

Freie wissenschaftliche Arbeit zur Erlangung des akademischen Grades
Bachelor of Science in Wirtschaftsinformatik

Text-Mining als Instrument der evidenzbasierten Analyse konvergierender Forschungsgebiete

Bachelorthesis

im Fachbereich Wirtschaftswissenschaften II
im Studiengang Wirtschaftsinformatik
der Hochschule für Technik und Wirtschaft Berlin

vorgelegt von: Clemens Brauer
Matrikel-Nr.: 557110

Erstbetreuer: Prof. Dr. Axel Hochstein
Zweitbetreuer: M. Eng. Mila Galeitzke

Abgabetermin: 25.02.2019

Abstract

Aus den Bereichen der wissenschaftlichen Literatur werden wir zunehmend mit einer überwältigenden Informationsflut konfrontiert, durch die es schwerfällt, manuell die aktuellen Erkenntnisse und Entwicklungen zu überblicken. Um den Menschen dabei zu unterstützen bietet Text-Mining, ein relativ junges Forschungsgebiet, welches Methoden aus Teilgebieten der Informatik verbindet, eine effiziente Möglichkeit, die Verarbeitung dieser Informationen zu bewerkstelligen. Im Themengebiet des Science Mapping konnten dadurch bereits gute Ergebnisse anhand von Literatur, wie wissenschaftlichen Publikationen, Patenten oder Büchern, erzielt werden. Diese ermöglichen es beispielsweise, Zusammenhänge zwischen verschiedenen Wissensdomänen anhand von Zitationsverknüpfungen sichtbar zu machen. In dieser Bachelorarbeit wird, in Kooperation mit dem Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik, ein Verfahren entwickelt, um darüber hinaus Konvergenzen zwischen wissenschaftlichen Bereichen, durch die Zielbeschreibungen von zukünftig beginnenden Forschungsprojekten sichtbar zu machen. Dafür werden die Resultate von Text-Mining Anwendungen, wie Textklassifikation und Schlüsselwort-Extraktion, genutzt, um einen Netzwerkgraphen zu konstruieren, welcher Verbindungen zwischen wissenschaftlichen Bereichen anhand von gemeinsamen Schlüsselwörtern darstellt.

Inhaltsverzeichnis

Abstract.....	II
Inhaltsverzeichnis	III
Abbildungsverzeichnis	V
Tabellenverzeichnis	VI
1 Einleitung.....	1
2 Text-Mining.....	3
2.1 Definition	3
2.2 Anwendungsbereiche.....	3
2.2.1 Informationsrückgewinnung.....	4
2.2.2 Textklassifizierung	4
2.2.3 Text Clustering	5
2.2.4 Extraktion von Informationen.....	5
2.2.5 Konzept Extraktion	5
2.2.6 Web Mining.....	5
2.2.7 Verarbeitung von natürlicher Sprache	6
3 Textvorverarbeitung	7
3.1 Tokenisierung.....	7
3.2 Stopwörter.....	7
3.3 Wortartenbestimmung.....	7
3.4 N-Gramm	8
3.5 Stammformreduktion.....	8
4 Automatische Textklassifizierung	9
4.1 Definition	9
4.2 Methodologie.....	9
4.3 Prozessbeschreibung.....	11
4.3.1 Merkmalsextraktion	12
4.3.2 Merkmalsreduktion	13
4.3.3 Erstellung eines Vektorraummodell.....	14
4.3.4 Klassifikation.....	15
5 Automatische Schlüsselwort-Extraktion	18
5.1 Definition	18
5.2 Methodologie.....	18
5.3 Prozessbeschreibung.....	21
5.3.1 Ermittlung der Schlüsselwort-Kandidaten.....	21
5.3.2 Extraktion der Schlüsselwörter	21
5.3.3 Evaluation	22
5.4 Schlüsselwort-Extraktionsverfahren.....	24
5.4.1 YAKE!	25
5.4.2 KEA	28
5.4.3 PositionRank	34
5.4.4 Evaluationsergebnisse aktueller Verfahren	38

6	Praktische Ausarbeitung	39
6.1	Daten.....	39
6.1.1	Quelle	39
6.1.2	Selektierung und Extrahierung	40
6.2	Automatische Textklassifizierung	41
6.2.1	Tool	41
6.2.2	Klassifizierungsprozess	42
6.3	Automatische Schlüsselwort-Extraktion	43
6.3.1	Schlüsselwort-Extraktorklasse.....	43
6.3.2	Schlüsselwort-Extraktionsverfahren	45
6.3.3	Schlüsselwort-Extraktionsprozess.....	50
6.3.4	Evaluation.....	51
6.4	Graphaufbau	52
6.4.1	Graph-Generatorklasse	52
6.4.2	Graph-Erstellungsprozess	56
6.5	Visualisierung.....	56
6.5.1	Tool	56
6.5.2	Ergebnisse.....	57
7	Science Mapping	59
7.1	Definition	59
7.2	Aspekte	59
7.2.1	Datenquellen	59
7.2.2	Analyseeinheiten	60
7.2.3	Datenvorverarbeitung	61
7.2.4	Beziehungsmaße.....	61
7.2.5	Mapping-Prozess.....	62
7.2.6	Visualisierungstechniken	62
7.3	Projekte	63
7.3.1	Paperscape	63
7.3.2	VOSviewer.....	64
7.3.3	CiteSpace	65
8	Schlussbemerkungen und Ausblick	67
	Literaturverzeichnis.....	VII
	Anhang	XVIII
A	CORDIS Beispielprojekt.....	XVIII
B	Visualisierung Health and Fitness und Science	XIX
C	Visualisierung Health and Fitness und Science Version 2	XX
D	Visualisierung Health and Fitness und Business and Industrial	XXI
E	Visualisierung Health and Fitness und Business and Industrial Version 2	XXII
F	Visualisierung Science und Business and Industrial.....	XXIII
G	Visualisierung Science und Business and Industrial Version 2	XXIV
H	Visualisierung Science, Business and Industrial und Health and Fitness....	XXV

