche Struktur gebracht. Dazu werden alle Tokens mit Satzzeichen und Zahlen entfernt, da sie für die weitere Analyse keine Aussagekraft besitzen. Außerdem werden alle Buchstaben der Tokens in ihre Minuskeln umgewandelt.

Stopword Entfernung

Stopwords werden im Kontext des Text Mining Wörter genannt, die für die Relevanz eines Texts nicht ausschlaggebend sind und dementsprechend entfernt werden können (Vijayarani et al., 2015). Beispiele dafür bilden Artikel, Konjunktionen, Präpositionen und Pronomen. Da sie einen nicht unerheblichen Teil eines Textes ausmachen, kann durch die Entfernung eine deutliche Reduzierung der Dimension der Texte erreicht werden. Die gängige Methode der Stopword-Erkennung verläuft über bereits erzeugte oder selbst angelegte Listen mit vorab definierten Stopwords (Vijayarani et al., 2015). Solche Listen werden für die meisten Sprachen im Internet angeboten. Ferner verfügen Tools, die für die Ausführung von Preprocessing vorgesehen sind, über eigene Listen. Zur Überprüfung, ob es sich bei einem Wort bzw. einem Token eines Texts um ein Stopword handelt, wird dann geprüft, ob das Wort in der Stopword Liste vorhanden ist. Ist dies der Fall, so wird der entsprechende Token entfernt.

Stemming

Stemming bezeichnet eine Methode, die Grundform eines Worts zu bilden, beispielsweise durch die Entfernung des „Plural s“ bei Nomen oder der Suffixe von konjugierten Verben (Hotho, Nürnberger, & Paaß, 2005). Die von Präfixen und Suffixen befreite Version eines Worts wird Stem genannt und entspricht nicht immer einer orthografisch korrekten Form eines Worts (siehe Beispiel in Abbildung 3.3).

Ein Stem repräsentiert weiterhin eine Gruppe von Wörtern mit der gleichen (oder einer sehr ähnlichen) Bedeutung (Hotho et al., 2005). Die Informationen darüber, wie ein Stem gebildet wird, befindet sich in einer zuvor angelegten Stem Liste in Form von Anwendungsregeln. Ein bekannter Stemming Algorithmus stammt von M. F. Porter aus dem Jahr 1980 (Porter et al., 2006). Anzumerken ist, dass die meisten Stemming Algorithmen für die englische Sprache existieren.

Eine Abwandlung des Stemmings ist die Lemmatization mit dem Ziel, Wörter auf eine orthografisch korrekte und verkürzte Grundform abzubilden. Dazu werden Verben in ihren Infinitiv und Nomen in ihre Singular-Form umgewandelt (Hotho et al., 2005). Voraussetzung dafür ist jedoch, dass zu jedem Wort die Grundform bekannt ist.

TF/IDF

TF/IDF steht für Term Frequency / Inverse Document Frequency und bezeichnet eine Statistik, die aufzeigt, wie wichtig ein Wort für ein Dokument in einer Sammlung von Dokumenten ist (Vijayarani et al., 2015). Dabei steigt der TF/IDF-Wert eines Worts proportional zu der Anzahl des Aufkommens des Worts innerhalb eines Dokuments. Dementgegen setzt sich die Häufigkeit des Worts innerhalb der gesamten Dokumentensammlung (Vijayarani et al., 2015). Somit gilt, dass ein höherer Wert bedeutet, dass ein Wort in den weiteren Dokumenten seltener vorkommt und folglich eine höhere Relevanz für sein Ursprungsdokument besitzt.

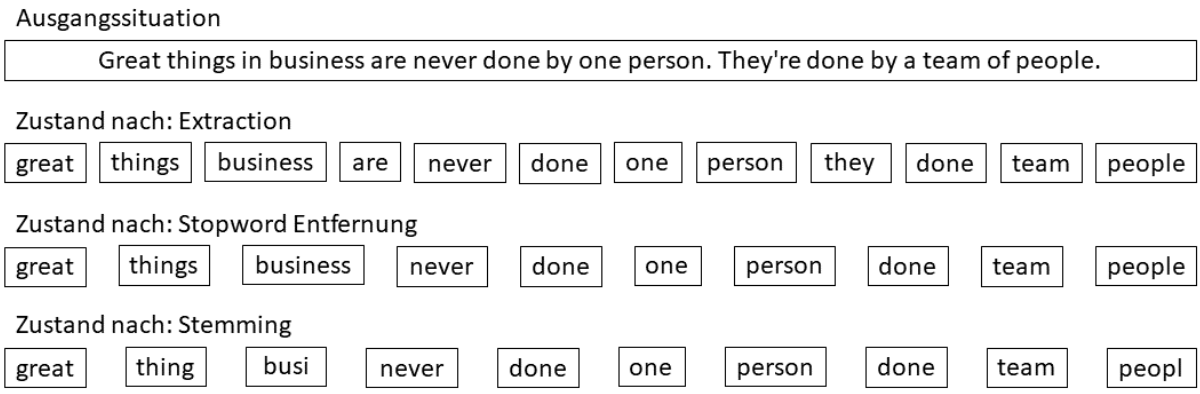

Abb. 3.3: Beispielhafte Anwendung der Preprocessing-Schritte[6]

Eine TF/IDF Bestimmung wurde in Abbildung 3.3 ausgelassen, da sie im Falle des sehr geringen Textumfangs und des Fehlens von weiteren Dokumenten nicht aussagekräftig wäre. Es ist zudem zu erkennen, dass die aus dem Stemming resultierenden Stems teilweise orthographisch inkorrekt sind. Eine Zuordnung dieser zu einer bestimmten Wortgruppe ist aber dennoch möglich.

[6] Die Stopword Entfernung wurde anhand der englischen Stopword Liste des Python-Frameworks NLTK (siehe Kapitel 4.1) durchgeführt. Die Stemming-Methode basiert auf den Algorithmus von Porter (Porter et al., 2006).

3.3 Zum Einsatz kommende Methoden

Das Forschungsgebiet des Text Mining umfasst sieben Anwendungsbereiche (Miner et al., 2012). Die nachfolgende Abbildung 3.4 zeigt diese Anwendungsbereiche in Form eines Venn-Diagramms als Schnittmenge der übergeordneten Themenfelder (mit * versehen).

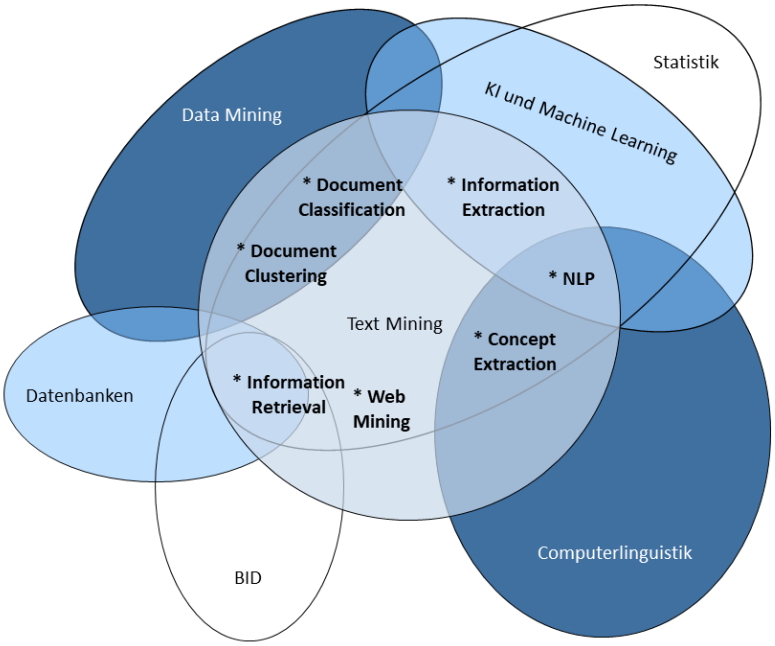

Abb. 3.4: Anwendungsbereiche des Text Mining in Bezug auf ihre übergeordneten Themenfelder (Miner et al., 2012) [BID = Bibliotheks-, Informations- und Dokumentationswissenschaft; NLP = Natural Language Processing]

Im Rahmen dieser Arbeit werden fünf dieser Anwendungsbereiche durch die Analyse der Content Data von UniConnect abgedeckt. Dabei handelt es sich um Document Classification, Document Clustering, Information Extraction, Natural Language Processing und Concept Extraction. Diese werden im Folgenden allgemein erläutert. Eine kurze Beschreibung der zwei weiteren Anwendungsbereiche folgt in Tabelle 3-2.

Document Classification

Das Ziel der Document Classification ist die Zuordnung von vordefinierten Klassen zu Dokumenten (Hotho et al., 2005). Ein Beispiel dafür ist die Zuordnung von bestimmten Themen, wie beispielsweise „Sport“, „Politik“ oder „Unterhaltung“, zu Artikeln von Nachrichtenportalen. Bekannte Klassifikationsmethoden sind Naive Bayes, K-Nearest-Neighbor, Decision Trees und Support Vector Machines. Die Basis der zuvor genannten Methoden ist stets ein sogenanntes Training Set von Dokumenten, die bereits einer Klasse zugordnet sind (Hotho et al., 2005). Anhand dieses Sets können die Methoden dann ein Klassifikationsmodell für die Zuordnung von neuen Dokumenten erstellen.

Document Clustering

Document Clustering beschreibt als Teilgebiet des Data Clustering den Prozess, Dokumente in verschiedene Gruppen, genannt Cluster, einzuordnen (Miner et al., 2012). Dabei teilen Dokumente, die in dasselbe Cluster eingeordnet werden, gemeinsame Eigenschaften, die durch Ähnlichkeitsermittung definiert werden (Shah & Mahajan, 2012). Dabei macht es Gebrauch von Konzepten des Information Retrieval (siehe Tabelle 3-2), des Natural Language Processings und des Machine Learnings (ML) (Shah & Mahajan, 2012). Document Clustering unterscheidet sich von der zuvor vorgestellten Document Classification dadurch, dass die zu identifizierenden Klassen zuvor unbekannt sind.

Information Extraction (IE)

IE beschreibt die Aufgabe, automatisch Informationen und Fakten aus unstrukturierten und semistrukturierten Daten zu gewinnen (Allahyari et al., 2017). Es umfasst in der Regel mehrere Arbeitsschritte, darunter Tokenization, Satzsegmentierung, Part-of-Speech Zuordnung (Zuordnung zu Wortart) und die Identifikation von sogenannten Named Entities, also Objekten aus der realen Welt (Hotho et al., 2005). Darunter fallen beispielsweise Namen von Personen, Orten oder Organisationen (Hotho et al., 2005). Ein Beispiel für die Identifikation von Named Entities zeigt die nachfolgende Abbildung 3.5.

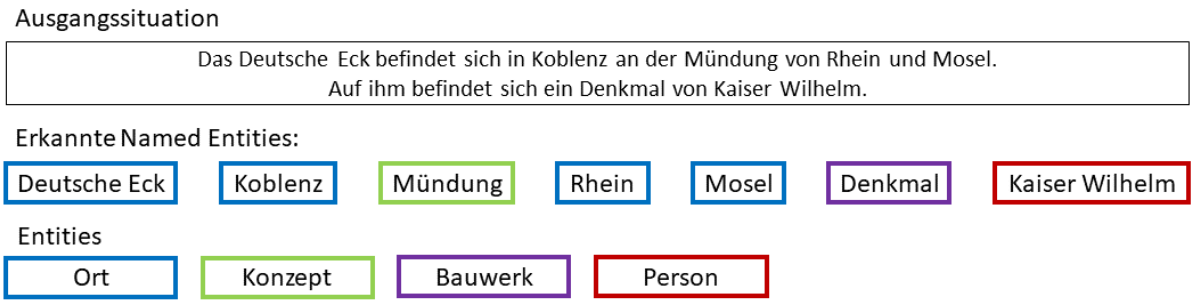

Abb. 3.5: Beispielhafte Identifikation von Named Entities[7]

Natural Language Processing (NLP)

NLP beschreibt eine Reihe von rechnergestützten Techniken zur Analyse und Darstellung von in natürlicher Sprache verfassten Texten auf der Basis von linguistischer Analyse, deren Ziel es ist, eine dem Menschen ähnliche Sprachverarbeitung zu erzielen (Liddy, 2001). Letzterer Punkt deckt auf, dass NLP auch als Disziplin innerhalb des Gebiets der künstlichen Intelligenz zugeordnet werden kann. Die für NLP verwendeten Texte können in jeglicher Sprache verfasst sein, sollten jedoch dem üblichen Sprachgebrauch zur Kommunikation mit Anderen entsprechen und nicht speziell für den Zweck des

[7] Die Identifikation der Named Entities basiert auf einer Demo der Dandelion API von SpazioDati (https://dandelion.eu/semantic-text/entity-extraction-demo/)

NLP konstruiert sein (Liddy, 2001). Einen Überblick darüber, auf welchen sprachlichen Ebenen NLP stattfinden kann, gibt die Tabelle 3-1.

Tab. 3-1: Sprachliche Ebenen, auf denen NLP angewendet werden kann (Liddy, 2001)

Sprachliche Ebene	**Beschreibung**
Phonologie (Lautlehre)	Interpretation von gesprochener Sprache durch Analyse und Umwandlung der Sprachdateien in digitale Signale
Morphologie (Wortbildung)	Untersuchung der einzelnen Morpheme eines Worts (kleinste Spracheinheiten) zum Erhalt des Verständnisses des gesamten Worts
Lexikalische Ebene	Interpretation der Bedeutung eines Worts basierend auf dem Kontext in dem es auftritt. Benötigt zur korrekten Funktionsweise ein Lexikon
Syntaktische Ebene	Analyse der grammatikalischen Struktur eines Satzes. Benötigt vorab definierte Grammatik und einen Parser
Semantische Ebene	Analyse der möglichen Bedeutung eines Satzes, auf Basis der Bedeutungen der einzelnen Wörter und ihres Kontexts
Diskursebene	Analyse von zusammenhängend strukturierten Gruppen von Sätzen, wie beispielsweise Monologe oder Dialoge (Kohärenz)
Pragmatik	Betrachtung der zielgerichteten Nutzung von Sprache in Situationen unter Beachtung des Kontexts

Concept Extraction

Concept Extraction, gelegentlich als Concept Mining bezeichnet, beschreibt den Vorgang, in Texten vorhandene Konzepte zu suchen oder zu gewinnen (Prasad, Saritha, & Saxena, 2017). Ein Konzept beschreibt dabei einen Begriff oder eine Menge von Begriffen, mit einer gemeinsamen Bedeutung oder einer Beziehung zwischen einzelnen Wörtern (Prasad et al., 2017). Ein Beispiel für ein Konzept ist „Brandenburger Tor“.

Tab. 3-2: Weitere Anwendungsgebiete des Text Mining (Miner et al., 2012)

Anwendungsgebiet	**Erläuterung**
Search and Information Retrieval (IR)	Aufbewahrung und Abruf von Textdokumenten, inklusive Suchmaschinen und Stichwortsuchen
Web Mining	Data und Text Mining im Internet, mit Fokus auf dem Ausmaß und der Vernetzung des Webs

Eine genaue Erläuterung der in den Java-Klassen eingesetzten Methoden der zuvor genannten Anwendungsgebiete folgt in Kapitel 4.1.

4 Die Text Mining Java-Klassen

Aus dem im vorherigen Kapitel erlernten theoretischen Wissen über Text Mining, folgt in diesem Kapitel die praktische Anwendung in Form der Text Mining Java-Klassen sowie der Wechsel in die Arbeitsphase „Modeling“. Kapitel 4.1 befasst sich zunächst mit der Vermittlung von grundlegenden Informationen und gibt eine Auflistung über die gewünschten Funktionen der Klassen. Einen Überblick darüber, wie die Funktionen konkret umgesetzt werden folgt in Kapitel 4.2.

4.1 Grundlagen

Die für diese Arbeit erstellten Text Mining Java-Klassen basieren auf der Programmiersprache Java[8] in der Version 1.6. Mit über neun Millionen Softwareentwicklern weltweit und wöchentlich zehn Millionen Downloads der Laufzeitumgebung, zählt die objektorientierte Programmiersprache zu einer der etabliertesten ihrer Art (Ullenboom, 2014). Zudem unterstützt das zur Entwicklung des Dashboards genutzte Framework XPages (siehe Kapitel 5.1) Java in vollem Umfang. Zur Erweiterung der Funktionsweise hinsichtlich der Text Mining Methoden, wurden diverse externe Libraries verwendet. Eine Auflistung der im Rahmen dieser Arbeit verwendeten Libraries befindet sich in Anhang 3.

Als Basis für die Erstellung der Java-Klassen diente die folgende Ausgangsfrage:

„Was sind populäre Themen auf der Plattform UniConnect und in welchen Communities werden sie behandelt?“

Aus dieser Frage abgeleitet ergeben sich die in den Klassen zu implementierenden Funktionen:

- Abruf der Content Data aus den DB2-Datenbanken
- Identifizierung von populären (communityübergreifenden) Themen und Begriffen durch Analyse der Content Data mittels verschiedener Methoden des Text Mining
- Visualisierung der Ergebnisse der Analyse

Darüber hinaus wird eine einfache Umsetzung einer Sentimentanalyse (Stimmungserkennung) implementiert. Diese fällt unter den Anwendungsbereich der Concept Extraction und wertet die Haltung eines Beitrags als positiv, neutral oder negativ aus.

Eine genaue Erläuterung der konkreten Umsetzung der Funktionen und der dafür verwendeten externen Libraries folgt im nächsten Kapitel.

[8] Java. Sun Microsystems, version 1.6. Available at http://www.oracle.com/technetwork/java/index.html

4.2 Umsetzung

Dieses Kapitel wird in die im vorherigen Kapitel genannten Funktionen unterteilt. Dabei wird für jede Funktion angegeben, wie diese konkret in Java umgesetzt wurde und welche externen Libraries dafür benötigt wurden. Ergänzt wird dies durch Codeausschnitte in Pseudocode-Notation und beispielhaften Visualisierungen von Analyseergebnissen. Die tatsächliche Umsetzung der Visualisierungen wird in Kapitel 5.3 näher erläutert und können von den hier gezeigten abgweichen.

Die Identifizierung der populären Themen wird im weiteren Verlauf dieses Kapitels in fünf Methoden unterteilt. Für jede dieser Methoden wurden eigene Java-Klassen erstellt, in welchen die entsprechenden Methoden unter Zuhilfenahme von Prozessen des Text Mining implementiert wurden. Zusätzlich wurde eine Klasse für die Sentimentanalyse erstellt. Jede dieser Java-Klassen umfasst zusätzlich Algorithmen zum Abruf der benötigten Content Data und zum Preprocessing dieser. Im Nachfolgenden werden die Umsetzungen dieser beiden Algorithmen beschrieben.

Abruf der Content Data

Der Abruf der Content Data aus den DB2-Datenbanken von UniConnect erfolgt über JDBC-Treiber. Zur Herstellung einer Verbindung zu einer der benötigten Inhaltsdatenbanken, werden der zugehörige Name, die Adresse, der Port und die Zugangsdaten des zugreifenden Benutzers benötigt. Im Falle einer bestehenden Verbindung, ist es dann möglich eine SQL-Abfrage auszuführen. Die Ergebnisse der Abfrage werden in eine String-Liste übertragen. Hierbei umfasst jeder Listeneintrag einen Beitrag der gewünschten Funktion. Der nachfolgende Quellcodeausschnitt 4-1 in Pseudocode-Notation zeigt den schematischen Ablauf einer Verbindung mit der BLOGS-Datenbank, der Ausführung einer beispielhaften SQL-Abfrage und der Speicherung der Ergebnisse.