Abbildungsverzeichnis

Abbildung 1 Zeitstrahl Forschungsquellen.....	1
Abbildung 2 Überblick über Text-Mining Anwendungsbereiche.....	4
Abbildung 3 Methoden Textklassifizierung.....	10
Abbildung 4 Prozessbeschreibung Textklassifikation.....	12
Abbildung 5 Kategorisierung der automatischen Schlüsselwort-Extraktion.....	19
Abbildung 6 Graphbasierende Verfahren.....	20
Abbildung 7 Konfusionsmatrix.....	23
Abbildung 8 PositionRank Graphenerstellung.....	35
Abbildung 9 Iterative Scoreberechnung.....	37
Abbildung 10 Prozess der praktischen Ausarbeitung.....	39
Abbildung 11 Datenextrahierung und Selektierung.....	40
Abbildung 12 IBM Kategorien.....	41
Abbildung 13 Daten Klassifizierung.....	42
Abbildung 14 Implementierung Schlüsselwort-Extraktorklasse.....	44
Abbildung 15 Implementierung YAKE! Schlüsselwort-Extraktion.....	45
Abbildung 16 Implementierung KEA Lernprozess.....	47
Abbildung 17 Implementierung KEA Schlüsselwort-Extraktion.....	49
Abbildung 18 Implementierung PositionRank Schlüsselwort-Extraktion.....	49
Abbildung 19 Implementierung Schlüsselwort-Extraktionsprozess.....	50
Abbildung 20 Evaluation der Implementierung.....	52
Abbildung 21 Implementierung Graph-Generatorklasse.....	53
Abbildung 22 Implementierung filterSchluesselwoerter.....	53
Abbildung 23 Implementierung schluesselwoerterZuGraph.....	54
Abbildung 24 Implementierung kategorieZuGraph.....	55
Abbildung 25 Implementierung schreibeGraphInGmlDatei.....	55
Abbildung 26 Implementierung Graph-Erstellungsprozess.....	56
Abbildung 27 Visualisierung Kategorie Übersicht.....	57
Abbildung 28 Paperscape Kartenausschnitt.....	64
Abbildung 29 VOSviewer Darstellung.....	65
Abbildung 30 CiteSpace Darstellung einer Publikation.....	66

Tabellenverzeichnis

Tabelle 1 Bag-Of-Words Darstellung.....	12
Tabelle 2 Beispiel Textklassifizierung Vektorraummodell.....	16
Tabelle 3 Beispiel Textklassifikation Wahrscheinlichkeitstabelle.....	17
Tabelle 4 Evaluation Datensets.....	22
Tabelle 5 Schlüsselwort-Extraktionsverfahren.	24
Tabelle 6 Evaluation YAKE!.....	28
Tabelle 7 Diskretisierungstabelle.....	31
Tabelle 8 Klassenmerkmalsgewichtung.	32
Tabelle 9 Trainingswahrscheinlichkeit.	32
Tabelle 10 Merkmale Beispielwortphrasen.....	32
Tabelle 11 KEA Gesamteffektivität.	33
Tabelle 12 KEA Effekt Anzahl Trainingsdokumente.	34
Tabelle 13 KEA Effekt Dokumentenlänge.	34
Tabelle 14 Evaluation PositionRank.	38
Tabelle 15 Top Score Evaluation 2014.	38
Tabelle 16 Evaluation Implementierung.	51
Tabelle 17 Visualisierung Kategorie Übersicht.	58
Tabelle 18 Bibliografische Online-Datenbanken.....	60
Tabelle 19 Bibliometrische Techniken.	60

1 Einleitung

Aufgrund der Verschmelzung von verschiedenen wissenschaftlichen Bereichen konnten bereits in der Vergangenheit neue Erkenntnisse und Entwicklungen entstehen. Beispiele dafür sind die Entdeckung von Augenkrankheiten durch die Anwendung von maschinell lernenden Algorithmen (De Fauw, et al. 2018), das maschinelle Auslesen von menschlichem Verhalten während der Autofahrt (López, et al. 2018) oder Rückschlüsse auf das frühe Universum, welche mittels der Erkenntnisse aus terrestrischen Superfluide-Experimenten gezogen wurden (Zurek 1985). Aufgrund dieser und einer Menge weiterer Beispiele lässt sich eine ansteigende Popularität von interdisziplinär konvergierenden Forschungsansätzen erkennen.

Roco, et al. (2013, S. 1) beschreiben dabei die Wichtigkeit dieser Konvergenzen wie folgt:

„Convergence in knowledge, technology, and society is the accelerating, transformative interaction among seemingly distinct scientific disciplines, technologies, and communities to achieve mutual compatibility, synergism, and integration, and through this process to create added value for societal benefit.“

Um einen Einblick über den aktuellen Stand dieser Verschmelzung von Wissensdomänen zu erhalten, können durch Techniken aus dem Bereich des Science Mapping Visualisierungen erstellt und untersucht werden. Wie in Abbildung 1 dargestellt, werden dafür Quellen wie beispielsweise Bücher, Patente, Forschungsergebnisse und wissenschaftliche Publikationen betrachtet. Diese spiegeln den aktuellen Entwicklungsstand der Vergangenheit bis hin zur Gegenwart wider. Darüber hinaus ermöglichen aktuelle Forschungsprojekte und bereits existierende Ausschreibungen für zukünftige Forschungsprogramme einen Ausblick über die kommenden Entwicklungen und Technologien.

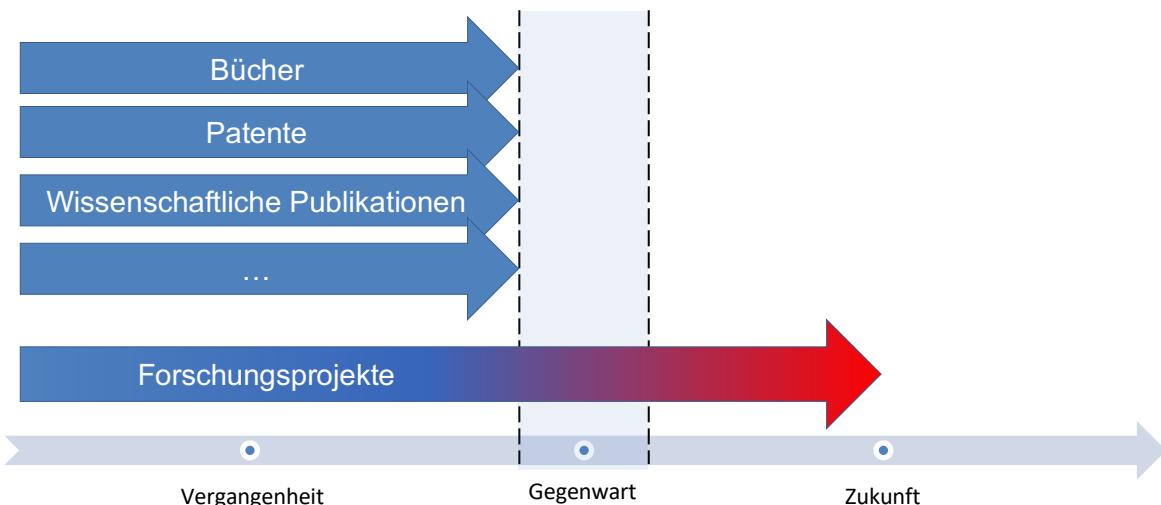


Abbildung 1 Zeitstrahl Forschungsquellen. (Quelle: Eigene Darstellung)

Um die zukünftige Entwicklung der Konvergenzen zwischen wissenschaftlichen Bereichen zu analysieren, ist ein Ansatz der Untersuchung von für die Zukunft ausgeschriebenen Forschungsprojekten denkbar. Daraus resultierende Ergebnisse könnten genutzt werden, um beispielsweise neue Schnittstellen von Wissensdomänen über Bereiche hinweg zu identifizieren und damit zukünftige Entwicklungen, durch eine engere Kooperation dieser, effektiver voranzutreiben.