QC 4-1: Ablauf des Abrufs und der Speicherung der Content Data mittels JDBC

```
#Festlegung der Verbindungsdaten und SQL-Abfrage
Treiber = "com.ibm.db2.jcc.DB2Driver";
Adresse = "jdbc:db2://db2gcc.uct.de:50001/blogs";
Nutzer = "maxmustermann";
Passwort = "123456";
Abfrage = "Gebe alle Blogeinträge zurück";

#Verbindung zur Datenbank und Durchführung der Abfrage
TreiberFestlegen(Treiber);
VerbindungHerstellen(Adresse, Nutzer, Passwort)
AbfrageDurchführen(Abfrage);

#Speicherung der Ergebnisse der Abfrage
solange ErgebnisVorhanden {
    Übertrage Ergebnis in Liste;
}
```

Der konkret verwendete Code kann in Anhang 4 eingesehen werden.

Preprocessing

Die Analyse von populären Themen wird durch verschiedene Herangehensweisen umgesetzt. Voraussetzung für jedes dieser Vorgehen ist das in Kapitel 3.2 erläuterte Preprocessing. Dieses wurde mithilfe der externen Libraries jsoup[9], language-detector[10] und snowballstem[11] umgesetzt. Hinzu kommen einfache Modifikationen mittels Java-eigenen Methoden. Hierzu zählen beispielsweise das Entfernen von Satzzeichen und Zahlen sowie das Umwandeln von Umlauten in ihre ausgeschriebene Form. Der in den Java-Klassen angewendete Ablauf des Preprocessings wird in Quellcode 4-2 als Pseudocode gezeigt.

QC 4-2: Ablauf des sprachenabhängigen Preprocessings

```
1  für jeden Eintrag in Liste {
2          EntferneHTMLCode();
3          EntferneHTMLCode();
4          EntferneUmlaute();
5          EntferneURLs();
6          EntferneZahlenUndZeichen();
7          ErkenneSprache();
8          wenn Sprache = deutsch {
9                  wenn Stemming = deaktiviert {
10                         StopwordEntfernungDE();
11                 } sonst {
12                         StopwordEntfernungDE();
13                         StemmingDE();
14                         StopwordEntfernungDE();
15                 }
16         } oder wenn Sprache = englisch {
17                 wenn Stemming = deaktiviert {
18                         StopwordEntfernungEN();
19                 } sonst {
20                         StopwordEntfernungEN();
21                         StemmingEN();
22                         StopwordEntfernungEN();
23                 }
24         } sonst {
25                 StopwordEntfernungIND();
26         }
27         Eintrag in Liste speichern;
28 }
```

Der konkret verwendete Code kann in Anhang 5 eingesehen werden.

Wie in Quellcode 4-2 zu erkennen ist, wird über jeden Eintrag der mittels Quellcode 4-1 erzeugten Liste iteriert und jeweils das Preprocessing durchgeführt. Jeder Beitrag wird somit einzeln verarbeitet. Zunächst wird jeder Beitrag mithilfe der Library jsoup zwei Mal von eventuell vorhandenem HTML Code befreit (Zeilen 2 und 3). Die erneute Anwendung von jsoup dient zur Entfernung von potenziell noch vorhandenen HTML Code nach der ersten Anwendung. Anschließend werden Umlaute in ihre

[9] https://jsoup.org

[10] https://github.com/optimaize/language-detector

[11] https://github.com/snowballstem/snowball

ausgeschriebene Form umgewandelt (Zeile 4) und URLs, Satzzeichen und Ziffern entfernt (Zeilen 5 und 6). Dieser Vorgang umfasst zudem eine Tokenization. Wie bereits in Kapitel 2.2.2 erwähnt wurde, wird UniConnect von den Nutzern bilingual verwendet. Es werden also Beiträge sowohl in deutscher als auch in englischer Sprache verfasst. Da sich diese beiden Sprachen in ihrer Syntax und Semantik für die rechnergestütze Verarbeitung zu sehr unterscheiden, ist es nötig, die Beiträge jeweils sprachenabhängig im Preprocessing zu behandeln. Um dies gewährleisten zu können, wird eine Identifikation der Sprache des jeweiligen Beitrags mithilfe der Library language-detector durchgeführt (Zeile 7). Die identifizierte Sprache wird für den weiteren Gebrauch in einer Variablen gespeichert. Die sprachenabhängigen Operationen beginnen mit der Abfrage der erkannten Sprache in Zeile 8. Für deutschsprachige und englischsprachige Beiträge wird jeweils eine eigene Stopword Entfernung durchgeführt. Dies geschieht über zuvor festgelegte Sammlungen von vordefinierten Stopwords der jeweiligen Sprachen. Diese werden innerhalb der Klassen als Arrays gespeichert und für die weitere Verwendung in Sets umgewandelt. Sollte keine oder eine andere Sprache erkannt werden, so werden Stopwords anhand einer Liste entfernt, welche deutsche und englische Stopwords zusammenführt. Auch diese Liste wird in den Klassen als Array gespeichert. In diesem Fall kann es jedoch vorkommen, dass Wörter entfernt werden, die in der zweiten Sprache womöglich keinem Stopword entsprechen. Ein Beispiel dafür ist das Wort „die“. In der deutschen Sprache steht dieses Wort für einen Artikel und entspricht somit einem Stopword. In der englischen Sprache hingegen würde dieses Wort einem Verb mit einer für dessen Kontext wichtigen Bedeutung entsprechen.

Im fertigen Dashboard wird den Nutzern die Möglichkeit geboten, ein optionales Stemming der Wörter der Beiträge zu aktivieren. Die Abfragen in den Zeilen 9 und 17 prüfen, ob diese Wahl vom Nutzer getroffen wurde. Ähnlich wie bei der Stopword Entfernung, findet das Stemming sprachenabhängig statt. Für die Wörter von deutschen und englischen Beiträgen wird das Stemming mithilfe der Library snowballstem durchgeführt. Diese Library umfasst für beide Sprachen eigene Stem Listen, die auf dem Stemming Algorithmus von Porter basieren. Wird das Stemming aktiviert, so wird vor und nach dessen Durchführung eine Stopword Entfernung vorgenommen. Beiträge, denen keine oder eine andere Sprache zugeordnet wurden, werden nicht gestemmt.

Die nun fertig verarbeiteten Beiträge werden erneut in einer Liste gespeichert.

Die während des Preprocessings verwendeten Methoden stammen aus den Anwendungsgebieten des Natural Language Processings und der Information Extraction.

Identifizierung von populären Themen und Begriffen

Methode 1: Ermittlung der meist verwendeten Begriffe

Der einfachste Ansatz zur Analyse von populären Themen und Begriffen, ist die Ermittlung der meist verwendeten Begriffe innerhalb der Beiträge einer oder mehrerer Funktionen der vorhandenen Communities. Das dazu verwendete Verfahren wird in Quellcode 4-3 als Pseudocode gezeigt.

QC 4-3: Ablauf der Ermittlung der meist verwendeten Begriffe

```
1  ErstelleWortliste();
2  ErstelleHäufigkeitsverteilung();
3  SortiereMap(absteigend);
4         n Top Begriffe in Array speichern;
5         n Häufigkeiten in Array speichern;
6         100 Top Begriffe als String speichern;
```

Der konkret verwendete Code kann in Anhang 6 eingesehen werden.

Die Ermittlung der meist verwendeten Begriffe beginnt mit der Erstellung einer Wortliste. Diese umfasst sämtliche Wörter der Einträge der im Preprocessing erzeugten Liste. Zum Erhalt der Wortliste, wird über jeden Eintrag der Input-Liste iteriert und die einzelnen Wörter der Einträge der Wortliste hinzugefügt. Anschließend wird eine Häufigkeitsverteilung der Wörter erstellt. Dazu wird über jedes Wort der Wortliste iteriert und dieses mit einem Initialwert von eins in einer Map gespeichert. Sollte ein bereits der Map hinzugefügtes Wort in der Iteration auftreten, so wird dessen Wert in der Map um eins erhöht. Dieser Wert entspricht am Ende der Iteration der Häufigkeit des Auftretens des zugehörigen Worts. In einem weiteren Schritt wird die Map anhand der Häufigkeitswerte absteigend sortiert.

Umgesetzt wurde diese Methode in drei Java-Klassen mit jeweils unterschiedlichen Ausgaben. Geschuldet ist dies durch die Art der Einbindung der Klassen zur Visualisierung der Ergebnisse im Dashboard. Die erste Klasse gibt eine vom Nutzer des Dashboards festgelegte Anzahl von Begriffen der Map als Array aus. Die zweite Klasse gibt die zugehörigen Häufigkeitswerte der Begriffe in der gewählten Anzahl ebenfalls als Array aus. Beide Arrays werden dazu dienen, ein Balkendiagramm aus den Werten zu erstellen. Die dritte Klasse gibt einen String mit den ersten 100 Positionen der Map aus. Dieser String dient als Input für die Erstellung einer Wordcloud. Die genaue Einbindung der Klassen im Dashboard wird in Kapitel 5.3 erläutert.

Die nachfolgende Abbildung 4.1 zeigt ein beispielhaftes Säulendiagramm als Ergebnis der Ermittlung der zehn meist verwendeten Begriffe in der Blogs-Funktion der Community „[Deutsch] UniConnect Community“. Diese dient als Einstiegscommunity für alle Mitglieder der Plattform zum Ersuchen von Hilfe und als Informationskanal des UCT. Eine Visualisierung derselben Ergebnisse als Wordcloud ist in Abbildung 4.2 dargestellt. Durch die Schriftgröße und Schriftfarbe der gezeigten Begriffe, werden die Häufigkeiten dieser repräsentiert. Je größer und farblich auffälliger ein Begriff darstellt ist, desto häufiger findet dieser Erwähnung.

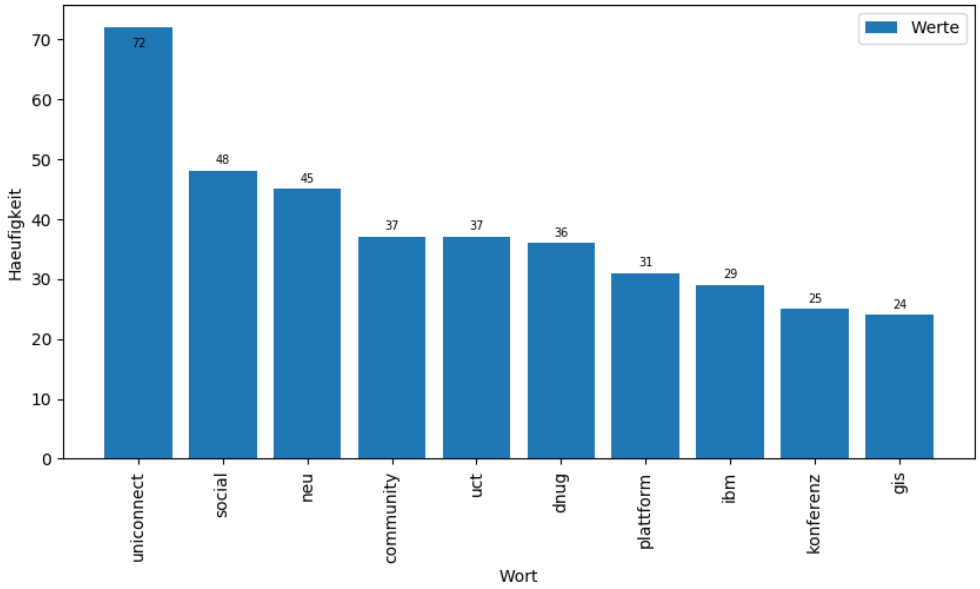

Abb. 4.1: Säulendiagramm als Ergebnis der Ermittlung der meist verwendeten Begriffe

Abb. 4.2: Wordcloud als Ergebnis der Ermittlung der meist verwendeten Begriffe

Anhand beider Visualisierungen ist zu erkennen, dass die meist verwendeten Begriffe der oben genannten Community im Zusammenhang mit der Plattform selbst („uniconnect“, „plattform“, „anwendung“, „community“) stehen oder aus dem Umfeld des UCT („uct“, „gis“, „dnug“, „ibm“) stammen. Mögliche Themenbereiche, die in dieser Community Erwähnung finden, sind somit die Plattform selbst und Informationen seitens des Betreibers.

Methode 2: Ermittlung von n-Grams

Ein zweiter Ansatz ist die Ermittlung von sogenannten n-Grams (deutsch: n-Gramme). Diese beschreiben Wortketten, bestehend aus n Wörtern ($n \in \mathbb{N}$), die besonders häufig gemeinsam in einem Dokument auftreten. Die Ermittlung der am häufigsten auftretenden n-Grams kann mit Java-eigenen Mitteln durchgeführt werden. Quellcode 4-4 zeigt den dazu entsprechenden Pseudocode. Auch hier dient die aus dem Preprocessing gewonnene Liste als Input. Die ermittelten n-Grams werden mit ihren zugehörigen Häufigkeiten als HTML-Tabelle in Form eines Strings ausgegeben.

QC 4-4: Ablauf der Ermittlung der häufigsten n-Grams

```
1 ErstelleString();
2 BildeAlleNGramsUndSortiere(n);
3 ErstelleHäufigkeitsverteilung();
4 SortiereMap(absteigend);
5 HTML Tabelle als String speichern;
```

Der konkret verwendete Code kann in Anhang 7 eingesehen werden.

Zunächst werden alle Einträge der resultierenden Liste des Preprocessings zu einem String konkateniert. Aus diesem String werden sämtliche formbare n-Grams der vom Nutzer bestimmten Länge ermittelt. Dabei kann es vorkommen, dass bestimme n-Grams mehrfach oder permutiert auftreten. Aus diesem Grund werden die Wörter jedes n-Grams aufsteigend sortiert, sodass Permutationen gelöst und mehrfach auftretende Sequenzen entfernt werden können. Für jedes der verbliebenen n-Grams wird anschließend ermittelt, wie häufig es in dem erstellten String auftaucht. Ähnlich wie bei der vorherigen Methode, werden auch hier die n-Grams mit ihren zugehören Häufigkeiten in einer Map gespeichert und anhand der Häufigkeitswerte absteigend sortiert. Eine vom Nutzer festgelegte Anzahl von Einträgen wird dann als HTML Tabelle in Form eines Strings ausgegeben. Diese Tabelle wird mithilfe von Methoden einer externen Java-Klasse[12] erzeugt.

Beispiele für die fünf häufigsten 2-grams (auch bigrams genannt, deutsch: Bigramme) und 3-grams (auch trigrams genannt, deutsch: Trigramme) der Blogeinträge der zuvor betrachteten Community „[Deutsch] UniConnect Community“ zeigen die nachfolgenden Tabellen 4-1 und 4-2.

Tab. 4-1: Übersicht der fünf häufigsten 2-grams der Community „[Deutsch] UniConnect Community“

ngrams	**count**
social medium	14
social business	13
uct team	12
ibm connections	12
uniconnect community	11

[12] https://gist.github.com/2sbsbsb/2951464

Tab. 4-2: Übersicht der fünf häufigsten 3-grams der Community „[Deutsch] UniConnect Community“

ngrams	count
enterprise social software	7
english uniconnect community	4
abgeschlossen uniconnect verfuegbar	4
ibm connections version	4
wartung abgeschlossen uniconnect	4

Es lässt sich anhand der 3-grams feststellen, dass in der Community häufig über Wartungsarbeiten an der Plattform informiert wird. Weiterhin werden Begriffe aus dem Bereich der Social Software und dem Umfeld der Plattform verwendet. Mögliche Themenbereiche sind somit „Wartungsarbeiten“, „UniConnect“ und „allgemeine Diskussion“.

Methode 3: Berechnung von TF/IDF Werten

Ebenfalls lassen sich wichtige Begriffe durch die Berechnung der jeweiligen TF/IDF Werte bestimmen. Wie in Kapitel 3.2 bereits erläutert, repräsentiert ein hoher Wert eine höhere Relevanz eines Worts für ein Dokument. Als Input für die Berechnung der Werte dient eine Liste von Listen (List<List<String>>). Der Inhalt der inneren Listen variiert je nach Auswahl der zu betrachtenden Funktion und der Anzahl der ausgewählten Communities. Die nachfolgende Tabelle 4-3 gibt eine Übersicht über die Aufteilung dieser Listen.

Tab. 4-3: Übersicht des Inhalts der inneren Listen bei der Berechnung von TF/IDF Werten

Anzahl der Communities	Betrachtete Funktion	Inhalt der inneren Listen
1	mindestens 2 Funktionen	pro Funktion eine Liste mit den zugehörigen Beiträgen
1	Blogs oder Forum oder Statusaktualisierungen oder Wikis	keine Berechnung von TF/IDF-Werten möglich, durch fehlende Vergleichslisten
mehr als 1	mindestens 2 Funktionen	pro Community eine Liste mit den Beiträgen der gewählten Funktionen
mehr als 1	Blogs oder Forum oder Statusaktualisierungen oder Wikis	pro Community eine Liste mit den Beiträgen der gewählten Funktion

Die Einträge der inneren Listen werden jeweils zu einzelnen Strings konkateniert, welche in einer weiteren Liste gespeichert werden. Für jedes Wort dieser Strings werden dann folgende Berechnungen durchgeführt:

1. Term Frequency (TF): Häufigkeit des Auftretens des Worts im zugehörigen String dividiert durch die Anzahl aller Wörter im String

2. Inverse Document Frequency (IDF): Logarithmus der Anzahl der Strings dividiert durch die Anzahl der weiteren Strings, in denen das Wort auftritt + 1. Dieser Wert repräsentiert die Gebräuchlichkeit des Worts.
3. TF/IDF: Produkt der Werte TF und IDF

Daraus ergibt sich pro String eine Liste der TF/IDF Werte pro Wort, welche gemäß der Werte absteigend geordnet ist. Der Nutzer bestimmt auch hier, wie viele Begriffe jeweils in Form von HTML Tabellen als String ausgegeben werden. In der folgenden Tabelle 4-4 wird das Ergebnis der TF/IDF Berechnungen für den Vergleich der Blogeinträge der Communities „[Deutsch] UniConnect Community", „[English] UniConnect Community" und „Betriebliche Anwendungssysteme 2017" gezeigt.

Tab. 4-4: Vergleich der TF/IDF Werte der Communities „[Deutsch] UniConnect Community", „[English] UniConnect Community" und „Betriebliche Anwendungssysteme 2017"

[Deutsch] UniConnect Community	
term	**tfidf-value**
social	0,00573
dnug	0,0043
konferenz	0,00299
gis	0,00287
datei	0,00251

[English] UniConnect Community	
term	**tfidf-value**
document	0,00538
dear	0,00493
unavailable	0,00358
temporarily	0,00314
folder	0,00314

Betriebliche Anwendungssysteme 2017	
term	**tfidf-value**
klausureinsicht	0,0065
zertifikat	0,0052
einsicht	0,0039
dienstags	0,0039
uebung	0,0039

Die jeweils zu den Communities gezeigten Begriffe mit den höchsten TF/IDF Werten spiegeln die Themen der Communities wider. So lässt sich anhand der Begriffe „Klausureinsicht" und „Übung" feststellen, dass das grundlegende Thema der Community „Betriebliche Anwendungssysteme 2017" eine Vorlesung ist. Die Begriffe der zwei weiteren gezeigten Communites deuten Communities zum Informationsaustausch bezüglich der Plattform an.

Alle bisher genannten Methoden zur Ermittlung von populären Themen entstammen aus dem Text Mining Anwendungsgebiet der Information Extraction. Im Folgenden werden zwei Methoden aus den Gebieten Document Classification und Document Clustering zur Klassifikation von Dokumenten vorgestellt (siehe Kapitel 3.3). Beide Methoden implementieren zudem Mechanismen des Machine Learnings.

Methode 4: Bestimmung der Themen mittels eines selbsterstellten Classifiers

Die erste dieser zwei Methoden stammt aus dem Gebiet der Document Classification und basiert auf einem selbsterstellten und manuell gelabelten Datenset, mit dessen Hilfe ein auf Naive Bayes basierender Classifier erstellt wurde. Dieser ermöglicht es automatisiert Beiträgen ein Thema zuzuordnen. Im Umfeld des Machine Learnings wird dieser Ansatz zum Supervised Machine Learning (überwachtes ML) gezählt, da die Themen zuvor bekannt sind (Shah & Mahajan, 2012).

Zur Erstellung des Datensets wurden zunächst Beiträge der vier ausgewählten Funktionen von 13 Communities zwischengespeichert und anschließend manuell mit einem Label des zugehörigen Themas versehen. Dabei wurden Beiträge entfernt, die keinem der Themen zugeordnet werden konnten oder keine Relevanz in Bezug auf ihren Inhalt aufwiesen. Die Einteilung der Beiträge erfolgte in zehn Themen, die durch eine vorhergehende nähere Analyse der Beiträge festgestellt werden konnten. Eine erläuterte Auflistung dieser zeigt die nachfolgende Tabelle 4-5.

Tab. 4-5: Vordefinierte Themen des Classifiers

Label / Thema	**Erläuterung**
discussion	Diskussionen ohne festgelegtem Thema
event	Beiträge mit Bezug zu bevorstehenden oder zurückliegenden Veranstaltungen
external	Beiträge ohne Bezug zu Forschung & Lehre („off-topic“)
lecture	Beiträge mit Bezug zu einer Vorlesung
maintenance	Ankündigungen zu Wartungsarbeiten oder sonstigen Auszeiten der Plattform
people	Beiträge mit persönlichem Bezug zu Mitgliedern der Plattform
project	Beiträge mit Bezug zu Projektarbeiten
support	hilfeleistende Beiträge
uct	Bekanntgaben des UCT
uniconnect	Beiträge mit Bezug zu UniConnect

Nach Abschluss des manuellen Labelns und der Entfernung irrelevanter Beiträge, verblieben 384 mit einem Label versehene Beiträge. Die vergebenen Label wurden, getrennt durch ein Komma, an das Ende der zugehörigen Beiträge angehängt und in einer separaten Klasse als Array gespeichert. Dieses Datenset dient als Trainings-Datenset für einen Naive Bayes Algorithmus, der mithilfe der Java Library Java Naive Bayes Classifier[13] umgesetzt wurde. Dieser erstellt mittels probabilistischer Methoden den Classifier, welcher anschließend in einer Variablen zwischengespeichert wird. Er wird somit bei jedem Aufruf der Klasse erneut erstellt. Dies geschieht jedoch ohne signifikante zeitliche Verzögerungen. Eine Veranschaulichung der exakten Funktionsweise des Naive Bayes Algorithmus bietet (Rish, 2001).

[13] https://github.com/ptnplanet/Java-Naive-Bayes-Classifier

Zum Identifizieren der Themen, die innerhalb einer Community besprochen werden, genügt es dann, die ausgewählten Beiträge mit dem erstellten Classifier abzugleichen. Aus den erhaltenen Themen wird dann eine geordnete Häufigkeitsverteilung erstellt, die als HTML Tabelle in Form eines Strings ausgegeben wird. Aus dieser kann das übergeordnete Thema der Community erfasst werden. Die nachfolgende Tabelle 4-6 zeigt die Häufigkeitsverteilung der identifizierten Themen der Community „[Deutsch] UniConnect Community".

Tab. 4-6: Identifizierte Themen der Community „[Deutsch] UniConnect Community"

Thema	**Häufigkeit**
maintenance	198
discussion	15
event	12
uniconnect	12

Wie die Tabelle zeigt, konnten vier Themen durch den Classifier in der Community „[Deutsch] UniConnect Community" festgestellt werden. So scheint diese Community hauptsächlich zum Kommunizieren von Wartungsarbeiten („Maintenance") verwendet zu werden. Ebenfalls dient diese Community zur allgemeinen Diskussion unter den Mitgliedern („Discussion") sowie zum Teilen von Informationen bezüglich Veranstaltungen („Event") oder der Plattform („UniConnect").

Methode 5: Bestimmung der Themen mittels Document Clustering

Die zweite Methode stammt aus dem Gebiet des Document Clusterings und implementiert ein sogenanntes Topic Model basierend auf der Latent Dirichlet Allocation (LDA). Topic Models beschreiben Algorithmen, die in einem Dokument verstreute Themen identifizieren (Blei, 2012). Jedes dieser Themen wird durch eine bestimmte Anzahl von Wörtern beschrieben (Steyvers & Griffiths, n.d.). Diese Wörter bilden somit Cluster der zugehörigen Themen. Im Umfeld des ML ist diese Methode im Bereich des Unsupervised Machine Learning (unüberwachtes ML) angesiedelt, da die zu identifizierenden Themen zuvor nicht bekannt sind (Shah & Mahajan, 2012). Implementiert wurde der Algorithmus mithilfe der Java Library MALLET[14]. Als Datengrundlage dient ebenfalls die aus dem Preprocessing resultierende Liste. Die Ausgabe ist eine HTML Tabelle als String. Zur korrekten Funktionsweise wird aus diesen Beiträgen eine Term Document Matrix erstellt. Dies beschreibt eine Matrix, dessen Spalten die Dokumente (in diesem Fall die Beiträge) enthalten, wohingegen sämtliche in den Dokumenten auftretenden Begriffe in den Zeilen aufgefunden werden können (Rishel, Perkins, Yenduri, Zand, & Iyengar, 2006). Die Zelleninhalte beschreiben die Auftrittshäufigkeit eines Begriffes innerhalb eines Dokuments (Rishel et al., 2006). Einen genauen Überblick über die Funktionsweise eines LDA Algorithmus bietet (Blei, Ng, & Jordan, 2003).

[14] http://mallet.cs.umass.edu

Die nachfolgende Tabelle 4-7 zeigt die identifizierten Cluster der bereits zuvor mehrfach betrachteten Community „[Deutsch] UniConnect Community“.

Tab. 4-7: Identifizierte Cluster der Community „[Deutsch] UniConnect Community“

Thema	Cluster
1	geplante, abgeschlossen, wartung, verfuegbar, uniconnect, verfuegung, verstaendnis, stehen, vorfeld, ausfaelle
2	flemming, johannes, goetz, christoph, team, jan, support, hinweis, fehler, uct
3	bestaetigen, funktionieren, fehler, umfragen, sebastian, marcel, hinweis, behoben, geloest, umfrage
4	dnug, social, konferenz, stand, besucher, vertreten, business, informieren, uct, collaboration
5	community, ibm, uniconnect, dateien, connections, funktionen, migration, gis, plattform, anwendungen
6	mesz, voruebergehend, verfuegbar, uniconnect, freundlichen, team, gruessen, montag, uct, dienstag
7	plattform, bitten, wartungsarbeiten, freundlichen, laeuft, gruessen, ludwig, paulsen, erreichbar, nutzer
8	funktioniert, loesung, browser, schnelle, chrome, firefox, helfen, hilfe, safari, erkennt

Die Tabelle zeigt, dass acht Cluster identifiziert werden konnten. Vier dieser Cluster (1, 5, 6, 7) thematisieren Wartungsarbeiten der Plattform. Drei weitere Cluster (2, 3, 8) thematisieren Hilfestellungen durch das UCT. Cluster vier behandelt eine anstehende beziehungsweise bereits vergangene Veranstaltung.

Sentiment Analyse als Zusatzfunktion

Als Zusatzfunktion wurde eine einfache Sentiment Analyse (Stimmungserkennung) implementiert. Diese aus dem Anwendungsgebiet der Concept Extraction stammende Analyse prüft, ob die Haltung eines Beitrags positiv, neutral oder negativ ist. Die Datengrundlage zur Analyse ist erneut die aus dem Preprocessing resultierende Liste der Beiträge. Zudem werden vordefinierte Listen mit Wörtern benötigt, die eine positive oder negative Haltung ausdrücken. Analog zur Stopword Erkennung, werden diese jeweils in deutsche und englische Begriffe unterteilt. Hier tritt eine ähnliche Problematik durch die Bilingualität der Plattform ein, die jedoch mithilfe der gleichen Vorgehensweisen gelöst werden konnte. Um nun die Haltung eines einzelnen Beitrags zu überprüfen, prüft der Algorithmus für jedes Wort, ob es in einer der Listen vorkommt. Kommt es in keiner der Listen vor, so gilt es als neutral. Überwiegt in einem Beitrag die Menge an positiven Begriffen, so gilt der ganze Beitrag als positiv (analog für negative Begriffe). Ist die Anzahl an positiven und negativen Begriffen ausgeglichen, so ist der Beitrag neutral. Das Ergebnis der Analyse ist ein Array, welches die Anzahl der zugeordneten Beiträge der drei Kategorien enthält. Die Abbildung 4.3 zeigt ein beispielhaftes Kreisdiagramm für die Community „[Deutsch] UniConnect Community“.

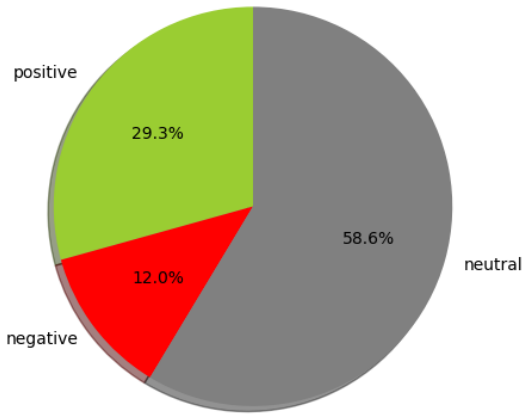

Abb. 4.3: Verteilung der Haltungen der Beiträge der Community „[Deutsch] UniConnect Community“

Die Verteilung der Haltungen zeigt, dass der überwiegende Teil der Beiträge der analysierten Community eine neutrale Haltung besitzt. Lediglich 41,3 % der Beiträge weisen eine positive oder negative Haltung auf. Neutrale Beiträge lassen sich in dieser Community auf Ankündigungen zu Wartungsarbeiten oder Informationen des UCT zurückführen. Problemberichte von Nutzern können zu Beiträgen mit negativer Haltung geführt haben. Berichte über gelöste Probleme lassen wiederum positive Haltungen vermuten.

Zusätzlich zu den oben genannten Funktionen, bieten die Klassen die Möglichkeiten, nur Beiträge einer bestimmten Funktion oder eines bestimmten Zeitraums zu analysieren. Ebenfalls besteht die Option, Antworten beziehungsweise Kommentare nicht mit in die Analyse einzubeziehen.

Die Java-Klassen dienen nun als Basis des Dashboard-Prototyps, der im folgenden Kapitel vorgestellt wird. Der Quellcode kann in einem Gitlab Repository eingesehen werden. Der Link dazu befindet sich in Anhang 8.

5 Das Dashboard

Analog zur CRISP-DM Phase Modeling, dient dieses Kapitel zur Erläuterung des Dashboard-Prototyps. Dazu werden zunächst Informationen zu der zugrundeliegenden Programmierumgebung XPages und den gewünschten Funktionen des Dashboards vermittelt. Unterstützt wird dies durch Mockups. Ein Überblick über die genaue Umsetzung folgt in Kapitel 5.3.

5.1 XPages

Als Programmiergrundlage für das Dashboard dient das Framework XPages. Dabei handelt es sich um ein Framework für die Entwicklung von Webanwendungen für IBM Notes und IBM Domino, welches momentan in der Version 9.0.1[15] vorliegt (Donnelly, McGuckin, & Wallace, 2011). Es umfasst zudem eine Java Laufzeitumgebung, die es ermöglicht alle wichtigen Features des Web 2.0 zu implementieren (Donnelly et al., 2011). Gleichzeitig erweitert es das zur Entwicklung von GUIs verwendete Standard-Framework JavaServer Faces (JSF) (Donnelly et al., 2011). Entwickelt werden XPages in der auf Eclipse basierenden und für den Einsatzzweck angepassten Entwicklungsumgebung IBM Domino Designer. In dessen WYSIWYG[16]-Editor können XPages mittels Drag-and-Drop oder durch die kombinierte Nutzung von CSS, HTML und JavaScript erstellt und anschließend im Webbrowser abgerufen werden (Donnelly et al., 2011).

5.2 Grundlagen

Dashboards beschreiben im Allgemeinen mehrschichtige Systeme zum Leistungsmanagement, errichtet auf einer Business Intelligence- und Datenintegratrions-Infrastruktur (Watson, 2006). Sie ermöglichen es Unternehmen, geschäftliche Aktivitäten durch die Nutzung von finanziellen und nicht-finanziellen Maßeinheiten zu messen, zu überwachen und zu leiten (Watson, 2006). Im Falle dieser Arbeit dient es als grafische Oberfläche für die Bedienung der erstellten Java-Klassen und der Anzeige der Analyseergebnisse.

Folgende Funktionen soll das Dashboard beherrschen:

- Möglichkeit zur Eingabe der Benutzerdaten zur Authentifizierung gegenüber der UniConnect Datenbanken
- Auswahl der zu analysierenden Community oder Communities
- Setzen von Filtern, wie beispielsweise eine Zeitspanne oder eine maximale Anzahl an zu identifizierenden Begriffen
- verständliche und übersichtliche Anzeige der Analyseergebnisse

[15] Die Vergabe der Versionsnummer erfolgt analog zu der des Produktes IBM Notes.

[16] Akronym für “What you see is what you get”.

- einfache Bedienung

Um einen ersten Entwurf und Anhaltspunkt für die tatsächliche Umsetzung des Dashboards zu erhalten, wurden Mockups erstellt, die im Folgenden gezeigt werden. Anhand dieser kann zudem erläutert werden, wie die vorab genannten wünschenswerten Funktionen umgesetzt werden können. Zur Wahrung der Übersichtlichkeit und Einfacheit wird das Dashboard auf rechteckige und freisitzende Formen (Module) setzen. Die erfolgten Abweichungen bei der konkreten Umsetzung des Dashboards werden im nachfolgenden Kapitel erläutert.

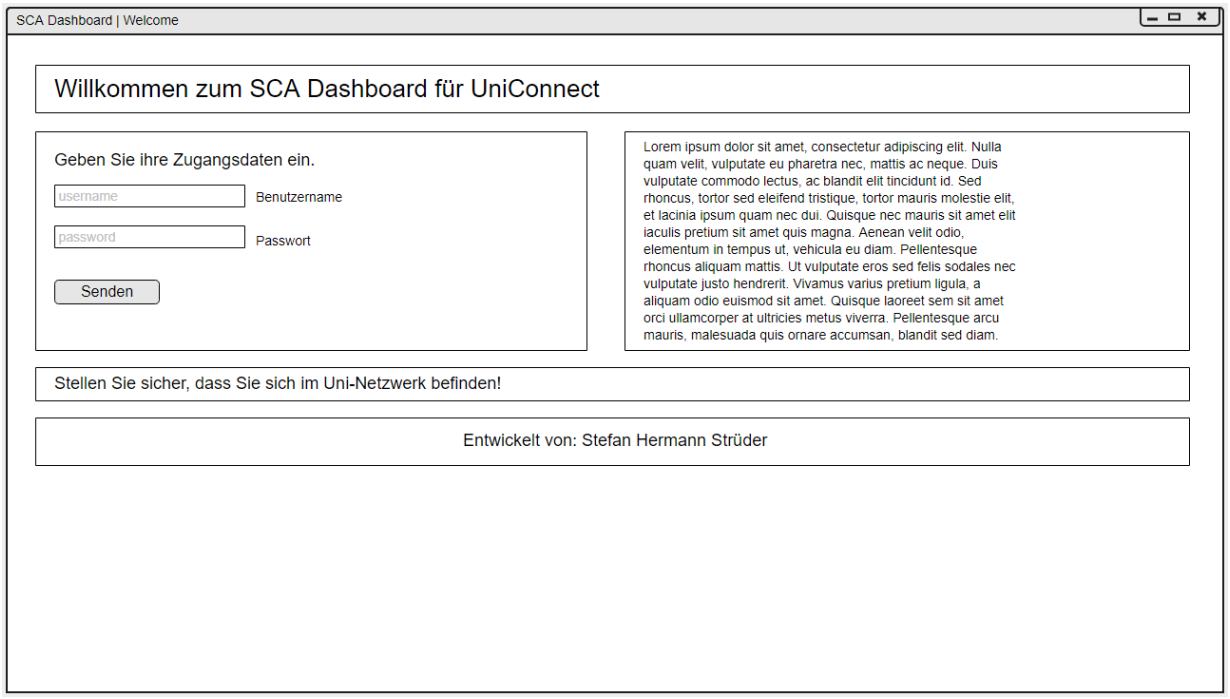

Abb. 5.1: Mockup des Begrüßungsbildschirms

Abbildung 5.1 zeigt das Mockup des Begrüßungsbildschirms. Dabei handelt es sich um die erste für den Nutzer sichtbare Seite des Dashboards. Hier soll die Möglichkeit bestehen, die Zugangsdaten zu den UniConnect Datenbanken einzugeben. Vergeben werden die Zugangsdaten an berechtigte Personen vom UCT. Darüber hinaus wird eine kurze Erläuterung des Dashboards und dessen Funktionen auf dieser Seite eingefügt. Nach Betätigung des „Senden“ Buttons, wird der Nutzer auf eine zweite Seite weitergeleitet. Diese ist in Abbildung 5.2 dargestellt.

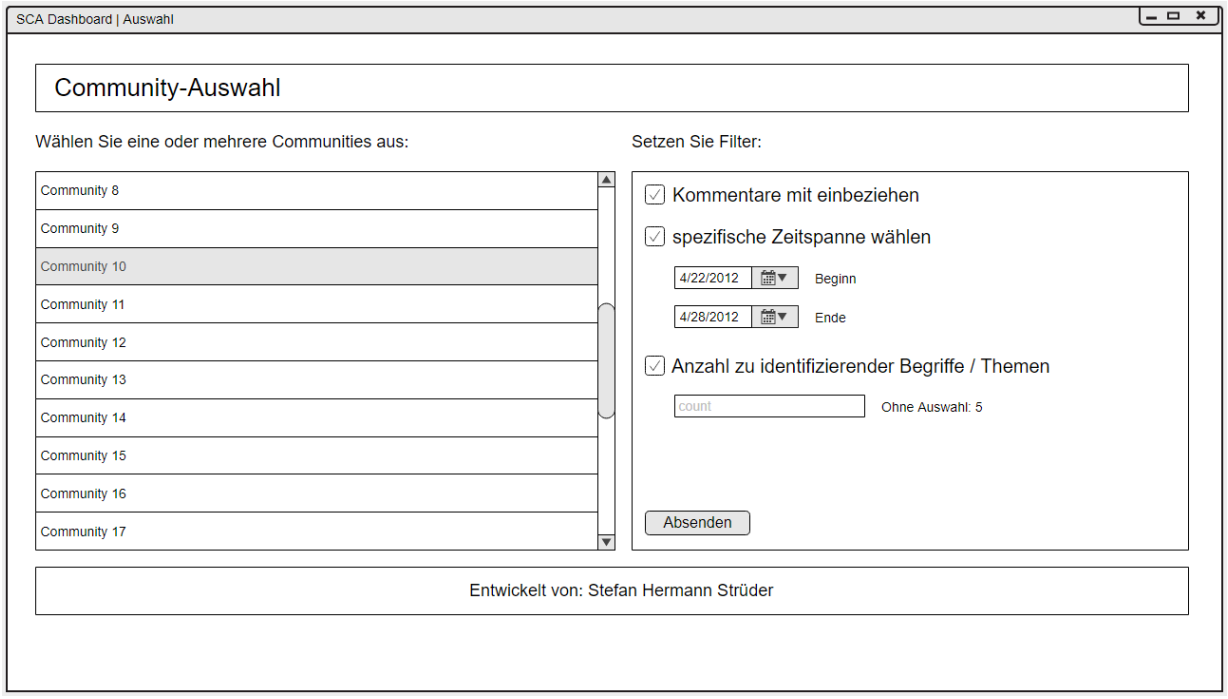

Abb. 5.2: Mockup des Bildschirms zur Communityauswahl

Auf der zweiten Seite findet die Auswahl der zu analysierenden Community oder Communities statt. Diese können aus einer Liste ausgewählt werden. Erzeugt wird diese Liste durch einen Datenbankabruf, der beim Laden der Seite ausgeführt wird. Zusätzlich können Filter gesetzt werden. Es ist mittels Checkboxen und Datepickern möglich Kommentare mit einzubeziehen und eine spezifische Zeitspanne auszuwählen. Weiterhin kann eine individuelle Anzahl an zu identifizierenden Begriffen oder Themen angegeben werden. Wird diese Option nicht ausgewählt, so beträgt der Wert fünf. Nach erfolgter Auswahl und Vergabe der Filter, wird mit einem Klick auf den „Absenden“ Button die Analyse gestartet. Diese verläuft im Hintergrund und kann zu einer geringen Wartezeit führen, ehe auf die dritte Seite des Dashboards weitergeleitet wird. Diese wird in Abbildung 5.3 dargestellt.