Zielsetzung

Aus diesem Grund ist, in Zusammenarbeit mit dem Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik, das Ziel dieser Bachelorarbeit ein Verfahren zu entwickeln, welches anhand von zukünftigen Forschungsprojekten etwaig auftretende Schnittpunkte zwischen Forschungsbereichen mithilfe von Text-Mining Methoden sichtbar macht.

Dabei stehen die folgenden Forschungsfragen im Fokus:

Können, mithilfe von Text-Mining Methoden, Schnittstellen zwischen wissenschaftlichen Bereichen anhand von Zielbeschreibungen zukünftiger Forschungsprojekte identifiziert werden?

Wie können die relevanten Informationen aus den Texten selektiert, extrahiert, klassifiziert und visualisiert werden?

Aufbau der Arbeit

In Kapitel 2 wird ein grundlegender Überblick über das Thema Text-Mining vermittelt, wobei neben der Definition die verschiedenen Anwendungsbereiche betrachtet werden.

In Kapitel 3 werden die Begriffe Tokenisierung, Stoppwörter, Wortartenbestimmung, N-Gramm und Stammformreduktion, als Prozessschritte aus dem Bereich der Textvorverarbeitung, genauer definiert.

Im Anschluss wird in Kapitel 4 auf die automatische Textklassifizierung eingegangen, wobei diese im ersten Abschnitt definiert wird. Daraufhin werden die verschiedenen Methoden und der Prozessablauf für die Klassifizierung von Texten erläutert.

Um ein Grundverständnis über den Prozess der automatischen Schlüsselwort-Extraktion zu erhalten, wird in Kapitel 5 genauer auf das Thema eingegangen. Neben der Definition werden dabei die aktuellen Methoden benannt. Darauf folgt eine Beschreibung des Prozessablaufes, von der Ermittlung der Schlüsselwort-Kandidaten bis hin zur Evaluation der Resultate. Da in der praktischen Ausarbeitung ein Vergleich zwischen drei verschiedenen Schlüsselwort-Extraktionsverfahren durchgeführt wird, erfolgt eine genaue Erläuterung deren Funktionsweise im letzten Abschnitt des Kapitels.

Aufbauend auf dem vorab gesammelten theoretischen Wissen wird in Kapitel 6 die praktische Ausarbeitung behandelt. In den Abschnitten Daten, automatische Textklassifizierung, automatische Schlüsselwort-Extraktion, Graphaufbau und Visualisierung wird jeder einzelne Prozessschritt des entwickelten Verfahrens beschrieben.

In Kapitel 7 wird der interdisziplinäre Bereich des Science Mapping näher erläutert. Dabei werden neben der Definition die einzelnen Aspekte erklärt. Des Weiteren erfolgt eine Beschreibung von verschiedenen Projekten aus dem Bereich.

Um die Erkenntnisse der Arbeit zusammenzufassen erfolgt in Kapitel 8 die Schlussbemerkung sowie ein Ausblick über die Thematik.

2 Text-Mining

Da Text-Mining Prozesse in dieser Bachelorarbeit eine entscheidende Rolle spielen, wird in diesem Kapitel ein kurzer Überblick über das Thema gegeben sowie die Anwendungsbereiche erklärt.

2.1 Definition

Heyer, Quasthoff und Wittig (2008, S. 1-3) beschreiben Text-Mining als eine Folge von Prozessen, bei denen mithilfe von verschiedenen Werkzeugen, Wissen aus unstrukturierten bzw. schwach-strukturierten Texten gewonnen wird. Texte werden dabei als unstrukturiert bezeichnet, da in diesen beispielsweise Schlüsselbegriffe, Grundkonzepte oder inhaltliche Beziehungen nicht hervorgehoben werden.

Gemäß Feldman und Sanger (2006, S. 1) wird durch die Anwendung von Text-Mining Methoden versucht, nützliche Informationen aus Textquellen zu extrahieren, indem interessante Muster identifiziert und untersucht werden. Dabei wird anhand von computergestützten Verfahren vorgegangen, welche eine automatische bzw. semi-automatische Strukturierung von sehr großen Mengen von Texten unterstützen.

Kao und Poteet (2010, S.1) definiert Text-Mining wie folgt:

„Text mining is the discovery and extraction of interesting, non-trivial knowledge from free or unstructured text. This encompasses everything from information retrieval (i.e. document or web site retrieval) to text classification and clustering, to (somewhat more recently) entity, relation, and event extraction.“

2.2 Anwendungsbereiche

Wie in Abbildung 2 dargestellt, teilt Miner et al. (2012, S. 31) Text-Mining in 7 Anwendungsbereiche auf. Dazu zählen die Informationsrückgewinnung (englisch Information Retrieval), das Clustern von Dokumenten (englisch Document Clustering), die Klassifizierung von Dokumenten (englisch Document Classification), die Extraktion von Informationen (englisch Information Extraction), die Verarbeitung von natürlicher Sprache (englisch Natural Language Processing), die Konzept Extraktion (englisch Concept Extraction) und das Web Mining. Die Anwendungsbereiche lassen sich verschiedenen wissenschaftlichen Forschungsgebieten wie dem Data Mining, Künstliche Intelligenz und Maschinelles Lernen (englisch Artificial Intelligence (AI) and Machine Learning), Statistik (englisch Statistics), Computerlinguistik (englisch Computational Linguistics), Datenbanken (englisch Databases) und der Bibliotheks- und Informationswissenschaft (englisch Library and Information Sciences) zuordnen, wobei diese sich thematisch teilweise überschneiden.