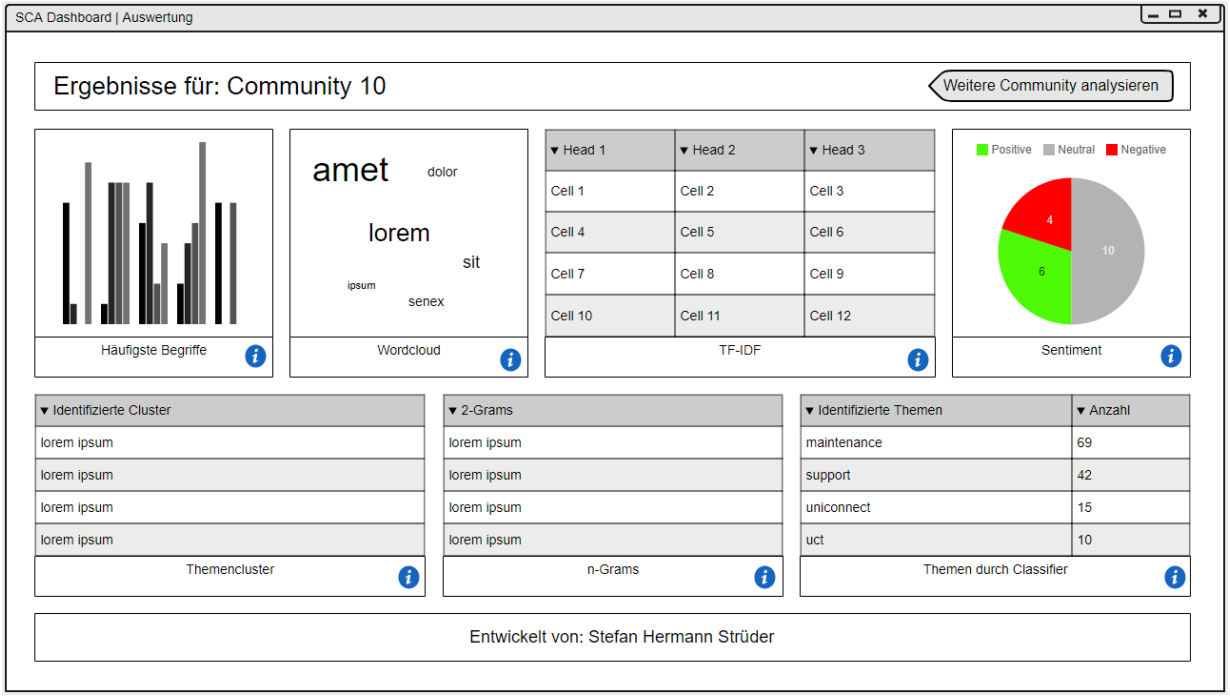

Abb. 5.3: Mockup des Ergebnisbildschirms

Die dritte Seite umfasst die Visualisierungen der Analyseergebnisse. Für jede in Kapitel 4.3 beschriebene Analysemethode existiert ein Modul, welches entweder ein Diagramm, eine Wordcloud oder eine Tabelle enthält. Über „Info“ Buttons können Erklärungen zu den jeweiligen Methoden aufgerufen werden. Um eine weitere Analyse durchzuführen, kann mithilfe eines „Zurück“ Buttons auf die vorherige Seite zurückgekehrt werden.

Wie das Dashboard konkret unter der Verwendung von XPages umgesetzt wurde, kann dem folgenden Kapitel entnommen werden.

5.3 Umsetzung

Die Umsetzung des Dashboards orientiert sich an dem im vorherigen Kapitel vorgestellten Konzept. Vorgenommene Änderungen, die zu Abweichungen des angedachten Designs und der Funktionalität führten, werden im weiteren Verlauf dieses Kapitels erwähnt.

Die grundlegende Farbgebung des Dashboards basiert auf drei Farben. Dunkelblaue Elemente bilden das Gerüst der drei Seiten des Dashboards. Sie werden für Header und Footer verwendet. Hellblaue Elemente beinhalten die Ergebnisse der Analysen und allgemeine Informationen sowie Kontrollelemente, die vom Nutzer zur Steuerung des Dashboards verwendet werden können. Rote Elemente dienen hingegen zur Anzeige von wichtigen Informationen. Sie werden für Hinweisfenster (siehe Abbildung 5.7) verwendet.

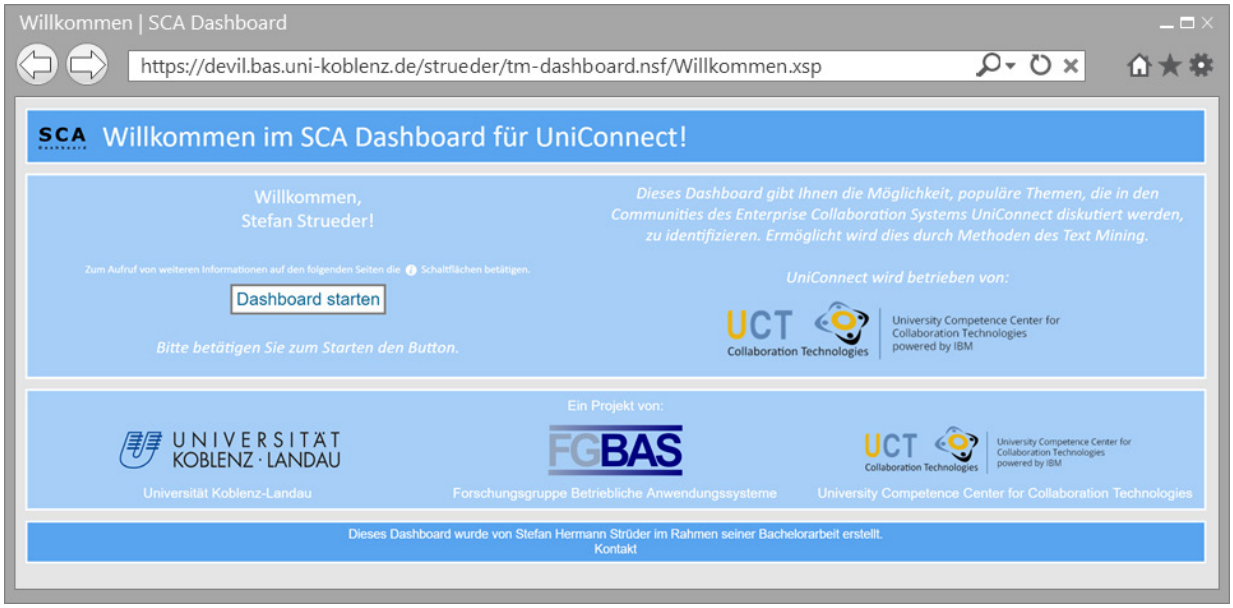

Abb. 5.4: Begrüßungsbildschirm des Dashboards

Die vorangegangene Abbildung 5.4 zeigt den finalen Begrüßungsbildschirm. Die auffälligste Änderung ist der Wegfall der Elemente zur Eingabe der Nutzerdaten. Bei Aufruf des Dashboards über den Webbrowser, erscheint automatisiert eine Landing-Page des Hosting-Servers, die zur Authentifizierung des Nutzers mittels Benutzernamen und Passwort auffordert und den Zugang zum Dashboard blockiert, sofern die Authentifizierung fehlschlägt. Ausgegeben werden diese Zugangsdaten an berechtigte Personen vom UCT. Ein Screenshot dieser Landing-Page ist in Abbildung 5.5 dargestellt. Eine weitere Authentifizierung im Dashboard selbst ist somit nicht nötig. Ersetzt wurden die Elemente durch einen Button, der bei Betätigung auf die zweite Seite des Dashboards weiterleitet. Diese Seite dient in erster Linie zur Informationsvermittlung bezüglich des Dashboards und den damit verbundenen Institutionen.

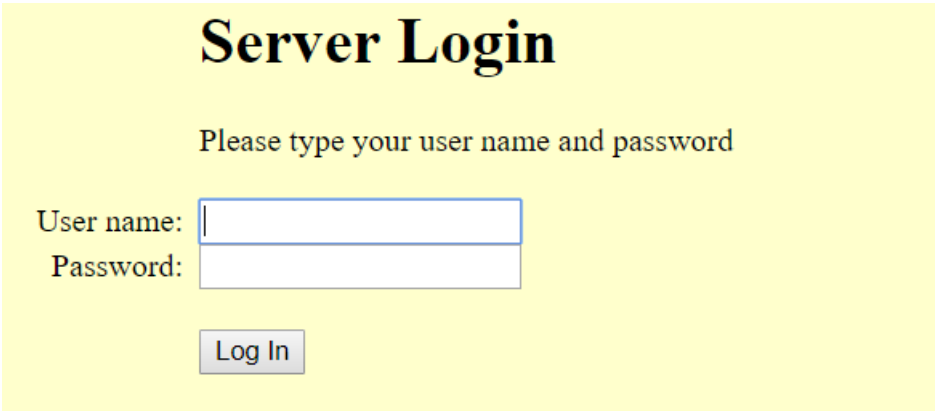

Abb. 5.5: Landing-Page des Hosting-Servers zur Authentifizierung

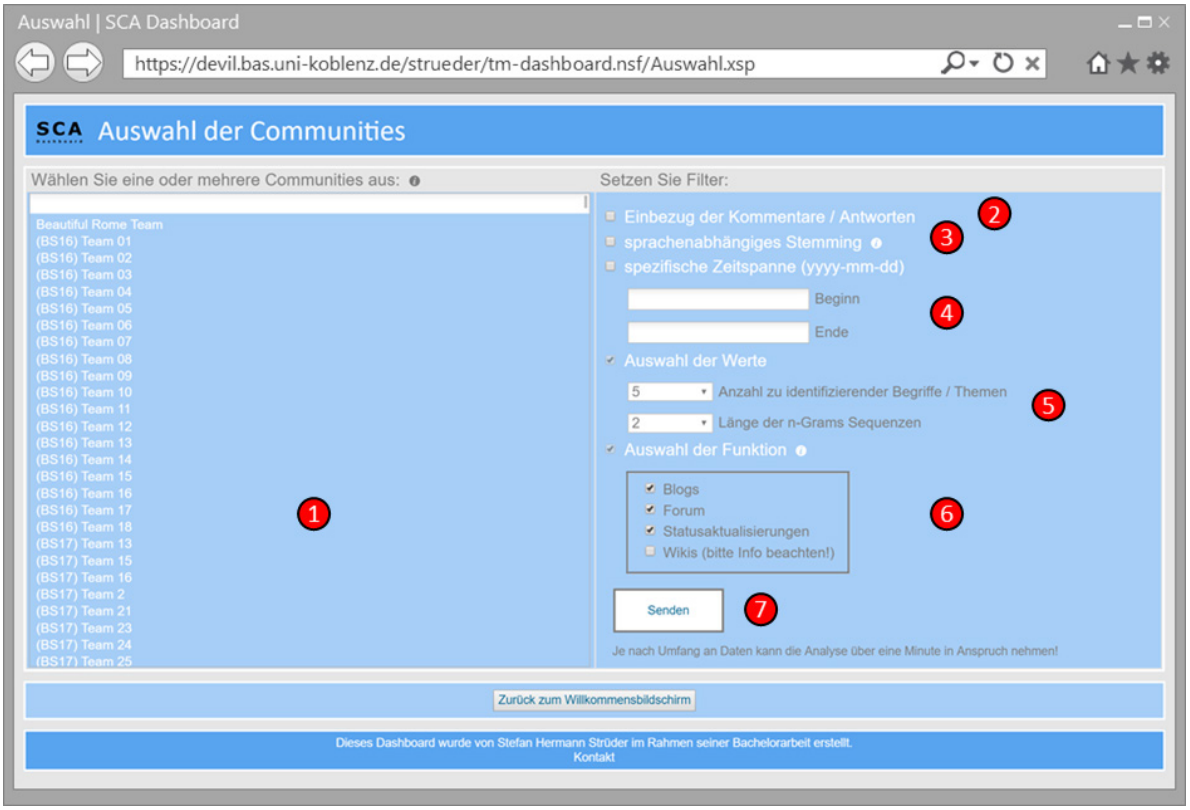

Abb. 5.6: Auswahlbildschirm des Dashboards

Abbildung 5.6 zeigt den finalen Auswahlbildschirm des Dashboards. Dieser stimmt weitestgehend mit dem im Mockup vorgestellten Modell überein. Hinzugekommen sind Info-Buttons ⓘ, welche Hinweisfenster zu den ihnen nebenstehenden Kontrollelementen anzeigen. Umgesetzt wurde dies mittels Dialog-Elementen der XPages Extension Library[17]. Das Hinweisfenster zum dritten Kontrollelement ist in Abbildung 5.7 dargestellt.

Ferner wurden einige Kontrollelemente zum Setzen weiterer Filter hinzugefügt. Nachfolgend werden die Kontrollelemente anhand der in Abbildung 5.6 festgelegten Nummerierung beschrieben.

Communityauswahl

Kontrollelement 1: Auswahlliste

In dieser Liste können die zu analysierenden Communities ausgewählt werden. Es besteht die Möglichkeit nur eine Community oder mittels der Strg-Taste mehrere Communities auszuwählen. Die Daten erhält die Liste über eine Methode (in sämtlichen Klassen „mainFunction()“ benannt) einer Java-Klasse, die beim Laden der Seite über ein serverseitigen JavaScript-Befehl aufgerufen wird. Dieser JavaScript-Befehl ist bei sämtlichen Aufrufen von Methoden aus Java-Klassen gleich aufgebaut und wird in Quellcode 5-1 gezeigt. Innerhalb der aufgerufenen Methode wird eine Datenbankabfrage an die

[17] https://extlib.openntf.org

UniConnect Datenbank *sncomm* durchgeführt, welche die Namen der vorhandenen Communities mit ihren zugehörigen UUIDs zurückgibt. Diese werden anschließend in einem Array in der Form [“community_name1|uuid1“, ...] gespeichert und an die Auswahlliste übergeben. Für den Nutzer sichtbar sind nur die Namen der Communities. Die zugehörigen UUIDs werden für den Nutzer verdeckt gespeichert. Die Auswahl einer oder mehrerer Communities erzeugt ein Array mit den zugehörigen UUIDs als Einträgen.

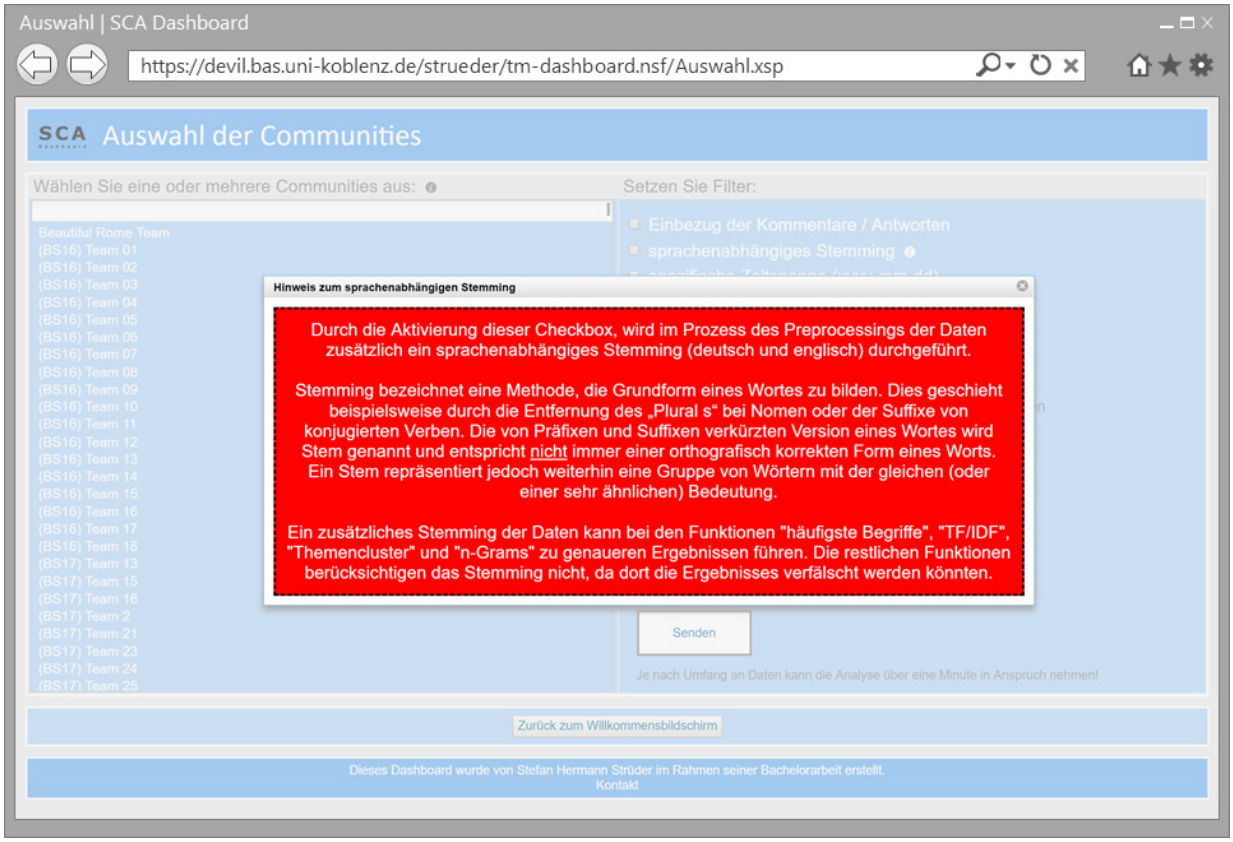

Abb. 5.7: Hinweisfenster zu einem Kontrollelement des Dashboards

QC 5-1: Serverseitiger JavaScript Aufruf einer Methode einer Java-Klasse

```
1  // Erzeugung einer neuen Instanz der Klasse
2  var v = new package.Klassenname();
3  // Aufruf der Methode mainFunction() der Klasse
4  v.mainFunction();
```

Filterauswahl

Kontrollelement 2: Checkbox

Dieses Kontrollelement bietet die Möglichkeit, bei Aktivierung, Kommentare beziehungsweise Antworten von Beiträgen in die Analyse mit einfließen zu lassen. Der Rückgabewert der Checkbox ist im Falle einer Aktivierung eine „1“. Ist diese Deaktiviert, so wird eine „0“ zurückgegeben. Dieser Wert beein-

flusst die Erzeugung der SQL-Abfragen in den Java-Klassen. Standardmäßig ist diese Checkbox deaktiviert.

Kontrollelement 3: Checkbox

Dieses neu hinzugefügte Kontrollelement widmet sich dem Preprocessing der Daten. Bei Aktivierung der Checkbox wird das Preprocessing in den Java-Klassen für die Ermittlung der meist verwendeten Begriffe, der Berechnung der TF/IDF Werte, der Identifizierung von Themenclustern und der Ermittlung der meist auftretenden n-Grams um ein sprachenabhängiges Stemming ergänzt (siehe Kapitel 4.2). Die restlichen Methoden übernehmen diese zusätzliche Funktion nicht, da durch das eventuelle Auftauchen von orthografisch nicht korrekten Stems Ergebnisse verfälscht werden könnten. Ein Klick auf den Info-Button klärt den Nutzer über diesen Umstand auf (siehe Abbildung 5.7). Auch dieses Kontrollfeld gibt entweder eine „0“ oder eine „1“ als Wert zurück. Dieser Wert beeinflusst das Preprocessing in den Java-Klassen. Standardmäßig ist diese Checkbox deaktiviert.

Kontrollelemente 4: Checkbox + Textfelder

Durch diese Elemente ist es möglich, die Analyse der Beiträge nur auf einen bestimmten Zeitraum einzugrenzen. Dieser kann vom Nutzer des Dashboards in den angegebenen Textfeldern im Format yyyy-mm-dd eingebeben werden. Damit diese Daten berücksichtigt werden, ist es erforderlich die zugehörige Checkbox zu aktivieren, um somit eine Übergabe der Strings an die Java-Klassen zu ermöglichen. Berücksichtigt werden die Eingaben der Textfelder und der Wert der Checkbox bei der Erzeugung der SQL-Abfragen. Standardmäßig ist diese Filterfunktion deaktiviert.

Kontrollelemente 5: Auswahlfelder

Mithilfe des ersten Auswahlfelds ist es möglich, die Anzahl der zu identifizierenden Themen / Begriffe, die von den Java-Klassen zurückgegeben werden, festzulegen. Erlaubt sind Werte von vier bis 20, standardmäßig voreingestellt ist der Wert fünf. Mit dem zweiten Auswahlfeld kann die Wortlänge der zu ermittelnden n-Grams („n“) festgelegt werden. Möglich ist eine Auswahl zwischen zwei und fünf. Voreingestellt ist der Wert zwei. Beide Werte beeinflussen in den Java-Klassen jeweils die Erzeugung derer Rückgabewerte.

Kontrollelement 6: Checkbox-Gruppe

Dieses neu hinzugefügte Kontrollelement in Form einer Checkbox-Gruppe gibt dem Nutzer die Möglichkeit die Quellen der zu analysierenden Beiträge auszuwählen. Zur Auswahl stehen die bereits mehrfach erwähnten Funktionen Blogs, Forum, Statusaktualisierungen und Wikis. Es besteht die Möglichkeit, eine oder mehrere Funktionen gleichzeitig zu betrachten. Standardmäßig vorausgewählt sind die Funktionen Blogs, Forum und Statusaktualisierungen. Der Rückgabewert der Checkbox-Gruppe, in Form eines Arrays, beeinflusst den Kontrollfluss der Java-Klassen. Ein Hinweisfenster, welches über den Info-Button angezeigt werden kann, informiert über den Umstand, dass nur verkürzte Versionen der Wikieinträge abgerufen werden können. Aus diesem Grund ist diese Funktion nicht standardmäßig vorausgewählt.

Kontrollelement 7: Button

Bei Betätigung des „Senden“-Buttons wird auf den Ergebnisbildschirm des Dashboards weitergeleitet.

Ergänzt wird der Auswahlbildschirm durch einen Button, der bei Betätigung auf den Willkommensbildschirm zurückleitet.

Die Analysen und Auswertungen werden beim Aufruf des Ergebnisbildschirms gestartet. Dabei werden die serverseitigen JavaScript-Befehle zum Ausführen der Methoden der Java-Klassen verrichtet. Da es nicht möglich ist, auf die Ausgabewerte der Kontrollelemente des Auswahlbildschirms zuzugreifen, da diese ohne Weiteres nicht seitenübergreifend weitergegeben werden können, wurde auf eine sogenannte Managed Bean als Datenobjekt zurückgegriffen. Dabei handelt es sich um eine JavaBean, die vom JSF Framework innerhalb des XPages Frameworks verwaltet wird (Donnelly et al., 2011). JavaBeans besitzen folgende Merkmale, die für die Wahl als Datenmodell für das Dashboard ausschlaggebend waren (Ullenboom, 2014):

1. Eigenschaften (Properties): JavaBeans steuern ihren Zustand durch Eigenschaften in Form von Attributen, die von außen verändert werden können.
2. Speicherung (Persistenz): Eine JavaBean kann ihre Eigenschaften durch Serialisierung speichern.
3. Selbstbeobachtung (Introspection): Eine JavaBean lässt sich von außen auslesen, um die Eigenschaften abzufragen.

Zusätzlich verfügt eine Managed Bean über einen festlegbaren Lebenszyklus („Scope“), der es ermöglicht, die Bean über die gesamte Zeit der Verwendung des Dashboards durch den Nutzer („Session“) zugreifbar zu machen. Übertragen auf die Anwendung der Managed Bean im Dashboard bedeutet dies, dass für jedes Kontrollelement des Auswahlbildschirms eine Eigenschaft innerhalb der Bean zur Verfügung steht, welche durch Getter und Setter von außen ausgelesen und verändert werden kann. Die Kontrollelemente besitzen jeweils einen serverseitigen JavaScript-Befehl, der bei jeder vorgenommenen Änderung den Rückgabewert an die entsprechende Eigenschaft der Bean überträgt. Die Bean speichert diesen Wert anschließend über die gesamte Zeit der Verwendung des Dashboards. Somit ist es auch möglich, aufgrund des Merkmals der Selbstbeobachtung, seitenübergreifend auf die Rückgabewerte zuzugreifen. Beim Aufruf der Methoden der Java-Klassen, werden die benötigten Eigenschaften der Bean zur weiteren Verwendung, durch Aufruf des zugehörigen Getters abgefragt. Der Quellcode der erstellten Managed Bean kann in Anhang 9 eingesehen werden.

Damit eine Managed Bean verwendet werden kann, muss sie im XPages Framework registriert werden. Dies geschieht durch einen Eintrag in der XML-Datei faces-config.xml. Der nachfolgende Quellcode 5-2 zeigt diesen Eintrag. Eine Erklärung folgt im Anschluss.