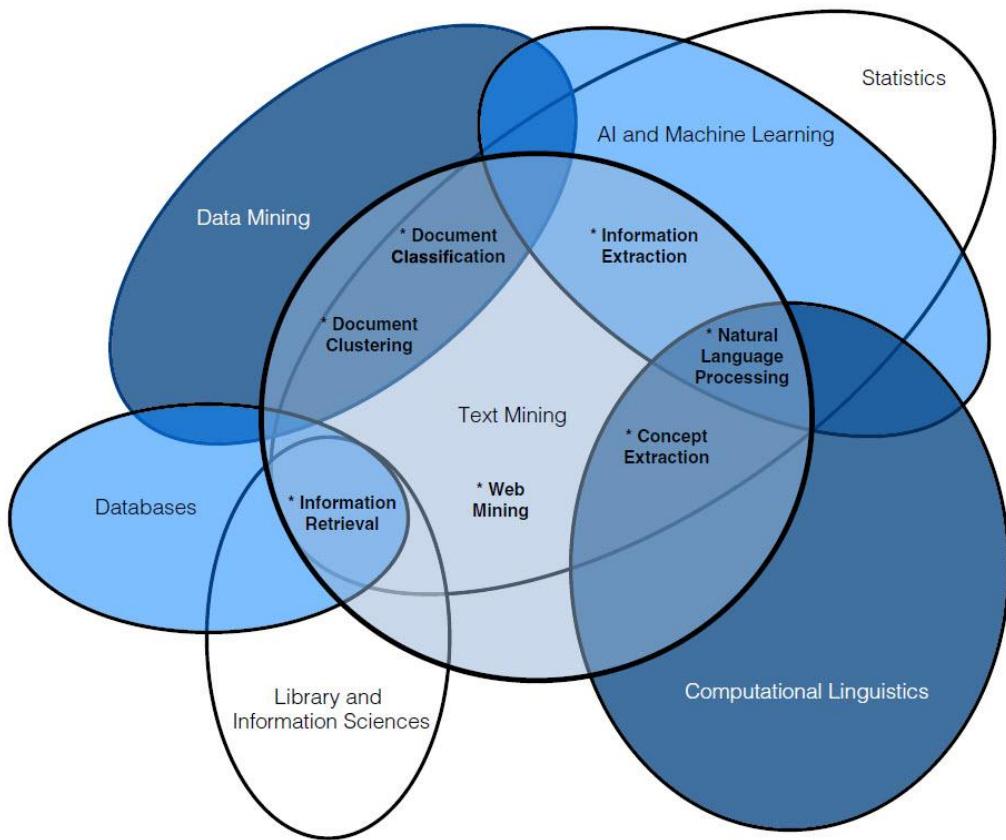


Abbildung 2 Überblick über Text-Mining Anwendungsbereiche. (Quelle: Miner et al. (2012, S. 31))

In den folgenden Unterabschnitten werden die Anwendungsbereiche kurz beschrieben.

2.2.1 Informationsrückgewinnung

Gerhard Salton (1989) beschreibt zwei Hauptziele von Informationsrückgewinnungssystemen:

1. Dokumente indizieren

Alle eingehenden Dokumente werden so verarbeitet und gespeichert, dass sie später einfach wieder abgerufen werden können. Dafür wird eine Indexstruktur konstruiert, welche jedem Dokument bestimmte Schlüsselwörter zuweist.

2. Dokumente abrufen

Anhand von Abfrageinformationen werden alle Dokumente nach ihrer Relevanz geordnet ausgegeben. Die Relevanz der Dokumente ist dabei von der Ähnlichkeit zwischen den Abfrageinformationen und dem Dokumenteninhalt abhängig.

2.2.2 Textklassifizierung

Gemäß Ignatov und Mihalcea (2017, S. 117 - 127) hat der Prozess der Textklassifizierung die Aufgabe, Dokumenten automatisch eine oder mehrere vordefinierte Klassen zuzuweisen. Dazu werden typischerweise bereits vorab klassifizierte Trainingsdokumente genutzt, mittels welcher Zusammenhänge zwischen deren Texten und Klassen erkannt werden. Auf Grundlage dieser Erkenntnisse, lassen sich neue Dokumente klassifizieren. Da dies im praktischen Teil dieser Bachelorarbeit eine Rolle spielt, wird in Kapitel 4 genauer darauf eingegangen.

2.2.3 Text Clustering

Heyer, Quasthoff und Wittig (2008, S.196) beschreiben das Text Clustering als ein Verfahren, bei dem aus einer Menge von Dokumenten oder Wörtern Teilmengen gebildet werden, welche sich inhaltlich ähneln. Die gebildeten Teilmengen werden dabei Cluster genannt. Die Erkennung der Dokumenten- bzw. Wortähnlichkeiten wird anhand von Clustering Algorithmen durchgeführt, die, im Gegensatz zu der Textklassifizierung, keine Trainingsdaten benötigen. Neben der Gruppierung durch ihre Ähnlichkeiten, ermöglicht es das Text Clustering neue Zusammenhänge zwischen Dokumenten zu entdecken (Miner, et al. 2012, S. 960 ff.).

2.2.4 Extraktion von Informationen

Bei diesem Anwendungsbereich werden strukturierte Informationen aus unstrukturierten Texten extrahiert. Die Informationen werden im Regelfall einem Typen zugeordnet, der vorab bereits bestimmt ist. Beispielsweise könnte es sich dabei um Personen, Organisationen, Events oder Länder handeln, wobei dies von den Anforderungen der jeweiligen Nutzung abhängig ist. Wie in Abbildung 2 zu sehen, wird die Extraktion von Informationen dem Forschungsbereich des Maschinellen Lernens zugeordnet. Das hat den Hintergrund, dass die Informationen anhand von Satzmustern automatisch erkannt werden können. Dafür werden vorab Trainingsdaten benötigt, bei welchen die zu findenden Informationen bereits markiert wurden. Dadurch lernt die Maschine beispielsweise, dass eine Organisation immer mit den Worten: „arbeitet bei ...“ oder „GmbH“ in Zusammenhang gebracht werden kann (Ignatov und Mihalcea 2017, S. 131). Dennoch beschreibt Jiang (2012, S. 30), dass momentan ein großes Interesse an Verfahren besteht, welche ohne den Einsatz von maschinellem Lernen auskommen. Dafür müssen vorab alle Informationstypen in eine Liste aufgenommen werden, anhand derer sich die Wörter von Dokumenten abgleichen lassen.

2.2.5 Konzept Extraktion

Laut Prasad, Saritha und Saxena (2017, S. 7) befasst sich der Anwendungsbereich der Konzept Extraktion damit, das Hauptthema von Texten zu suchen und extrahieren. Dafür wird die Bedeutung informativer Begriffe im Text untersucht und anhand dieser das Konzept bestimmt. Das Konzept selber muss dabei im Text nicht direkt erwähnt werden, sondern erschließt sich aus den Verbindungen der vorab untersuchten Begriffe.

2.2.6 Web Mining

Gemäß Mabrathu und Srinivasulu (2017, S. 52) umfasst Web Mining den gesamten Prozess, anhand von Webdaten potenziell nützliche Informationen oder Kenntnisse zu erlangen. Web Mining kann in die folgenden drei Bereiche unterteilt werden:

1. Web-Usage-Mining

Bei diesem Verfahren werden die Nutzungsdaten einer Webseite analysiert. Dabei wird die Interaktion von Usern mit einer Webseite erfasst, um beispielsweise Verhaltensweisen zu erkennen.

2. Web-Content-Mining

Web-Content-Mining befasst sich mit der Erkenntnisgewinnung aus den Inhalten von Webseiten. Diese können als unstrukturierte Daten in Form von Texten, semi-strukturierten Daten in Form von HTML oder XML und in stark strukturierter Form durch beispielsweise Tabellen dargestellt werden. Aufgrund der Anwendung von verschiedenen Algorithmen lassen sich daraus Informationen generieren.

3. Web-Structure-Mining

Dieses Verfahren untersucht die Beziehung zwischen Webseiten. Dies wird mit der Hilfe von Graphen realisiert, wobei die Webseiten als Knoten und die Beziehungen

als Kanten dargestellt werden. Ziel ist es, Beziehungen zwischen den einzelnen Webseiten zu erkennen, um daraus Erkenntnisse zu gewinnen.

2.2.7 Verarbeitung von natürlicher Sprache

Gemäß Kao und Poteet (2010, S.1) befasst sich dieser Anwendungsbereich damit, Texte anhand von Regeln und Algorithmen computerbasiert zu verarbeiten. Dafür werden sprachliche Konzepte sowie grammatische Strukturen genutzt. Im Bezug zum Text-Mining, stellt die Verarbeitung der natürlichen Sprache einen wichtigen Bestandteil der Datenvorverarbeitungsphase dar.