QC 5-2: Eintrag der Managed Bean in der Datei faces-config.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <faces-config>
3    <managed-bean>
4      <managed-bean-name>exportBean</managed-bean-name>
5      <managed-bean-class>query.Bean</managed-bean-class>
6      <managed-bean-scope>session</managed-bean-scope>
7    </managed-bean>
8    <!--AUTOGEN-START-BUILDER: Automatically generated by IBM Domino Designer. Do
   not modify.-->
9    <!--AUTOGEN-END-BUILDER: End of automatically generated section-->
10 </faces-config>
```

Der Eintrag der Managed Bean befindet sich in den Zeilen 3 – 7. Der Name in Zeile 4 dient zur Referenzierung der Bean in den serverseitigen JavaScript-Befehlen. Die Namen des Packages und der Java-Klasse, in welcher die Bean implementiert wurde, werden in Zeile 5 angeben. Zeile 6 wiederum dient zur Festlegung des Lebenszyklus der Bean. Im oben gezeigten Eintrag bedeutet „session“, dass die Bean zur gesamten Zeit der Nutzung des Dashboards verfügbar ist.

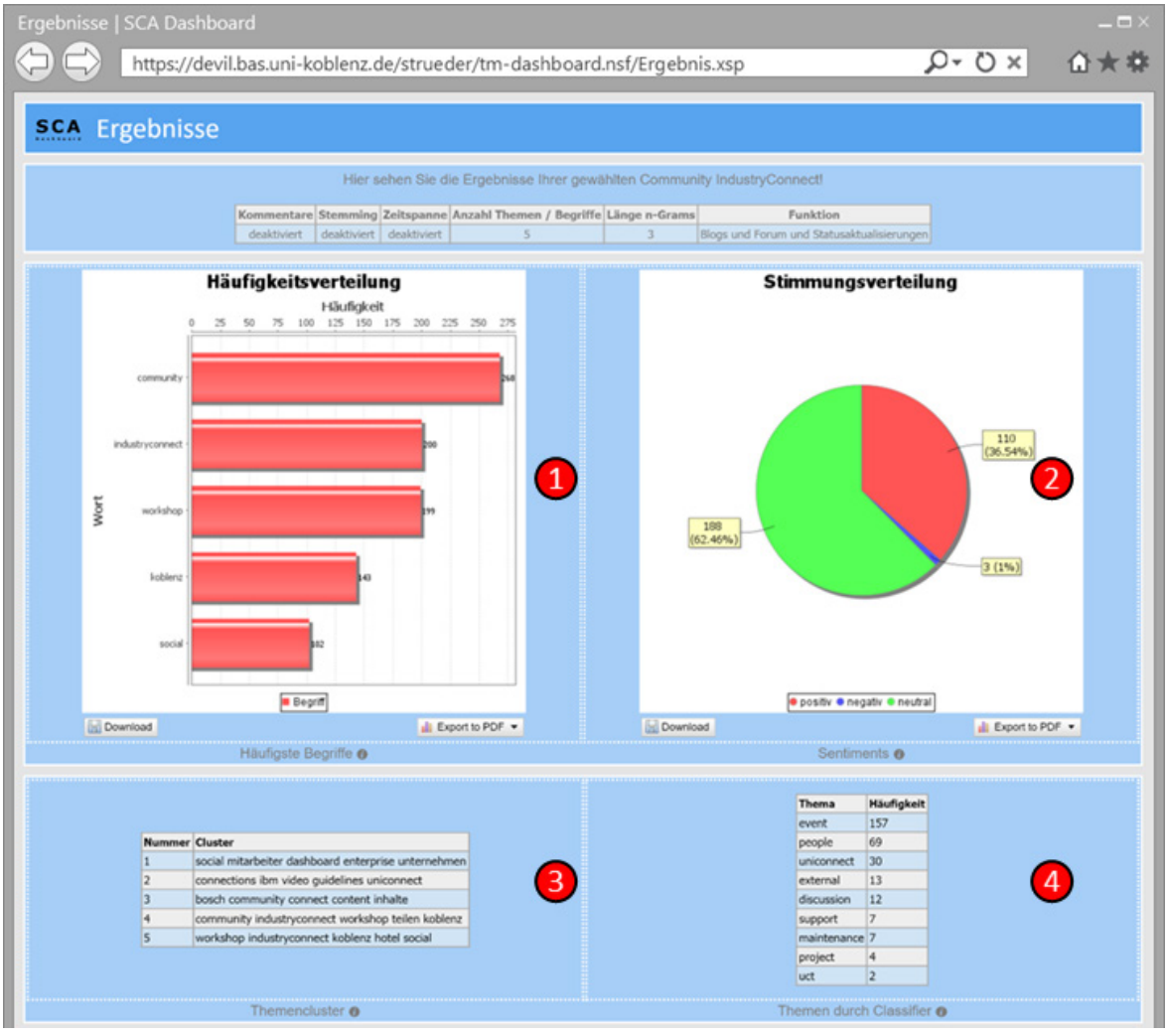

Abb. 5.8: Obere Hälfte des Ergebnisbildschirms zur Analyse der Community „IndustryConnect“

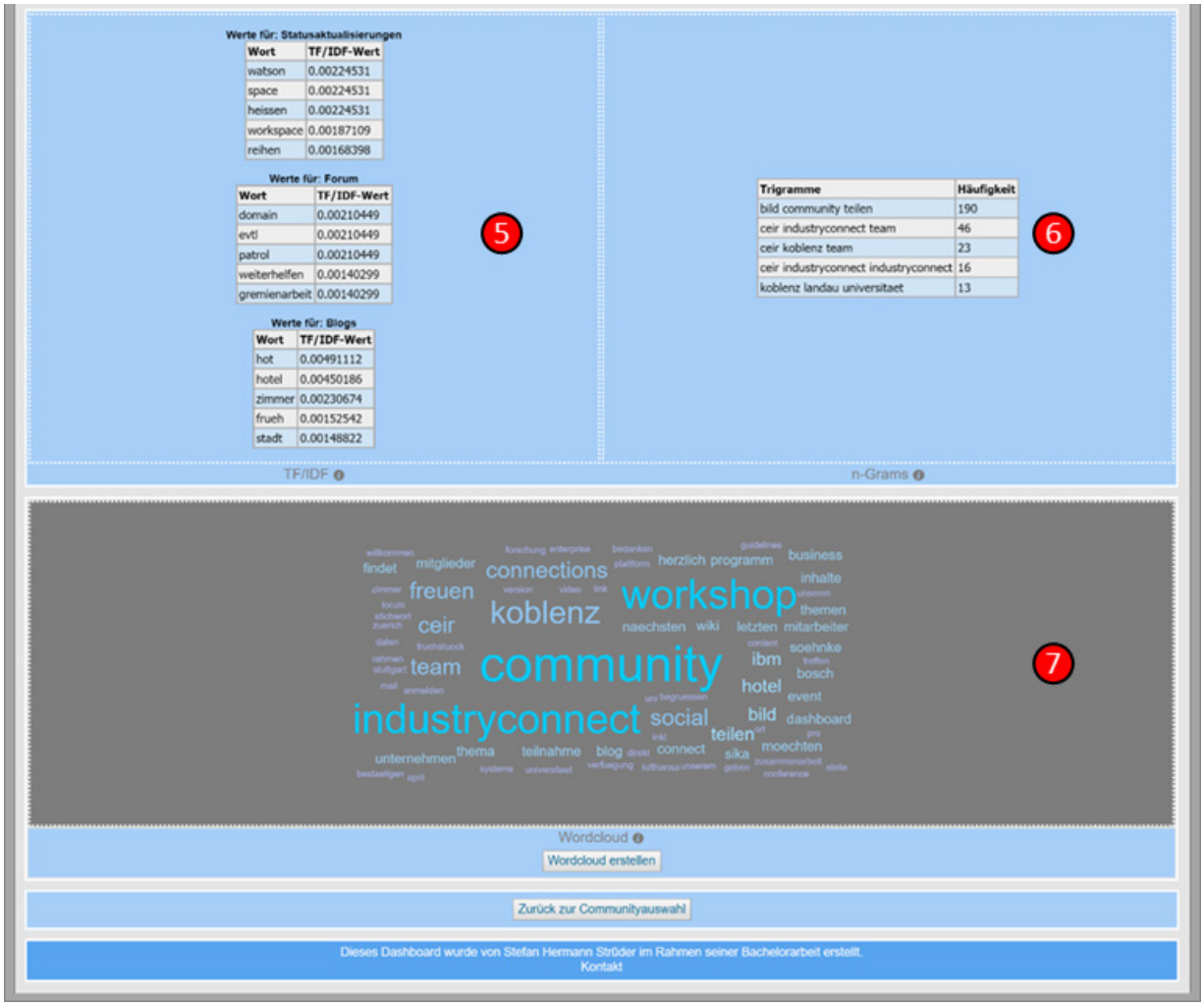

Abb. 5.9: Untere Hälfte des Ergebnisbildschirms zur Analyse der Community „IndustryConnect“

Der Ergebnisbildschirm hat die im Mockup vorgestellte Ausrichtung der Ergebnismodule nicht übernommen. Aus Gründen der Übersichtlichkeit wurden jeweils nur ein beziehungsweise zwei Module nebeneinander ausgerichtet. Die vorangegangenen Abbildungen 5.8 und 5.9 zeigen die obere und untere Hälfte des Ergebnisbildschirms mit der Auswertung der Analyse der Community „IndustryConnect“. Einen Überblick über die gesetzten Filter gibt die nachfolgende Tabelle 5.1. Eine Erläuterung der in den Abbildungen markierten Module erfolgt im Anschluss.

Tab. 5-1: Auflistung der gesetzten Filter bei der Analyse der Community „IndustryConnect“

Einbezug der Kommentare	**sprachenabhängiges Stemming**	**spezifische Zeitspanne**	**Anzahl Begriffe / Themen**	**Länge der n-Grams**	**ausgewählte Funktion**
deaktiviert	deaktiviert	deaktiviert	5	3	Blogs, Forum, Status-aktualisierungen

1. Modul: Häufigste Begriffe

Als Darstellungsform des Ergebnisses der Ermittlung der meist verwendeten Begriffe wurde ein Balkendiagramm gewählt. Auf der Y-Achse befinden sich die Begriffe. Die zugehörigen Häufigkeiten werden auf der X-Achse angezeigt. Erzeugt wird das Diagramm mithilfe eines Custom Controls des Plugins Java Charts[18]. Dieses erhält als Eingabedaten die von den zugehörigen Java-Klassen ausgegeben Arrays, die die Begriffe und Häufigkeiten beinhalten. Ausgeführt werden die Methoden der Klassen durch einen serverseitigen JavaScript Befehl. Über Schaltflächen unterhalb des Diagramms kann dieses als Bild- oder PDF-Datei heruntergeladen werden.

2. Modul: Sentiments

Für dieses Modul wurde als Darstellungsform der Ergebnisse ein Kreisdiagramm gewählt. Dieses zeigt in prozentualen und absoluten Werten die Aufteilung der Haltungen an. Erzeugt wird das Diagramm ebenfalls mithilfe des Plugins Java Charts. Das zugehörige Custom Control führt einen serverseitigen JavaScript Befehl zum Ausführen der Methode der zugehörigen Java-Klasse aus und erhält somit das von der Methode zurückgegebene Array als Eingabewert. Auch hier besteht die Möglichkeit das Diagramm über entsprechende Schaltflächen herunterzuladen.

Module 3 bis 6:

Wie in der Erläuterung der zugehörigen Java-Klassen bereits erwähnt wurde, ist der Rückgabewert der Methoden dieser jeweils eine HTML Tabelle in Form eines Strings. Zum Einsatz kommt hier pro Modul ein Computed Field, welches den String, als Rückgabewert der Ausführung des JavaScript Befehls zum Aufruf der Methode, als HTML Tabelle darstellen kann. Über ein CSS-Stylesheet wird das Design der Tabellen angepasst.

Modul 7: Wordcloud

Zur Erzeugung der Wordcloud wird das clientseitige jQuery Plugin jQCloud[19] verwendet. Zum Erhalt der Wortliste als String aus der Methode der entsprechenden Java-Klasse, wird ein Remote Service (auch Remote Procedure Call genannt) aufgerufen. Dieser ermöglicht es, serverseitige Befehle über die Client-Seite auszuführen und die Rückgabewerte zu erhalten. Der erhaltene String wird anschließend mittels eines JSON-Parsers in ein für das Plugin lesbares Format formatiert. Durch den Umweg über einen Remote Service ist es nicht möglich die Wordcloud bereits beim Laden des Ergebnisbildschirms zu erzeugen. Aus diesem Grund muss zusätzlich ein Button betätigt werden, welcher die Erzeugung der Wordcloud initiiert.

Über einen Button am Ende der Seite ist es möglich zum Auswahlbildschirm zurückzukehren. Ferner existiert für jedes Modul ein Info-Button, der bei Betätigung ein Hinweisfenster mit Erklärungen zu der

[18] https://javacharts.openntf.org

[19] http://mistic100.github.io/jQCloud/index.html

jeweiligen Funktion anzeigt. Die nachfolgende Abbildung 5.10 zeigt das Hinweisfenster zum vierten Modul „Classifier“.

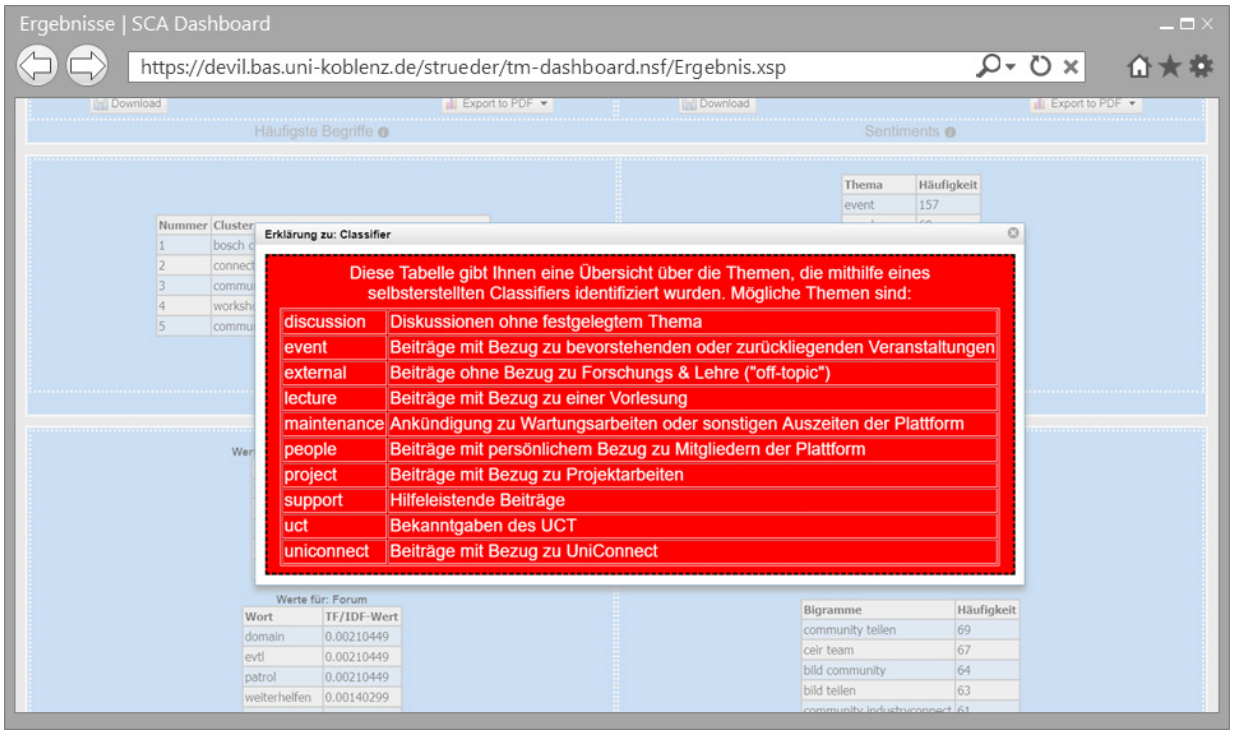

Abb. 5.10: Hinweisfenster zum Modul „Classifier“

Weiterhin ist anzumerken, dass die Wahl der zu betrachtenden Funktionen und die Anzahl der gewählten Communites einen Einfluss auf die Anzeige der Analyseergebnisse haben. Eine Übersicht der zu erwartenden Ergebnisse in Abhängigkeit der zuvor genannten Einflussfaktoren ist in Anhang 10 dargestellt.

6 Fazit

Das letzte Kapitel dieser Arbeit dient der explorativen Evaluation des Dashboards und dem Resümieren der vorherigen Kapitel sowie der gewonnenen Erkenntnisse. Ebenfalls wird ein Ausblick auf mögliche Erweiterungen und Verbesserungen des Dashboards gegeben.

6.1 Evaluation des Dashboards

Dieses Kapitel dient der explorativen Evaluation des entwickelten Dashboard-Prototyps. Dazu werden die Ergebnisse der Analysen von drei verschiedenartigen Communities der Plattform UniConnect miteinander verglichen, um zu verdeutlichen, welche Informationen sich aus den Ergebnissen extrahieren lassen und wie diese gedeutet werden können. Unterstützt wird dies durch gegenüberstellende Darstellungen der Visualisierungen des Dashboards.

Ausgewählt wurden drei Communities mit unterschiedlichen Bestimmungen. Im Folgenden werden die Communities vorgestellt:

Community 1: Allgemein zugängliche Einstiegscommunity

Die erste ausgewählte Community dient als Einstiegscommunity für sämtliche Nutzer der Plattform. In der Regel werden neu angemeldete Nutzer automatisch dieser Community hinzugefügt. Sie bietet den Nutzern die Möglichkeit, bei Problemen und Fragen zur Nutzung von UniConnect, mit den Betreibern der Plattform und anderen sachkundigen Nutzern in Kontakt zu treten. Weiterhin dient die Community als öffentlicher Kommunikationskanal des UCT. Dieses informiert dort beispielsweise über bevorstehende Wartungsarbeiten an der Plattform. Im weiteren Verlauf der Evaluation wird die Community mit „Einstieg“ bezeichnet.

Community 2: Kollaborationsprojekt zwischen Unternehmen und Vertretern des UCT

Die zweite Community dient als digitaler Versammlungsort eines Kollaborationsprojekts zwischen Unternehmen, welche ECS im Einsatz haben, und Vertretern des UCT. Die gesamte Kommunikation des Projekts findet innerhalb der Community unter Verwendung der zur Verfügung stehenden Funktionen statt. Im weiteren Verlauf der Evaluation wird die Community mit „Projekt“ bezeichnet.

Community 3: Lehrveranstaltung

Die dritte Community wird als digitaler Versammlungsort zur Organisation einer Lehrveranstaltung verwendet. Verantwortliche Dozenten nutzen die Community, um Lehrmaterialien aus Vorlesungen und Übungen den Studierenden zugängig zu machen sowie zur Ankündigung von wichtigen Terminen und zur Übermittlung von sonstigen Mitteilungen. Ebenfalls können die Studierenden die Community und ihre Funktionen zur Kontaktaufnahme mit den Dozenten und ihren Kommilitonen bei Problemen oder sonstigen Fragen nutzen. Im weiteren Verlauf der Evaluation wird die Community mit „Lehre“ bezeichnet.

Die nachfolgende Tabelle 6-1 gibt einen Überblick über die bei den Analysen gesetzten Filter. Eine Übersicht über die Anzahl der betrachteten Beiträge je Community ist in der darauffolgenden Tabelle 6-2 dargestellt.

Tab. 6-1: Auflistung der gesetzten Filter während der Evaluation

Einbezug der Kommentare	**sprachenabhängiges Stemming**	**spezifische Zeitspanne**	**Anzahl Begriffe / Themen**	**ausgewählte Funktion**
aktiviert	deaktiviert	deaktiviert	5	Blogs, Forum, Status-aktualisierungen

Tab. 6-2: Anzahl der bei der Evaluation betrachteten Beiträge je Community (Stand 18.09.2018)

	Anzahl der Beiträge der Funktion[20]			
Community	**Blogs**	**Forum**	**Statusaktualisierungen**	**Gesamtanzahl**
Einstieg	85	106	59	250
Projekt	311	124	281	716
Lehre	11	191	3	205

Im Folgenden werden die jeweiligen Ergebnisse der Module „Themen durch Classifier“, „Sentiments“, „Wordcloud“ und „Themencluster“ verglichen und gedeutet.

Thema	**Häufigkeit**
uniconnect	72
maintenance	69
support	43
event	26
uct	14
people	12
external	7
project	4
discussion	3

Community **Einstieg**

Thema	**Häufigkeit**
event	313
people	120
uniconnect	99
maintenance	60
discussion	44
external	32
project	21
support	21
uct	4
lecture	1

Community **Projekt**

Thema	**Häufigkeit**
uniconnect	60
event	38
maintenance	31
support	27
lecture	26
discussion	10
people	5
external	4
uct	4

Community **Lehre**

Abb. 6.1: Vergleich der Ergebnisse der Themenanalyse durch den Naive Bayes Classifier

[20] Zum Erhalt der Werte wurden die in Anhang 2 aufgelisteten SQL-Anfragen mit einer zusätzlichen Count-Funktion verwendet.

Anhand von Abbildung 6.1 ist zu erkennen, dass die Themenvielfalt der Beiträge in der Community Projekt am höchsten ist. Hier konnten alle zehn möglichen Themen identifiziert werden. Etwa die Hälfte der Beiträge widmet sich dem Thema „Events“. Eine Erklärung dafür kann sein, dass sich die Mitglieder des Projekts regelmäßig zu Meetings treffen und deren Terminkoordination bevorzugt über die textuellen Funktionen der Community vornehmen und im Nachgang auf der Plattform besprechen. Ferner ist die Anzahl der Beiträge rund um das Thema „People“ sehr hoch und lässt sich damit deuten, dass die Mitglieder die Funktionen oft zur Kommunikation untereinander verwenden und dafür nicht auf andere geeignetere Funktionen von UniConnect ausweichen. Wie in der Erklärung zu Community Einstieg schon erwähnt wurde, wird diese Community als Kommunikationskanal des UCT verwendet. Dies spiegeln auch die identifizierten Themen der Community wider. Die Mehrheit der Beiträge handelt von Informationen zur Plattform, zu Wartungsarbeiten und Veranstaltungen. Ebenfalls ist sichtbar, dass die Community auch für Hilfeersuchen („Support“) verwendet wird. Die Community Lehre hingegen zeigt Themen, die von einer Community zu einer Lehrveranstaltung erwartbar sind. So thematisieren etwa die Hälfte der Beiträge die Plattform und Informationen zu der Lehrveranstaltung selbst. Ebenfalls werden in dieser Community Hilfestellungen geleistet. Darauf deuten die 27 identifizierten Beiträge mit dem Thema „Support“ hin.

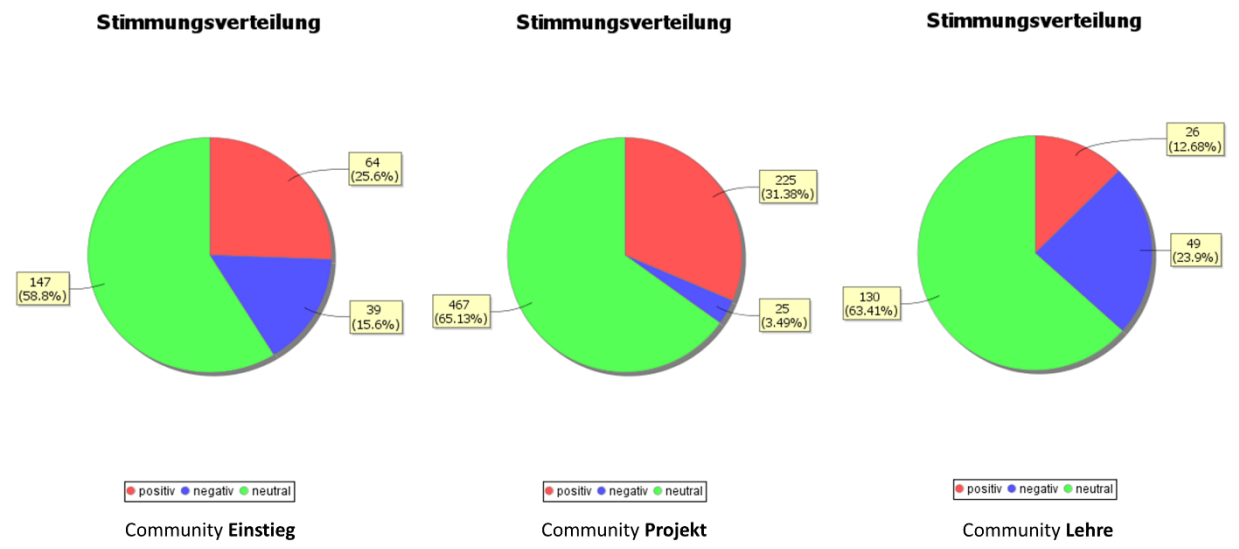

Abb. 6.2: Vergleich der Ergebnisse der Sentimentanalyse

Ein Vergleich der in Abbildung 6.2 dargestellten Ergebnisse der Sentimentanalysen zeigt, dass in jeder Community die Mehrheit der Beiträge eine neutrale Haltung aufweist. Mit nur 12,68 % (entspricht 26 Beiträgen) liegen die positiven Beiträge in Community Lehre nur auf dem letzten Platz der möglichen Haltungen. Besonders auffällig ist die Verteilung der negativen Haltungen. Während in der Community Projekt nur 3,49 % der Beiträge (entspricht 25 Beiträgen) eine negative Haltung aufweisen, liegen die Werte in den verbliebenen Communities bei 15,6 % in der Community Einstieg (entspricht 39 Beiträgen) und 23,9 % in der Community Lehre (entspricht 49 Beiträgen). Beide Werte können damit be-

gründet sein, dass die Communites zum Hilfeersuchen bei Problemen genutzt werden. Die Erklärungen der Probleme können negativ gewertete Begriffe beinhalten. So wird bereits das Wort „Problem“ als negativ aufgefasst. Ebenfalls können Ankündigungen zu Wartungsarbeiten, die in beiden Communities laut Themenanalyse bekannt gegeben werden, je nach Art der gewählten Worte negative Haltungen aufweisen.

Abb. 6.3: Vergleich der Wordclouds

Die in Abbildung 6.3 dargestellten Wordclouds zeigen die jeweils 100 meist verwendeten Begriffe innerhalb der ausgewählten Communities. Die abgebildeten Begriffe sind charakteristisch für die zugehörigen Communities und bestätigen die mittels des Naive Bayes Classifier identifizierten Themen.

Nummer	Cluster
1	dnug konferenz uct stand collaboration
2	problem fehler flemming loesung funktionieren
3	widget seite speichern kudos content
4	uniconnect plattform community ibm uct
5	social gis media business software

Community **Einstieg**

Nummer	Cluster
1	social unternehmen enterprise mitarbeiter dashboard
2	bosch connect inhalte content plattform
3	connections ibm kollegen thema antwort
4	community teilen freuen bild industryconnect
5	workshop koblenz industryconnect hotel social

Community **Projekt**

Nummer	Cluster
1	task betroffen mengeeinheit aufgabe punktzahl
2	fehler task nav edit voraus
3	artikel kreditoren buchung zertifikat studierende
4	basteam studierende klausureinsicht findet mail
5	punkte aufgabe fehler bild bekommen

Community **Lehre**

Abb. 6.4: Vergleich der identifizierten Themencluster

Die fünf jeweils in den Communities identifizierten Themencluster sind in Abbildung 6.4 dargestellt und decken sich mit den vorangegangenen Erkenntnissen. So deuten die Cluster 2 und 3 auf Hilfeersuchen hin, wohingegen die Cluster 4 und 5 auf bereitgestellte Informationen des UCT hindeuten und das erste Cluster auf eine Veranstaltung verweist. Die Cluster der Community Projekt weisen auf Dis-

kussionen zu Themen bezüglich der Plattform UniConnect, Veranstaltungen und den an dem Projekt beteiligten Unternehemen hin. Einen Hinweis darauf, dass die Community Lehre im Rahmen einer Lehrveranstaltung genutzt wird, geben die dort identifizierten Cluster. So weist das vierte Cluster auf eine Klausureinsicht hin. Ebenfalls deuten die restlichen Cluster auf Hilfeersuchen der Studierenden bezüglich der Veranstaltung hin.

Der Vergleich der Analysen der drei Communities zeigt, dass sich das erstellte Dashboard zur Identifizierung der in den Communities[21] besprochenen Themen nutzen lässt. Für jede der ausgewählten Communities konnten Themen identifiziert werden, die mit den zuvor verfassten Beschreibungen dieser übereinstimmten. Ebenfalls konnte eine Verbindung zwischen den Sentimentanalysen und den identifizierten Themen hergestellt werden. So ist es möglich bestimmte Haltungen mithilfe eines oder mehrerer Themen zu interpretieren.

6.2 Zusammenfassung und Erkenntnisse

Das Ziel dieser Arbeit war die Entwicklung eines Social Collaboration Analytics Dashboard-Prototyps, welcher durch Methoden des Text Mining, die Content Data des Enterprise Collaboration System UniConnect analysiert. Ziel der Analyse war die Identifikation der in den Communities diskutierten Themen mit der anschließenden Visualisierung der Ergebnisse auf verschiedene Arten. Zur Erreichung des Ziels dieser Arbeit galt es, die im ersten Kapitel dieser Arbeit genannten Forschungsziele mit ihren zugehörigen Forschungsfragen zu erfüllen und zu beantworten.

Das erste Forschungsziel umfasste die Entwicklung einer Methode zum Abruf und zur Aufbereitung der Content Data aus den UniConnect Datenbanken. Dazu wurden in Kapitel 2.2.3 zunächst die Grundlagen zu den UniConnect Datenbanken vermittelt und in Kapitel 3.2 eine Übersicht über das für das Text Mining wichtige Preprocessing gegeben. Die Implementierungen der Algorithmen in den zugehörigen Java-Klassen wurden in Kapitel 4.2 erläutert. Die zu diesem Forschungsziel gehörende Forschungsfrage wurde mit der Auflistung und Erklärung der einzelnen Schritte des Preprocessings in Kapitel 4.2 beantwortet.

Die Erstellung der Java-Klassen zur Analyse und Auswertung der aufbereiteten Daten unter Zuhilfenahme von Methoden des Text Mining war Gegenstand des zweiten Forschungsziels. Neben einer Einführung in das Thema Text Mining und der Vorstellung der Anwendungsgebiete dessen, folge die Veranschaulichung der praktischen Anwendung in Form der Java-Klassen in Kapitel 4.2. Dieses Kapitel enthält zudem die Antwort auf die erste Forschungsfrage. Die zweite Forschungsfrage thematisierte die Visualisierungen der Ergebnisse der Analysen. Diese wurde von den Java-Klassen losgelöst und über das XPages Framework abgewickelt. Erläutert wird dies in Kapitel 5.3.

[21] Folgende Communities wurden ausgewählt: 1. [Deutsch] UniConnect Community, 2. IndustryConnect, 3. Betriebliche Anwendungssysteme 2018

Das dritte Forschungsziel richtete sich an die Modellierung und prototypische Implementierung des Dashboards und der anschließenden Evaluation dessen. Die Modellierung eines Mockups des Dashboards ist in Kapitel 5.2 aufgeführt. Die anschließende Umsetzung mithilfe des XPages Frameworks wird im darauffolgenden Kapitel erläutert.

Zusammengefasst bietet diese Arbeit einen ersten Ansatz zur Verwendung von Text Mining Methoden bei der Ausführung von Social Collaboration Analytics. Es wird gezeigt, wie die Content Data der Plattform UniConnect zur Analyse der meist diskutierten Themen innerhalb von Communities verwendet wird. Die Verwendung von textuellen Daten stellt einen neuen Ansatz zur Analyse von Daten eines ECS dar, der im Rahmen der weiteren Forschung rund um SCA von Interesse sein kann und viele Wege zur Weiterentwicklung mit sich bringt. Das nachfolgende abschließende Kapitel zeigt, welche Möglichkeiten zur Weiterentwicklung das Dashboard mit sich bringt und welche Verbesserungsmöglichkeiten bestehen.

6.3 Ausblick

Die in dieser Arbeit gewonnenen Erkenntnisse sowie das erstellte Dashboard bieten zahlreiche Möglichkeiten zur weiteren Forschung und Weiterentwicklung.