3 Textvorverarbeitung

Aufbauend auf den Text-Mining-Anwendungsbereich der Verarbeitung von natürlicher Sprache, werden in diesem Kapitel grundlegende Begriffe für die Textvorverarbeitung beschreiben. Die Vorverarbeitung von Text ist nahezu immer erforderlich, bevor dieser analysiert werden kann. In Abhängigkeit von der Text Quelle und dem verfolgten Ziel der Analyse ist es notwendig, beispielsweise HTML oder XML Tags zu entfernen, Wörter voneinander zu trennen, einzelne Wörter auf ihre Stammform zu reduzieren oder uninteressante Wörter zu löschen (Ignatov und Mihalcea 2017, S. 53). Da die Textvorverarbeitung auch in dieser Bachelorarbeit für verschiedene Prozesse eine Rolle spielt, werden in diesem Kapitel ausgewählte Methoden für ein besseres Verständnis beschrieben.

3.1 Tokenisierung

Um den Inhalt von Dokumenten analysieren zu können, muss deren Text vorab in sinnvolle Segmente aufgeteilt werden. Dabei können diese unterschiedliche Textebenen, wie zum Beispiel Kapitel, Abschnitte, Absätze, Sätze, Wörter und sogar Silben oder Phoneme, repräsentieren. Für Text-Mining wird der Text üblicherweise in die Segmente Wort oder Satz unterteilt. Die aus dem Prozess resultierenden einzelnen Elemente werden als Tokens bezeichnet, woraus sich der Name des Vorganges selbst als Tokenisation herleiten lässt (Feldman und Sanger 2006, S. 60). Im Regelfall erfolgt die Trennung der einzelnen Segmente anhand von Leerzeichen und Punktnotationen, wobei die Handhabung von beispielsweise Abkürzungen, Bindestrichen, zusammenhängenden Entitäten und Zahlen beachtet werden muss (Ignatov und Mihalcea 2017, S. 54). Die beispielsweise in einem Satz S vorkommenden Wörter w , werden wie folgt getrennt:

$$S(w_1 \ w_2 \ w_3 \ w_4 \ .) \rightarrow (w_1, w_2, w_3, w_4, .)$$

3.2 Stoppwörter

Bei Stoppwörtern handelt es sich meist um Wörter, welche aufgrund ihres häufigen Auftretens und einer unzureichenden Vermittlung von themenspezifischen Informationen als nicht relevant angesehen werden. Dazu gehören im Regelfall Wortformen aus den geschlossenen Wortklassen Artikel, Konjunktionen und Präpositionen, wobei die daraus resultierende Liste an Stoppwörtern, in Abhängigkeit des Anwendungsbereiches, nach Belieben mit weiteren Wörtern ergänzt werden kann (Heyer, Quasthoff und Wittig 2008, S. 96). Um bei der Analyse von Text wichtigen Inhalten eine höhere Aufmerksamkeit zu geben, kann das Entfernen von Stoppwörtern hilfreich sein (Ignatov und Mihalcea 2017, S. 55).

Beispiele für Stoppwörter sind: aber, ich, dir, sich, ihnen, während, zwar (Auszug aus der spaCy Stopwortliste (ExplosionAI GmbH 2017, spaCy Stopwordlist))

3.3 Wortartenbestimmung

Bei diesem Vorgang wird die Wortart eines Wortes bestimmt und diesem zugewiesen. Während der Bestimmung muss der Satz, in welchem sich das Wort befindet, mit betrachtet werden, da beispielsweise bei den Sätzen:

„Wir müssen die Wortarten bestimmen.“

und

„Das Bestimmen der Wortarten ist leicht.“

„bestimmen“ als Verb oder Substantiv deklariert werden muss (Jackson und Moulinier 2002, S. 12). Für die Erkennung der Wortarten werden überwachte maschinelle Lernalgorithmen

angewandt, die das Wissen aus bereits manuell bestimmten Datensätzen nutzen, um die Wortarten für Wörter aus neuen Texten zu erkennen (Ignatov und Mihalcea 2017, S. 60).

3.4 N-Gramm

Als N-Gramm werden Fragmente bezeichnet, welche sich aus einer Sequenz von N aufeinander folgenden Elementen eines Textes zusammensetzen. Dabei können diese Elemente Buchstaben, Phoneme oder Wörter repräsentieren (Ingersoll, Morton und Farris 2013, S. 30). In dieser Bachelorarbeit werden N-Gramme als eine Sequenz aus Wörtern betrachtet. Fragmente bzw. Wortphrasen, die aus zwei und drei Wörtern zusammengesetzt werden, heißen Bi- und Trigramme (Heyer, Quasthoff und Wittig 2008, S. 105). Aus einem Satz S mit einer Anzahl an Wörtern w ergeben sich beispielsweise folgende Trigramme:

$$S(w_1 \ w_2 \ w_3 \ w_4 \ w_5) \rightarrow (w_1 w_2 w_3, w_2 w_3 w_4, w_3 w_4 w_5)$$

3.5 Stammformreduktion

Um die Anzahl an Tokens innerhalb eines Textes zu reduzieren und sie anhand ihres Wortstammes vergleichbar zu machen, kann eine Stammformreduktion durchgeführt werden. Dabei werden, durch die Anwendung von verschiedenen Regeln, die Präfixe und Suffixe der Wörter entfernt, sodass nur noch die Wortstämme übrigbleiben (Ignatov und Mihalcea 2017, S. 55). Ein bekannter Algorithmus dafür ist der Porter- Stemmer-Algorithmus (Porter 1980), welcher die Endungen eines Wortes iterativ auf immer kürzere Buchstabensequenzen reduziert. Des Weiteren werden bei diesem Prozess Pluralformen in Singularformen umgewandelt (Taeho 2018, S. 66).

Beispiel:	gefährlich	wird zu	Gefahr
	Hunde	wird zu	Hund

4 Automatische Textklassifizierung

Aufgrund der rasant anwachsenden Fülle an Informationen, die durch die Digitalisierung vorangetrieben wird, ist es für den Menschen unmöglich geworden, alle eingehenden Daten zu verstehen oder in Kategorien zu klassifizieren. Mit diesem Informationszuwachs und dem gleichzeitigen Anstieg der verfügbaren Rechenleistung von Computern, gewinnt die automatische Klassifizierung von Daten, insbesondere von Textdaten, zunehmend an Bedeutung (Mahinovs und Tiwari 2007, S. 1). Gemäß Sebastini (2005, S. 109) kann Textklassifizierung auch als Textkategorisierung bezeichnet werden. Beide Begriffe werden daher in dieser Arbeit mit der identischen Bedeutung eingesetzt.

Dieses Kapitel ist in drei Teile gegliedert, wobei im ersten Teil die Textklassifizierung definiert wird. Der zweite Teil verschafft einen Überblick über die aktuell angewandten methodischen Ansätze der Textklassifizierung. Zum Schluss wird der Prozess der automatisierten Textklassifikation für einen überwachten Ansatz beschrieben.