Hinsichtlich des erstellten Dashboards bieten sich noch weitere Ansätze zur Weiterentwicklung. So bestünde die Option, die Beiträge weiterer Funktionen des ECS UniConnect in die Analysen mit einfließen zu lassen. Denkbar sei hier die Weiterentwicklung der Funktion Wikis. Momentan ist es nur möglich, wie bereits in Kapitel 2.2.3 erwähnt, verkürzte Versionen der Wikieinträge abzurufen. Die vollständigen Einträge werden in XML-Dateien gespeichert, deren Abruf und Aufbereitung bisher nicht erforscht wurden. Zukünftige Arbeiten könnten sich mit der Lösung dieses Problems befassen. Ebenfalls wäre es möglich, Textdateien, die über die Dateien-Funktion einer Community bereitgestellt wurden, auf ihren Inhalt und ein mögliches Thema zu überprüfen. Beide hier vorgestellten Weiterentwicklungen benötigen im Falle einer Umsetzung umfangreiche und rechenintensive Anwendungen von Methoden der Computerlinguistik.

Abseits der Identifizierung von meist diskutierten Themen in den Communities, sind noch weitere Analyseszenarien der Content Data möglich. So beschreibt ein in Tabelle 2-5 vorgestelltes Schlüsselthema von SCA die Identifikation von charakteristischen Nutzertypen eines ECS basierend auf der Systemverwendung. Dies geschieht durch Analysen der Metrics-Datenbank. Alternativ ließe sich eine solche Identifizierung durch die Analyse der textuellen Daten mit Methoden des Text Mining durchführen. Denkbar sei hier die Anwendung von Methoden der Document Classification. Hierzu ist es nötig, ein Datenset zu erstellen, welches eine gelabelte Auflistung von charakteristischen Äußerungen bestimmter Nutzergruppen enthält. So ließe sich der Satz „Ich benötige Eure Hilfe“ einem hilfesuchenden Nutzer zuweisen. Ein entsprechender Algorithmus zur Document Classification, würde dann unter Zuhilfenahme des Datensets die Beiträge der Plattform analysieren und einen möglichen Nut-

zertyp ausgeben. Dazu kann der in Kapitel 4.2 vorgestellte Naive Bayes Algorithmus verwendet werden.

Eine weitere mögliche Anwendung der Auswertung der textuellen Daten, kann die Analyse der sozialen Interaktionen zwischen den Nutzern sein. Hier ließen sich insbesondere durch Sentimentanalysen die Stimmungen und Meinungen der Nutzer zueinander feststellen. Hierzu müssen Beiträge der Plattform identifiziert werden, die zur Konversation zwischen Nutzern verwendet werden. Ein Indiz dafür können @Nutzername-Markierungen sein.

Ein weiteres Schlüsselthema von SCA beschreibt die Identifikation von Sachkenntnis. Bezogen auf Communities, die sich Forschungsprojekten widmen, kann die in den Communities erzeugte Content Data verwendet werden, um festzustellen, wie sachdienlich die erstellten Beiträge eines bestimmten Nutzers im Kontext des Themas des Projektes sind. Durch die Erstellung von Themenclustern der Beiträge eines Nutzers, ließe sich feststellen, wie sehr die Cluster dem Thema des Projektes entsprechen. Weichen diese häufig vom Projektthema ab, so scheint der Nutzer der Kollaboration des Projektes zu schaden. Besonders für studentische Projekte im Rahmen von Projekt- oder Forschungspraktika, in denen die Leistung der einzelnen Studierenden bewertet werden, bildet eine solche Analyse einen Mehrwert für den zugehörigen Dozenten. Dieser kann sofort erkennen, wie hoch das eingebrachte Engagement und Wissen eines Studierenden ist.

Die für diese Arbeit erstellten Java-Klassen sowie das zugehörige Dashboard, sind speziell für die Analyse der Content Data des ECS UniConnect entwickelt worden. Im Rahmen von zukünftigen Forschungsarbeiten ließe sich feststellen, inwiefern die erstellte Arbeit auf weitere ECS und deren Datenbanken übertragen werden kann. Insbesondere die Kompatibilität mit anderen auf IBM Connections basierenden ECS bietet eine Möglichkeit zur Erforschung.

Literaturverzeichnis

Alarcon, R. A., Guerrero, L. A., & Pino, J. A. (2005). Groupware Components as Providers of Contextual Information, (August), 1–4.

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. Retrieved from http://arxiv.org/abs/1707.02919

Appelt, W. (2001). What groupware functionality do users really use? Analysis of the usage of the BSCW system. *Proceedings - 9th Euromicro Workshop on Parallel and Distributed Processing, PDP 2001*, 337–341. https://doi.org/10.1109/EMPDP.2001.905060

Bächle, M. (2006). Social Software. *Informatik-Spektrum*, *29*(2), 121–124. https://doi.org/10.1007/s00287-006-0063-2

Behrendt, S., & Richter, A. (2015). Business Intelligence 2.0. In *Business Intelligence for New-Generation Managers* (pp. 97–111). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-15696-5_8

Behrendt, S., Richter, A., & Trier, M. (2014). Mixed methods analysis of enterprise social networks. *Computer Networks*, *75*(PB), 560–577. https://doi.org/10.1016/j.comnet.2014.08.025

Bhardwaj, B. (2016). Text Mining, its Utilities, Challenges and Clustering Techniques. *International Journal of Computer Applications*, *135*(7), 975–8887. Retrieved from http://www.ijcaonline.org/research/volume135/number7/bhardwaj-2016-ijca-908452.pdf

Blei, D. M. (2012). Probabilistic Topic Models. *Communications of the ACM*, *55*(4), 77–84. https://doi.org/10.1109/MSP.2010.938079

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022. https://doi.org/10.1162/jmlr.2003.3.4-5.993

Bøving, K. B., & Simonsen, J. (2004). Http Log Analysis: An Approach to Studying the Use of Web-Based Information Systems. *Scandinavian Journal of Information Systems*, *16*(1), 145–174. Retrieved from http://aisel.aisnet.org/sjis%0Ahttp://aisel.aisnet.org/sjis/vol16/iss1/3

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0. *CRISP-DM Consortium*, 76. https://doi.org/10.1109/ICETET.2008.239

Chaves, M. A., & Córdoba, E. R. (2014). Deciphering event logs in SharePoint Server: A methodology based on process mining. *Proceedings of the 2014 Latin American Computing Conference, CLEI 2014*. https://doi.org/10.1109/CLEI.2014.6965174

Chong, R. F. (2002). *Getting Started with DB2 for z/OS ™ and OS/390 ® Version 7 for DB2 Distributed Platform Users*.

Debortoli, S., Müller, O., Junglas, I., & Brocke, J. vom. (2016). Text Mining for Information Systems Researchers: An Annotated Topic Modeling Tutorial. Communications of the Association for Information Systems (CAIS). *Communications of the Association for Information Systems*, *39*(1), 29. Retrieved from https://webdocs.uni.li/public/07274867.PDF

Diehl, R., Kuettner, T., & Schubert, P. (2013). Introduction of enterprise collaboration systems: In-depth studies show that laissez-faire does not work. *26th International Bled Conference*, 236–250. Retrieved from http://aisel.aisnet.org/bled2013/8/

Donnelly, M., McGuckin, T., & Wallace, M. (2011). *Mastering XPages: A Step-by-Step Guide to XPages Application Development and the XSP Language*. IBM Press.

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook*. Cambridge University Press. https://doi.org/10.1017/CBO9780511546914

Gupta, V., & Lehal, G. S. (2009). A Survey of Text Mining Techniques and Applications. *Journal of Emerging Technologies in Web Intelligence*, *1*(1), 17. https://doi.org/10.4304/jetwi.1.1.60-76

Gutwin, C., & Greenberg, S. (2002). A Descriptive Framework of Workspace Awareness for Real-Time Groupware Keywords. *Computer Supported Cooperative Work*, *11*, 411–446.

Hacker, J., Bodendorf, F., & Lorenz, P. (2017). Helper, Sharer or Seeker? – A Concept to Determine Knowledge Worker Roles in Enterprise Social Networks. *Proceedings Der 13. Internationale Tagung Wirtschaftsinformatik (WI)*, 668–682. Retrieved from http://www.wi2017.ch/de/proceedings

Herzog, C., Richter, A., & Steinhüser, M. (2015). Towards a Framework for the Evaluation Design of Enterprise Social Software. *Proceedings of the 36th International Conference on Information Systems*, (December), 1–20.

Hippner, H., & Rentzmann, R. (2006). Text mining. *Informatik-Spektrum*, *29*(4), 287–290. https://doi.org/10.1007/s00287-006-0091-y

Hotho, A., Nürnberger, A., & Paaß, G. (2005). A Brief Survey of Text Mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, *20*, 19–62. https://doi.org/10.1111/j.1365-2621.1978.tb09773.x

Jeners, N., & Prinz, W. (2014). Metrics for Cooperative Systems. In *Proceedings of the 18th International Conference on Supporting Group Work - GROUP '14* (pp. 91–99). New York, New York, USA: ACM Press. https://doi.org/10.1145/2660398.2660407

Koch, M., & Gross, T. (2006). Computer-Supported Cooperative Work - Concepts and Trends. *Proc Conf of the Association Information And Management AIM*, *75*, 165–172. Retrieved from http://www.kooperationssysteme.de/wordpress/wp-content/uploads/Koch2006-aim.pdf

Liddy, E. D. (2001). Natural Language Processing. *Encyclopedia of Library and Information Science*, *2*.

Litzel, N. (2016). Was ist Big Data Analytics? Retrieved August 19, 2018, from https://www.bigdata-insider.de/was-ist-big-data-analytics-a-575678/

Miner, G., Elder, J., Hill, T., Nisbet, R., Delen, D., & Fast, A. (2012). The Seven Practice Areas of Text Analytics. In *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications* (pp. 29–41). https://doi.org/10.1016/B978-0-12-386979-1.00002-5

Muller, M., Ehrlich, K., Matthews, T., Perer, A., Ronen, I., & Guy, I. (2012). Diversity Among Enterprise Online Communities: Collaborating, Teaming, and Innovating through Social Media. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2815–2824. https://doi.org/10.1145/2207676.2208685

Nasirifard, P., & Peristeras, V. (2009). Expertise Extracting Within Online Shared Workspaces.

Porter, M. F., Merico, D., Isserlin, R., Bader, G. D., Tosolini, M., Kirilovsky, A., … Greenblatt, J. (2006). An algorithm for suffix stripping. *Program*, *40*(3), 211–218. https://doi.org/10.1108/00330330610681286

Prasad, K. N. S. S. V., Saritha, S. K., & Saxena, D. (2017). A Survey Paper on Concept Mining in Text Documents. *International Journal of Computer Applications*, *166*(11), 7–10.

Richter, A., & Koch, M. (2007). Social Software - Status Quo und Zukunft, (2007-01), 1–49. https://doi.org/10.1111/j.1083-6101.2011.01559.x

Richter, A., Stocker, A., Müller, S., & Avram, G. (2011). Knowledge management goals revisited - A

cross-sectional analysis of social software adoption in corporate environments. *22nd Australasian Conference on Information Systems*, 1–10. https://doi.org/doi:10.1108/03055721311329927

Riemer, K., Stieglitz, S., & Meske, C. (2015). From Top to Bottom: Investigating the Changing Role of Hierarchy in Enterprise Social Networks. *Business and Information Systems Engineering*, *57*(3), 197–212. https://doi.org/10.1007/s12599-015-0375-3

Rish, I. (2001). An empirical study of the naive Bayes classifier. *Empirical Methods in Artificial Intelligence Workshop, IJCAI*, *22230*(JANUARY 2001), 41–46. https://doi.org/10.1039/b104835j

Rishel, T., Perkins, A. L., Yenduri, S., Zand, F., & Iyengar, S. S. (2006). Augmentation of a Term / Document Matrix with Part-of- Speech Tags to Improve Accuracy of Latent Semantic Analysis. *Proceedings of the 5th WSEAS International Conference on Applied Computer Science*, 573–578.

Schubert, P., & Glitsch, J. H. (2015). Adding Structure to Enterprise Collaboration Systems: Identification of Use Cases and Collaboration Scenarios. *Procedia Computer Science*, *64*, 161–169. https://doi.org/10.1016/j.procs.2015.08.477

Schubert, P., & Glitsch, J. H. (2016). Use cases and collaboration scenarios: How employees use socially-enabled enterprise collaboration systems (ECS). *International Journal of Information Systems and Project Management*, *4*(2), 41–62. https://doi.org/10.12821/ijispm040203

Schubert, P., & Williams, S. P. (2013). The Concept of Social Business : Oxymoron or Sign of a Changing Work Culture? *Proceedings of the 26th Bled Conference*, (Kernerman).

Schubert, P., & Williams, S. P. (2016). The Case of UniConnect - The Shaping of an Academic Collaboration Platform. *Multikonferenz Wirtschaftsinformatik (MKWI 2016)*, (Elbanna 2007), 1–12.

Schubert, P., & Winkelmann, A. (2016). Einführung in Betriebliche Anwendungssysteme. In *Betriebliche Anwendungssysteme* (pp. 1–21).

Schwade, F., & Schubert, P. (2017). Social Collaboration Analytics for Enterprise Collaboration Systems: Providing Business Intelligence on Collaboration Activities. In *50th Hawaii International Conference on System Sciences (HICSS)* (pp. 401–410). https://doi.org/10.24251/HICSS.2017.048

Schwade, F., & Schubert, P. (2018). Social Collaboration Analytics for Enterprise Social Software: A Literature Review. *Multikonferenz Wirtschaftsinformatik 2018*, (1), 205–216.

Shah, N., & Mahajan, S. (2012). Document Clustering: A Detailed Review. *International Journal of Applied Information Systems (IJAIS)*, *4*(5), 30–38. https://doi.org/10.5120/8202-1598

Sixtus, M. (2005, August 25). W wie Wiki. Retrieved August 17, 2018, from https://www.zeit.de/2005/35/C-Humannetz-Glossar

Smith, M., Hansen, D. L., & Gleave, E. (2009). Analyzing Enterprise Social Media Networks. *2009 International Conference on Computational Science and Engineering*, *4*, 705–710. https://doi.org/10.1109/CSE.2009.468

Steinhüser, M., Herzog, C., Richter, A., & Hoppe, U. (2015). A Process Perspective on the Evaluation of Enterprise Social Software. *2nd European Conference on Social Media*, (July), 429–436.

Steyvers, M., & Griffiths, T. (n.d.). Probabilistic Topic Models. *Latent Semantic Analysis: A Road to Meaning*.

Stieglitz, S., Dang-Xuan, L., Bruns, A., & Neuberger, C. (2014). Social Media Analytics - Ein interdisziplinärer Ansatz und seine Implikationen für die Wirtschaftsinformatik. *Wirtschaftsinformatik*, 1–9. https://doi.org/10.1007/s11576-014-0407-5

Stieglitz, S., Riemer, K., & Meske, C. (2014). Hierarchy or Activity ? the Role of Formal and Informal Influence in Eliciting Responses From. *Ecis*, (July), 1–14.

Talib, R., Hanif, M. K., Ayesha, S., & Fatima, F. (2016). Text Mining: Techniques, Applications and Issues. *International Journal of Advanced Computer Science and Applications*, *7*(11), 414–418. Retrieved from www.ijacsa.thesai.org

Tan, A.-H. (1999). Text Mining: The state of the art and the challenges. *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, *8*, 65–70. https://doi.org/10.1.1.38.7672

Tan, W., Blake, M. B., Saleh, I., & Dustdar, S. (2013). Social-network-sourced big data analytics. *IEEE Internet Computing*, *17*(5), 62–69. https://doi.org/10.1109/MIC.2013.100

Thompson, V. (2015). Worldwide Enterprise Social Networks Online Communities 2015-2019 and 2014 Vendor Shares, (July 2015), 1–21.

Ullenboom, C. (2014). *Java ist auch eine Insel*. *Galileo Press*.

Vaishnavi, V., & Kuechler, W. (2012). Design Science Research in Information Systems Overview of Design Science Research. *Information Systems*, (1978), 1–16. https://doi.org/10.1007/978-1-4419-5653-8

Vijayarani, S., Ilamathi, J., & Nithya, M. (2015). Preprocessing Techniques for Text Mining - An Overview. *International Journal of Computer Science & Communication Networks*, *5*(1), 7–16. Retrieved from http://www.ijcscn.com/Documents/Volumes/vol5issue1/ijcscn2015050102.pdf

Viol, J., & Hess, J. (2016). Information Systems Research on Enterprise Social Networks – A State-of-the-Art Analysis. *Multikonferenz Wirtschaftsinformatik (MKWI 2016)*, *0*, 351–362.

Watson, H. J. (2006). Dashboards and Scorecards. *Business Intelligence Journal*, *11*(1).

Wehner, B., Falk, T., & Leist, S. (2017). What Benefits Do They Bring? a Case Study Analysis on Enterprise Social Networks. *Ecis 2017*, *2017*, 2069–2085.

Williams, S. P. (2011). Business Software. In *Wettbewerbsfaktor Business Software* (pp. 11–21). https://doi.org/10.1080/10196780701503088

Williams, S. P. (2013). Enterprise Resource Planning (ERP). Retrieved August 17, 2018, from https://integrated-business-design.de/expertise/topics/enterprise-resource-planning-erp/

Williams, S. P., & Schubert, P. (2015). Social Business Readiness Survey 2014, (1), 1–25. https://doi.org/10.13140/RG.2.1.2375.2800

Anhang

Anhang 1: SQL-Datenbankabfragen zum Erhalt der Anzahl der Beiträge / Communities

Blogs
Anzahl der vorhandenen Blogeinträge
`SELECT Count (*) AS Count` `  FROM blogs.weblogentry`
Anzahl der vorhandenen Kommentare
`SELECT Count (*) AS Count` `  FROM blogs.roller_comment`

Community
Anzahl der vorhandenen öffentlichen Communities

`SELECT Count (*) AS Count`
`  FROM sncomm.community`
`  WHERE community_type = 'public'`

Anzahl der vorhandenen öffentlichen Communities mit Zugangskontrolle

`SELECT Count (*) AS Count`
`  FROM sncomm.community`
`  WHERE community_type = 'publicInviteOnly'`

Anzahl der vorhandenen privaten Communities

`SELECT Count (*) AS Count`
`  FROM sncomm.community`
`  WHERE community_type = 'private'`

Forum
Anzahl der vorhandenen Foren

`SELECT Count (*) AS Count`
`  FROM forum.df_node`
`  WHERE nodetype = 'application/forum'`

Anzahl der vorhandenen Themen

`SELECT Count (*) AS Count`
`  FROM forum.df_node`
`  WHERE nodetype = 'forum/topic'`

Anzahl der vorhandenen Antworten

`SELECT Count (*) AS Count`
`  FROM forum.df_node`
`  WHERE nodetype = 'forum/reply'`

Statusaktualisierungen[1]
Anzahl der vorhandenen Statusaktualisierungen

`SELECT Count (*) AS Count`
`  FROM homepage.board_entries`
`  WHERE source = 'COMMUNITIES'`

Anzahl der vorhandenen Kommentare

`SELECT Count (*) AS Count`
`  FROM homepage.board_comments`
`  WHERE item_url LIKE '/service%'`

Wikis
Anzahl der vorhandenen Wikieinträge
`SELECT Count (*) AS Count` `  FROM wikis.media`
Anzahl der vorhandenen Kommentare
`SELECT Count (*) AS Count` `  FROM wikis.media_comment`

[1] Es werden nur Beiträge berücksichtigt, die in Communities erstellt wurden. Siehe Selektion (`WHERE`) der Abfragen.

Anhang 2: SQL-Datenbankabfragen zum Erhalt der Content Data

Blogs

Auflistung sämtlicher Blogeinträge der Platform inklusive Kommentare