4.1 Definition

Sebastiani (2002, S. 1) definiert die Textklassifikation als einen Prozess, welcher Dokumente automatisch in vorbestimmte Kategorien unterteilt. Aus einer mehr formalen Sicht definieren Ikonomakis, Kotsiantis und Tampakas (2005, S. 966) den Prozess so, dass wenn di ein Dokument eines Dokumentensatzes D ist und $\{c1, c2, c3, \dots, cn\}$ ein Satz aller Kategorien darstellt, dann wird durch die Textklassifikation einem Dokument di eine oder mehrere Kategorien cj zugeordnet. Die Kategorien können dabei beispielsweise einfache ja/nein Entscheidungen (Ingersoll, Morton und Farris 2013, S. 177), thematische Bereiche welche mit einem Wort beschrieben werden können (Ingersoll, Morton und Farris 2013, S. 177) oder komplexere zusammenhängende hierarchische Verzweigungen (Ignatov und Mihalcea 2017, S. 117 - 118) repräsentieren.

Laut Ignatov und Mihalcea (2017, S. 119 ff) handelt es sich bei der Textklassifikation um eines der am meisten genutzten Text-Mining Verfahren, welches zur Lösung einer Vielzahl von Problemen genutzt wird. Beispiele dafür sind:

- Themen Klassifikation (McCallum und Nigam 1998)
- E-Mail Spam Filter (Patidar, Singh und Singh 2013)
- Stimmungsanalyse (Wiebe, Wilson und Cardie 2005, Pang und Lee 2008)
- Geschlechter Klassifikation (Koppel, Argamon und Shimoni 2002, Liu und Mihalcea 2007)

4.2 Methodologie

Um einen Überblick über die aktuell angewandten Methoden für die Textklassifizierung zu bekommen, haben Thangaraj und Sivakami (2018, S. 120) anhand von 74 wissenschaftlichen Artikeln zu dem Thema aus den Jahren 2010-2017 eine Baumstruktur erstellt. Das Ergebnis ihrer Arbeit ist in Abbildung 3 dargestellt. Demnach lassen sich die aktuellen Methoden grob in die statistischen und die des maschinellen Lernens unterteilen, wobei letztere in die Bereiche des überwachten Lernens, unbeaufsichtigten Lernens und semi-überwachtes Lernens gegliedert werden. Jedem Bereich werden wiederum verschiedene Algorithmen zugeordnet, die in der untersten Ebene der Baumstruktur erkennbar sind. Dabei werden die Algorithmen für das überwachte Lernen, abhängig von einer fixen oder dynamischen Anzahl an Parametern, in parametrisch und nicht parametrisch kategorisiert.

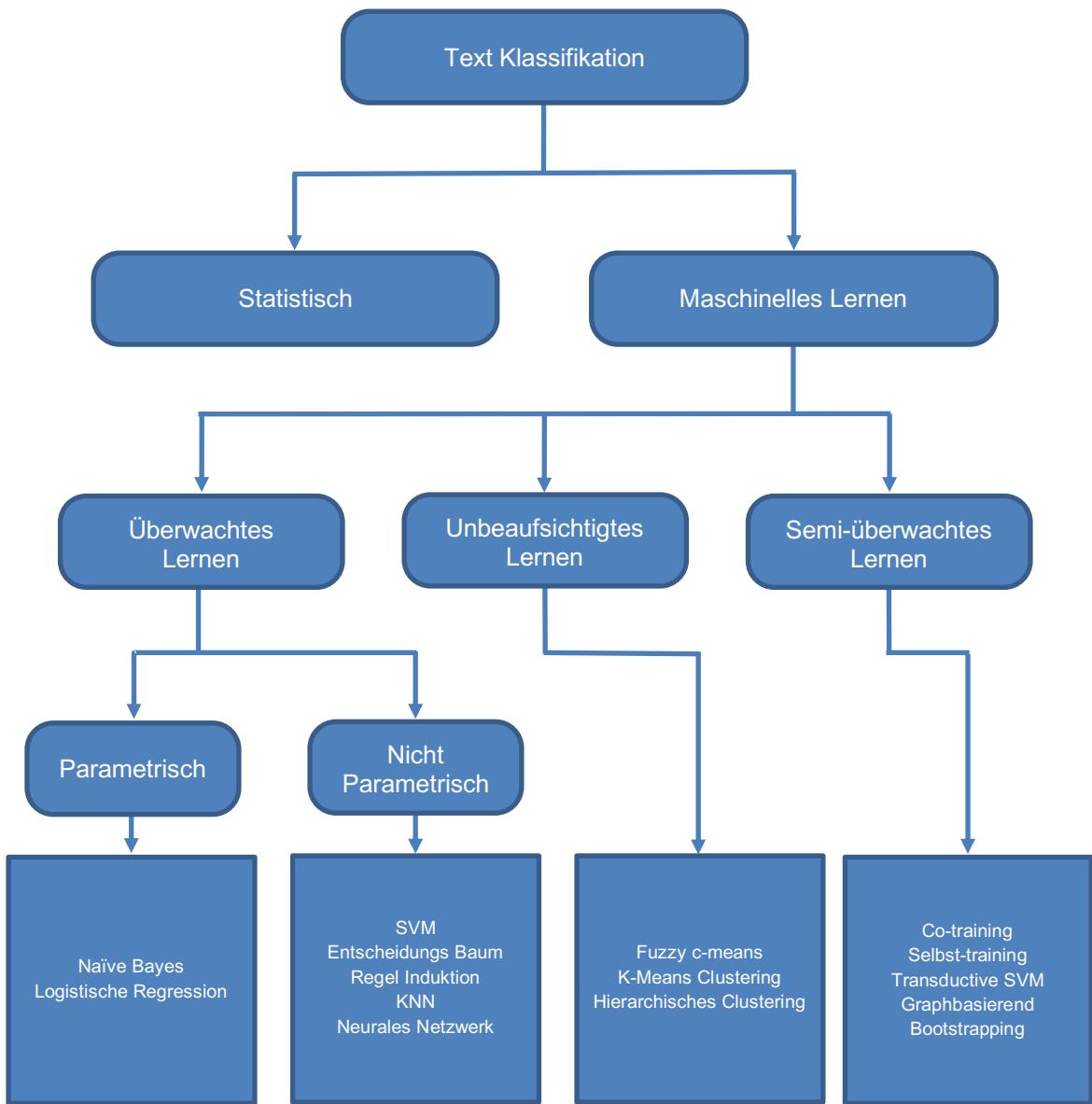


Abbildung 3 Methoden Textklassifizierung. (Quelle: Eigene Darstellung in Anlehnung an Thangaraj und Sivakami (2018, S. 120))

Statistischer Ansatz

Gemäß Srivastava (2015) handelt es sich bei dem statistischen Ansatz der Textklassifizierung um einen rein mathematischen Prozess, der, ähnlich wie ein Computerprogramm, die gegebenen Anweisungen ohne eigene Fähigkeiten ausführt. Gemäß Michie, Spiegelhalter und Taylor (1994, S. 2) liegt dem statistischen Ansatz ein Wahrscheinlichkeitsmodell zu Grunde, anhand dessen Klassenzugehörigkeiten bestimmt werden. Um gute Ergebnisse zu erzielen, müssen laut Vieira, Borrajo und Iglesias (2016, S. 119) die für den Prozess benötigten Informationen präzise definiert werden, was durch eine Reduktion der Daten-Dimensionalität erreicht wird. Beispielsweise können die Informationen in einem Datenset über Volkszählungen auf das Alter, den Ort und das Geschlecht reduziert werden. Aufgrund der Einschränkung auf wenige Informationen, beschreibt Srivastava (2015) den rein statistischen gegenüber dem des maschinellen Lernens als einen begrenzten Ansatz.

Ansatz des maschinellen Lernens

Gemäß Thangaraj und Sivakami (2018, S. 121) wurden die Techniken des maschinellen Lernens speziell für die Automatisierung von Problemlösungen entwickelt. Diese Automatisierung beruht bei der Textklassifizierung auf der Grundlage von Expertenmeinungen zur Kategorisierung von themenspezifischen Dokumenten, sowie der Definition eines auf Wissensverarbeitungstechniken aufbauenden, logischen Regelwerkes (Korde und Mahender 2012, S. 85). Die Textklassifizierung durch maschinelles Lernen kann in drei Kategorien unterteilt werden: überwachten Lernens, unbeaufsichtigten Lernen und semi-überwachtes Lernen (Dwivedi und Arya 2016).

Überwachtes Lernen

Gupta (2018) beschreibt das überwachte Lernen als einen Prozess des Schulungs- und Testprinzips. Dabei werden vorab kategorisierte Trainingsdaten genutzt, um mittels Algorithmen neue Daten zu kategorisieren. Gemäß Thangaraj und Sivakami (2018, S. 122) handelt es sich bei dem überwachten Lernen um das zeitintensivste der drei Verfahren des maschinellen Lernens, da ein großer Aufwand für das manuelle Zuweisen von Kategorien für die Trainingsdaten betrieben werden muss.

Unbeaufsichtigtes Lernen

Shafabady et al. (2016, S. 4) definieren das unbeaufsichtigte Lernen als einen Prozess, bei dem die zu klassifizierenden Daten, durch das in Abschnitt 2.2 beschriebene Text Clustering, anhand ihrer Ähnlichkeit in Kategorien bzw. Cluster eingeordnet werden. Dabei werden keine externen Daten oder eine manuelle Vorarbeit benötigt. Meist ist das Expertenwissen für die eindeutige Benennung dieser Cluster jedoch nicht vorhanden oder unzureichend. In diesem Fall können beispielsweise Kartendarstellungen und Korrelationskoeffizienten verwendet werden, um die „gebündelten“ Dokumentenfelder zu betrachten und zu kategorisieren. Ko und Seo (2000, S. 453) nennen einen weiteren Ansatz der Kategorienbeschriftung mittels im Text enthaltene Schlüsselwörter.

Semi-überwachtes Lernen

Diese Art des Lernens wird gemäß Altinel und Ganiz (2016, S. 50) dann angewandt, wenn eine geringe Menge an vorkategorisierten Trainingsdaten und eine große Menge an nicht kategorisierten Daten vorhanden ist. Aufgrund der geringen Anzahl an Trainingsdaten ist es sehr wahrscheinlich, dass die Klassifizierung der nicht kategorisierten Daten unbefriedigend ist. Um diese Qualität der Kategorisierung zu steigern, werden vorab die vorkategorisierten und nicht kategorisierten Daten kombiniert betrachtet, um durch verschiedene Algorithmen wie Co-training, Selbst-training oder Bootstrapping die Anzahl an Trainingsdaten in einer Art „Selbst Lernprozess“ zu erhöhen.

4.3 Prozessbeschreibung

In diesem Abschnitt wird der Prozess der automatisierten Textklassifikation erläutert. Die Prozessschritte für die Textklassifikation können nicht alle klar voneinander abgegrenzt werden, wodurch diese in der Anzahl und Beschreibung in diversen wissenschaftlichen Publikationen unterschiedlich definiert werden (Ingersoll, Morton und Farris 2013, S. 180, Ikonomakis, Kotsiantis und Tampakas 2005, Korde und Mahender 2012, Jindal, Malhotra und Jain 2015, Mahinovs und Tiwari 2007). Auf Grundlage dieser Informationen werden die Prozessschritte in dieser Arbeit, wie in Abbildung 4 dargestellt, beschrieben.

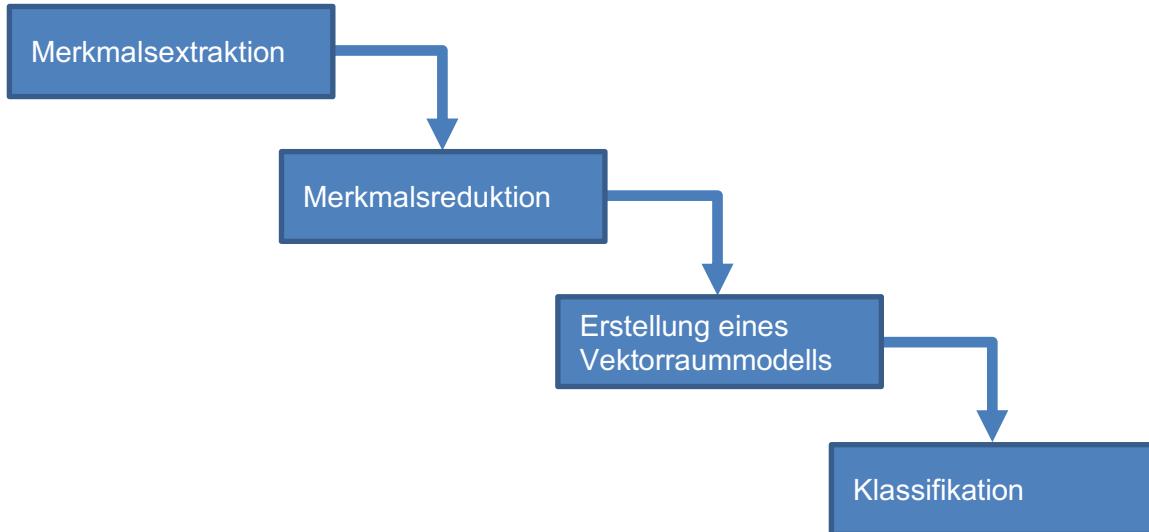


Abbildung 4 Prozessbeschreibung Textklassifikation. (Quelle: Eigene Darstellung)

Dabei wird in vier Prozessschritten vorgegangen. Im ersten Schritt werden die Merkmale aus den Dokumenten extrahiert. Mahinovs und Tiwari (2007, S. 5) beschreiben diesen als einen Schritt, bei dem der Text in Zahlen umgewandelt wird, damit später mittels mathematischer Methoden die Klassifizierung stattfinden kann. Im Anschluss werden die Merkmale reduziert und ein Vektorraummodell erstellt. Anhand des Vektorraummodells erfolgt im letzten Schritt, mithilfe eines Klassifikationsalgorithmus, die Klassifizierung der Dokumente, wofür beispielhaft ein Algorithmus genauer definiert wird (Jindal, Malhotra und Jain 2015, S. 7).