```
SELECT text
  FROM blogs.websiteassoc a
       JOIN (SELECT websiteid,text,pubtime
               FROM blogs.weblogentry
             UNION ALL
             (SELECT websiteid,content AS TEXT,publishtime AS PUBTIME
                FROM blogs.roller_comment)) b
         ON a.websiteid = b.websiteid
```

Auflistung sämtlicher Blogeinträge der Platform ohne Kommentare

```
SELECT text
  FROM blogs.websiteassoc a
       JOIN (SELECT websiteid,text,pubtime
               FROM blogs.weblogentry) b
         ON a.websiteid = b.websiteid
```

zusätzliche Parameter für Community- und Zeitraumauswahl

```
  WHERE associd LIKE 'COMMUNITYUUID'
   AND pubtime BETWEEN 'YYYY-MM-DD' AND 'YYYY-MM-DD'
```

Forum

Auflistung sämtlicher Foreneinträge der Platform inklusive Kommentare

```
SELECT description
  FROM forum.df_nodecommmap b
       JOIN (SELECT forumuuid,description
               FROM forum.df_node
              WHERE ( nodetype = 'forum/topic'
                       OR nodetype = 'forum/reply' )
                AND forum.df_node.description IS NOT NULL
                AND forum.df_node.description NOT LIKE '') a
         ON b.forumuuid = a.forumuuid
```

Fortsetzung auf der nächsten Seite.

Auflistung sämtlicher Foreneinträge der Platform ohne Kommentare

```
SELECT description
  FROM forum.df_nodecommmap b
       JOIN (SELECT forumuuid,description
               FROM forum.df_node
              WHERE ( nodetype = 'forum/topic' )
                AND forum.df_node.description IS NOT NULL
                AND forum.df_node.description NOT LIKE '') a
         ON b.forumuuid = a.forumuuid
```

zusätzliche Parameter für Community- und Zeitraumauswahl

```
  WHERE communityuuid LIKE 'COMMUNITYUUID'
   AND a.created BETWEEN 'YYYY-MM-DD' AND 'YYYY-MM-DD'
```

Statusaktualisierungen

Auflistung sämtlicher Statusaktualisierungen der Platform inklusive Kommentare

```
SELECT content
  FROM homepage.person e
       JOIN (SELECT board_owner_assoc_id,content,creation_date
               FROM homepage.board c
                    JOIN ((SELECT container_id,content,creation_date
                             FROM homepage.board_entries
                            WHERE homepage.board_entries.SOURCE = 'COMMUNITIES')
                          UNION ALL
                          (SELECT container_id,b.content,creation_date
                             FROM homepage.board_entries a
                                  JOIN (SELECT entry_id,content
                                          FROM homepage.board_comments) b
                                    ON a.entry_id = b.entry_id)) d
                      ON c.board_container_id = d.container_id) f
         ON f.board_owner_assoc_id = e.person_id
```

Auflistung sämtlicher Statusaktualisierungen der Platform ohne Kommentare

```
SELECT content
  FROM homepage.person c
       JOIN (SELECT board_owner_assoc_id,content,creation_date
               FROM homepage.board a
                    JOIN (SELECT container_id,content,creation_date
                            FROM homepage.board_entries
                           WHERE homepage.board_entries.SOURCE = 'COMMUNITIES')
                        b
                      ON a.board_container_id = b.container_id) d
         ON c.person_id = d.board_owner_assoc_id
```

zusätzliche Parameter für Community- und Zeitraumauswahl

```
  WHERE exid LIKE 'COMMUNITYUUID'
   AND (d/f).creation_date BETWEEN 'YYYY-MM-DD' AND 'YYYY-MM-DD'
```

Wikis
Auflistung sämtlicher Wikieinträge der Platform inklusive Kommentare

```
SELECT b.summary
  FROM ((SELECT external_container_id,id
          FROM wikis.library) a
        JOIN (SELECT library_id,summary,last_update
                FROM ((SELECT library_id,last_update,comment AS SUMMARY
                          FROM wikis.media_comment)
                       UNION ALL
                       (SELECT library_id,last_update,summary
                          FROM wikis.media))) b
          ON b.library_id = a.id)
```

Auflistung sämtlicher Wikieinträge der Platform ohne Kommentare

```
SELECT b.summary
  FROM ((SELECT external_container_id,id
          FROM wikis.library) a
        JOIN (SELECT library_id,last_update,summary
                FROM wikis.media) b
          ON a.id = b.library_id)
 WHERE b.summary NOT LIKE ''
```

zusätzliche Parameter für Community- und Zeitraumauswahl

```
  WHERE external_container_id LIKE 'COMMUNITYUUID'
   AND last_update BETWEEN 'YYYY-MM-DD' AND 'YYYY-MM-DD'
```

Anhang 3: Verwendete externe Java Libraries

Library	Link	Verwendungszweck
Guava	https://github.com/google/guava	wird zur korrekten Funktionsweise von language-detector benötigt
HPPC	https://github.com/carrotsearch/hppc	wird zur korrekten Funktionsweise von Mallet benötigt
iText	https://mvnrepository.com/artifact/com.lowagie/itext	wird zur korrekten Funktionsweise von Java Charts benötigt
JCommon	http://www.jfree.org/jcommon	wird zur korrekten Funktionsweise von Java Charts benötigt
JFreeChart	http://www.jfree.org/jfreechart	wird zur korrekten Funktionsweise von Java Charts benötigt
jsoup	https://jsoup.org	zum Entfernen von HTML Code im Preprocessing
language-detector	https://github.com/optimaize/language-detector	zur Spracherkennung der Beiträge
MALLET	http://mallet.cs.umass.edu	zur Durchführung des LDA Algorithmus
Java Naive Bayes Classifier	https://github.com/ptnplanet/Java-Naive-Bayes-Classifier	zur Durchführung der Klassifikation mittels Naive Bayes
SLF4J	https://www.slf4j.org	wird zur korrekten Funktionsweise von language-detector benötigt
snowballstem	https://github.com/snowballstem/snowball	zur Durchführung des Stemmings

Zusätzlich wurden folgende Quellen verwendet:

- HTMLTableBuilder zur Ausgabe von HTML Tabellen als String: https://gist.github.com/2sbsbsb/2951464
- Java Charts zum Erzeugen von Diagrammen in XPages: https://javacharts.openntf.org
- jQCloud zur Erzeugung der Wordclouds: http://mistic100.github.io/jQCloud/index.html

Anhang 4: Quellcode zum Abruf und zur Speicherung der Content Data

```
public static List<String> query(String[] array) {
        String jdbcClassName = "com.ibm.db2.jcc.DB2Driver";
        String url = array[1];
        String user = "reader7";
        String password = "eIt3nAso";
        Statement stmt = null;
        ResultSet rs = null;
        List<String> data = new ArrayList<String>();

        Connection connection = null;
        try {
            Class.forName(jdbcClassName);
            connection = DriverManager.getConnection(url, user, password);

            stmt = connection.createStatement();
            rs = stmt.executeQuery(array[0]);

            while (rs.next()) {
                String post = rs.getString(1);
                data.add(post);
            }
            rs.close();

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }

        return data;
    }
```

Anhang 5: Quellcode zur Ausführung des sprachenabhängigen Preprocessings

```
public static List<String> preprocessing(List<String> input, String lemma)
        throws IOException, InstantiationException, IllegalAccessException,
        ClassNotFoundException {

    List<String> output = new ArrayList<String>();
    List<LanguageProfile> languageProfiles = new LanguageProfileReader()
            .readAllBuiltIn();
    LanguageDetector languageDetector = LanguageDetectorBuilder.create(
            NgramExtractors.standard()).withProfiles(languageProfiles)
            .build();
    TextObjectFactory textObjectFactory = CommonTextObjectFactories
            .forDetectingOnLargeText();

    for (String post : input) {

        String text = Jsoup.parse(post).text();
        text = Jsoup.parse(text).text();

        StringBuilder builder = new StringBuilder();
        for (String word : text.split("\\s+")) {
            if (!word.contains("http") && !word.contains("https")
                    && !word.contains("@{{") && !word.contains("}}")
                    && !word.contains("www.") && !word.contains("@")
                    && !word.contains("{META")) {
                word = word.replaceAll("\\p{P}", " ").replace("ö", "oe")
                        .replace("Ö", "Oe").replace("ü", "ue").replace("Ü",
                                "Ue").replace("ä", "ae").replace("Ä", "Ae")
                        .replace("ß", "ss");
                builder.append(word).append(" ");
            }
        }

        text = builder.toString();
        text = text.replaceAll(" +", " ").replaceAll("[^a-zA-Z ]", "")
                .toLowerCase();

        TextObject textObject = textObjectFactory.forText(text);
        Optional<LdLocale> lang = languageDetector.detect(textObject);
        String loc = lang.toString();

        if (loc.equals("Optional.of(de)")) {
            if (lemma.equals("0")) {
                text = removeStopWords(text, 1);
            } else {
                text = removeStopWords(text, 1);
                StringBuilder stemmed = new StringBuilder();
                for (String word : text.split("\\s+")) {
                    Class stemClass = Class
                            .forName("org.tartarus.snowball.ext.germanStemmer");
                    SnowballStemmer stemmer = (SnowballStemmer) stemClass
                            .newInstance();
                    stemmer.setCurrent(word);
                    stemmer.stem();
                    String sword = stemmer.getCurrent();
                    stemmed.append(sword).append(" ");
                }
                text = stemmed.toString();
                text = removeStopWords(text, 1);
            }
        } else if (loc.equals("Optional.of(en)")) {
            if (lemma.equals("0")) {
                text = removeStopWords(text, 2);
            } else {
                text = removeStopWords(text, 2);
                StringBuilder stemmed = new StringBuilder();
                for (String word : text.split("\\s+")) {
                    Class stemClass = Class
                            .forName("org.tartarus.snowball.ext.germanStemmer");
                    SnowballStemmer stemmer = (SnowballStemmer) stemClass
                            .newInstance();
                    stemmer.setCurrent(word);
                    stemmer.stem();
                    String sword = stemmer.getCurrent();
                    stemmed.append(sword).append(" ");
                }
                text = stemmed.toString();
                text = removeStopWords(text, 2);
            }
        } else {
            text = removeStopWords(text, 0);
        }
        output.add(text);
    }

    return output;
}
```


Anhang 6: Quellcode zur Ermittlung der meist verwendeten Begriffe

```
1  public static String[] calculation(List<String> input, int count) {
2          List<String> terms = new ArrayList<String>();
3          for (String post : input) {
4              for (String word : post.split(" ")) {
5                  terms.add(word);
6              }
7          }
8
9          Map<String, Word> wordMap = new HashMap<String, Word>();
10         for (String word : terms) {
11             Word currentWord = wordMap.get(word);
12             if (currentWord == null)
13                 wordMap.put(word, new Word(word, 1));
14             else
15                 currentWord.frequency++;
16         }
17
18         List<Word> wordList = new ArrayList<Word>(wordMap.values());
19         Collections.sort(wordList, new Comparator<Word>() {
20             public int compare(Word o1, Word o2) {
21                 if (o1.frequency == o2.frequency)
22                     return o1.word.compareToIgnoreCase(o2.word);
23                 return Integer.valueOf(o2.frequency).compareTo(
                                Integer.valueOf(o1.frequency));
24             }
25         });
26
27         String[] list = new String[count];
28         String[] freqWords = new String[count];
29         int[] frequencies = new int[count];
30
31         for (int i = 0; i < count; i++) {
32             list[i] = wordList.get(i).toString();
33         }
34
35         int i = 0;
36
37         for (String elem : list) {
38             String[] temp = elem.split("-");
39             freqWords[i] = temp[0];
40             String number = temp[1];
41             int x = Integer.parseInt(number);
42             frequencies[i] = x;
43             i++;
44         }
45
46         return freqWords;
47     }
```

Anhang 7: Quellcode zur Ermittlung der Häufigkeiten von n-Grams

```
1  public static String grams(List<String> input, String count, String length) {
2      String table = "";
3      String name = "";
4
5      try {
6
7          String posts = "";
8
9          for (String post : input) {
10             posts = post + " " + posts;
11         }
12
13         int len = Integer.parseInt(length);
14
15         if (len == 2) {
16             name = "Bigramme";
17         } else if (len == 3) {
18             name = "Trigramme";
19         } else if (len == 4) {
20             name = "4-Grams";
21         } else if (len == 5) {
22             name = "5-Grams";
23         }
24
25         String[] parts = posts.split("\\s+");
26         String[] result = new String[parts.length - len + 1];
27
28         for (int i = 0; i < parts.length - len + 1; i++) {
29             List<String> ngrams = new ArrayList<String>();
30             for (int k = 0; k < len; k++) {
31                 ngrams.add(parts[i + k]);
32             }
33             Collections.sort(ngrams);
34             result[i] = ngrams.toString().replace("[", "").replace("]", "")
                       .replace(",", "");
35         }
36
37         List<String> asList = Arrays.asList(result);
38         Set<String> mySet = new HashSet<String>(asList);
39         HashMap<String, Integer> map = new HashMap<String, Integer>();
40         for (String s : mySet) {
41             map.put(s, Collections.frequency(asList, s));
42         }
43
44         HashMap<String, Integer> sortedmap = new HashMap<String, Integer>();
45         sortedmap = sortByComparator(map, false);
46         int counter = Integer.parseInt(count);
47         HTMLTableBuilder htmlBuilder = new HTMLTableBuilder(null, true,
                   counter - 1, 2);
48         htmlBuilder.addTableHeader(name, "Häufigkeit");
49         int i = 0;
50
51         for (Entry<String, Integer> pair : sortedmap.entrySet()) {
52             if (i < counter) {
53                 htmlBuilder.addRowValues(pair.getKey(), Integer
                           .toString(pair.getValue()));
54                 i++;
55             } else {
56                 break;
57             }
58         }
59
60         table = htmlBuilder.build();
61     } catch (NegativeArraySizeException neg) {
62         table = "<p><strong>Es konnten keine " + name
                   + " gebildet werden!</strong></p>";
63     }
64     return table;
65 }
```

Anhang 8: Link zum Gitlab Repository

Unter folgendem Link kann das Gitlab Repository mit dem Quellcode der Java-Klassen erreicht werden:

https://gitlab.uni-koblenz.de/stefanstrueder/tm-dashboard

Anhang 9: Quellcode der Managed Bean

```
1  public class Bean implements Serializable {
2
3      /**
4       *
5       */
6      private static final long serialVersionUID = 6774987059874733100L;
7
8      private static String[] communities; // Liste der ausgewählten Communities
9      private static String comments; // Auswahl Einbezug der Kommentare
10     private static String prepro; // Auswahl Stemming
11     private static String date; // Auswahl Zeitspanne
12     private static String begin; // Eingabe Startdatum
13     private static String end; // Eingabe Enddatum
14     private static String count; // Auswahl Anzahl von Themen
15     private static String ngrams; // Auswahl Länge der n-Grams
16     private static String[] functions; // Auswahl Funktionen
17
18     public Bean() {
19         System.out.println("Instantiating Bean");
20     }
21     public void setCommunities(String[] communities) {
22         Bean.communities = communities;
23     }
24     public static String[] getCommunities() {
25         return communities;
26     }
27     public void setComments(String comments) {
28         Bean.comments = comments;
29     }
30     public static String getComments() {
31         return comments;
32     }
33     public void setPrepro(String prepro) {
34         Bean.prepro = prepro;
35     }
36     public static String getPrepro() {
37         return prepro;
38     }
39     public void setDate(String date) {
40         Bean.date = date;
41     }
42     public static String getDate() {
43         return date;
44     }
45     public void setBegin(String begin) {
46         Bean.begin = begin;
47     }
48     public static String getBegin() {
49         return begin;
50     }
51     public void setEnd(String end) {
52         Bean.end = end;
53     }
54     public static String getEnd() {
55         return end;
56     }
57     public void setCount(String count) {
58         Bean.count = count;
59     }
60     public static String getCount() {
61         return count;
62     }
63     public void setNgrams(String ngrams) {
64         Bean.ngrams = ngrams;
65     }
66     public static String getNgrams() {
67         return ngrams;
68     }
69     public void setFunctions(String[] functions) {
70         Bean.functions = functions;
71     }
72     public static String[] getFunctions() {
73         return functions;
74     }
75 }
```

Anhang 10: Übersicht der zu erwartenden Ergebnisse in Abhängigkeit der gewählten Funktionen und der Anzahl der Communities

---

Zuordnung der Module:

1. häufigste Begriffe
2. Sentiments
3. Themencluster
4. Themen durch Classifier
5. TF/IDF
6. n-Grams
7. Wordcloud

Modul	Anzahl der Communities	Funktion	angezeigte Ergebnisse
1	eine Community	mehrere Funktionen	häufigste Begriffe der Verknüpfung der Beiträge der gewählten Funktionen
		Einzelfunktion	häufigste Begriffe der Beiträge der gewählten Funktion
	mehrere Communities	mehrere Funktionen	häufigste Begriffe der Verknüpfung der Beiträge der gewählten Funktionen und Communities
		Einzelfunktion	häufigste Begriffe der Verknüpfung der Beiträge der gewählten Funktion aller Communities
2	eine Community	mehrere Funktionen	Sentiments der Verknüpfung der Beiträge der gewählten Funktionen
		Einzelfunktion	Sentiments der Beiträge der gewählten Funktion
	mehrere Communities	mehrere Funktionen	Sentiments der Verknüpfung der Beiträge der gewählten Funktionen und Communities
		Einzelfunktion	Sentiments der Verknüpfung der Beiträge der gewählten Funktion aller Communities
3	eine Community	mehrere Funktionen	Cluster der Verknüpfung der Beiträge der gewählten Funktionen
		Einzelfunktion	Cluster der Verknüpfung der Beiträge der gewählten Funktion
	mehrere Communities	mehrere Funktionen	separate Tabelle mit Clustern der Verknüpfung der Beiträge der gewählten Funktionen für jede Community
		Einzelfunktion	separate Tabelle mit Clustern der Verknüpfung der Beiträge der gewählten Funktion für jede Community
4	eine Community	mehrere Funktionen	Themen der Verknüpfung der Beiträge der gewählten Funktionen
		Einzelfunktion	Themen der Verknüpfung der Beiträge der gewählten Funktion

Fortsetzung auf der nächsten Seite.

4	mehrere Communities	mehrere Funktionen	separate Tabelle mit Themen der Verknüpfung der Beiträge der gewählten Funktionen für jede Community
		Einzelfunktion	separate Tabelle mit Themen der Verknüpfung der Beiträge der gewählten Funktion für jede Community
5	eine Community	mehrere Funktionen	separate Tabelle mit TF/IDF Werten für jede gewählte Funktion
		Einzelfunktion	keine Berechnung von TF/IDF Werten möglich
	mehrere Communities	mehrere Funktionen	separate Tabelle mit Werten der Verknüpfung der Beiträge der gewählten Funktionen für jede Community
		Einzelfunktion	separate Tabelle mit Werten der Verknüpfung der Beiträge der gewählten Funktion für jede Community
6	eine Community	mehrere Funktionen	n-Grams der Verknüpfung der Beiträge der gewählten Funktionen
		Einzelfunktion	n-Grams der Verknüpfung der Beiträge der gewählten Funktion
	mehrere Communities	mehrere Funktionen	separate Tabelle mit n-Grams der Verknüpfung der Beiträge der gewählten Funktionen für jede Community
		Einzelfunktion	separate Tabelle mit n-Grams der Verknüpfung der Beiträge der gewählten Funktion für jede Community
7	eine Community	mehrere Funktionen	Wordcloud der häufigsten Begriffe der Verknüpfung der Beiträge der gewählten Funktionen
		Einzelfunktion	Wordcloud der häufigsten Begriffe der Beiträge der gewählten Funktion
	mehrere Communities	mehrere Funktionen	Wordcloud der häufigsten Begriffe der Verknüpfung der Beiträge der gewählten Funktionen und Communities
		Einzelfunktion	Wordcloud der häufigsten Begriffe der Verknüpfung der Beiträge der gewählten Funktion aller Communities