4.3.1 Merkmalsextraktion

Laut Korde und Mahender (2012, S. 86) werden anfangs Textvorverarbeitungsschritte durchgeführt, um die Dokumente in ein klares Wortformat zu überführen. Dabei werden gemäß Allahyari, et al. (2017, S. 3) die folgenden Schritte üblicherweise realisiert:

- Tokenisierung (Vergleich Unterabschnitt 3.1 Tokenisierung)
- Entfernung von Stoppwörtern (Vergleich Unterabschnitt 3.2 Stopwörter)
- Stammformreduktion (Vergleich Unterabschnitt 3.5 Stammformreduktion)

Die daraus resultierenden Tokens werden als Merkmale des Dokumentes bezeichnet (Ingersoll, Morton und Farris 2013, S. 182). Diese Merkmale werden üblicherweise in einem Bag-Of-Words Ansatz präsentiert, wobei jedem Merkmal die Häufigkeit seines Vorkommens als Gewichtung zugewiesen wird. Dokumente lassen sich durch diese Darstellung einfach als eine Sammlung von Wörtern betrachten, in der jedes Wort mindestens einmal vorkommt (Jindal, Malhotra und Jain 2015, S. 5). Folglich können einzelne Dokumente oder eine Kollektion von Trainingsdokumenten, wie in Tabelle 1 sichtbar, dargestellt werden, wobei W1-7 beispielhaft für die Wortmerkmale steht.

	W1	W2	W3	W4	W5	W6	W7
Dokument 1	2	1	3	2	1	1	4
	W1	W2	W3	W4	W5	W6	W7
Dokument 1	2	1	3	1	2	0	0
Dokument 2	1	2	0	0	0	2	3

Tabelle 1 Bag-Of-Words Darstellung. (Quelle: Eigene Tabelle)

4.3.2 Merkmalsreduktion

Aufgrund einer, wie in Tabelle 1 gezeigten Bag-Of-Words Herangehensweise, ist es sehr einfach möglich, dass mittels einer relativ kleinen Menge an Dokumenten eine sehr große Bag-Of-Words Repräsentation entsteht. Auch wenn bereits im Vorverarbeitungsschritt beispielsweise die Stopwörter entfernt oder alle Wörter auf ihre Stammform reduziert wurden, ist die Anzahl an Merkmalsdimensionen noch sehr hoch. Daher ist eine weitere Merkmalsreduktion notwendig. Dafür können diverse Bewertungsfunktionen für jedes Wortmerkmal angewandt werden, aus deren Ergebnissen ein Score für die Wichtigkeit eines einzelnen Merkmals resultiert (Jindal, Malhotra und Jain 2015, S. 6). Daraufhin wird laut Ikonomakis, Kotsiantis und Tampakas (2005, S. 967-968) aus einer vordefinierten Anzahl der höchsten Scores eine Teilmenge der besten Merkmale gebildet. Bekannte Bewertungsfunktionen sind dabei:

- **Dokumentenhäufigkeit**
Es wird berechnet, in wie vielen Dokumenten das Merkmal auftritt (Forman 2003, S. 1294).
- **Merkmalshäufigkeit**
Es wird berechnet, wie oft ein Merkmal in den Dokumenten vorkommt (Azam und Yao 2012, S. 4760).
- **χ^2 Static**
 χ^2 Static (CHI) wird gemäß Yang und Pedersen (1997, S. 415) wie folgt definiert:

$$\chi^2(t, c) = \frac{N \cdot (AD - CB)^2}{(A + C) \cdot (B + D) \cdot (A + B) \cdot (C + D)} \quad (1)$$

wobei

- N die Anzahl aller Dokumente,
- A die Anzahl aller Dokumente der Kategorie c welche das Merkmal t enthalten,
- B die Anzahl der Dokumente der anderen Kategorien (nicht c) welche das Merkmal t enthalten,
- C die Anzahl der Dokumente der Kategorie c welche nicht das Merkmal t enthalten und
- D die Anzahl der Dokumente der anderen Kategorien welche nicht das Merkmal t enthalten

repräsentiert. χ^2 Static (CHI) misst also den Mangel an Unabhängigkeit zwischen t und c. Dementsprechend ist die Ermittlung von χ^2 Static (CHI) nur in überwachten oder Semi-überwachten Ansätzen möglich, da die Berechnung der Formel bereits zugewiesene Kategorien benötigt.

- **Gini Index**

Aggarwal und Zhai (2012, S. 168) beschreiben den Gini Index als eine gebräuchliche Bewertungsfunktion zur Messung des Unterscheidungsgrades von Merkmalen. Dabei wird vorab definiert, dass:

$$\sum_{i=1}^k p_i(w) = 1 \quad (2)$$

wobei $p_1(w) \dots p_k(w)$ den Anteil von Kategorien darstellt, in denen das Merkmal w vorkommt. Anders gesagt ist $p_i(w)$ die bedingte Wahrscheinlichkeit, dass ein Dokument zur Kategorie i gehört, da es das Merkmal w enthält.

Darauffolgend kann der Gini Index für das Merkmal w wie folgt berechnet werden:

$$G(w) = \sum_{i=1}^k p_i(w)^2 \quad (3)$$

wobei das Ergebnis immer zwischen $\frac{1}{k}$ und 1 liegen muss. Wenn das Ergebnis also 1 ist, gehören dementsprechend alle Dokumente, die das Merkmal w beinhalten, zu derselben Kategorie. Ansonsten ergibt der Gini Index den anteiligen Wert des Vorkommens. Dementsprechend ist die Ermittlung des Gini Index nur in überwachten oder Semi-überwachten Ansätzen möglich, da die Berechnung der Formel bereits zugewiesene Kategorien benötigt.

Forman (2003, S. 1292-1294) erwähnt zusätzlich weitere Bewertungsfunktionen wie Information Gain, Odds Ratio und Probability Ratio.

Bei der Auswahl der angewandten Bewertungsfunktion ist darauf zu achten, dass nur irrelevante Merkmale aussortiert werden, damit dies keinen negativen Auswirkungen auf das spätere Klassifikationsergebnis hat (Ikonomakis, Kotsiantis und Tampakas 2005). Der Vorteil, welcher durch die Reduktion des Datensets entsteht, ist die beträchtliche Verkleinerung des Suchraumes und eine damit verbundene geringere Rechenanforderung für den Klassifizierungsprozess (Jindal, Malhotra und Jain 2015, S. 6).

4.3.3 Erstellung eines Vektorraummodells

Im Anschluss an die Merkmalsreduktion wird gemäß Jindal, Malhotra und Jain (2015, S. 6) das Vektorraummodell erstellt. Dieses wird, ähnlich dem Bag-Of-Words Ansatz, als eine Wortmerkmal/Dokumenten Matrix dargestellt, bestehend aus allen Vektoren der einzelnen Dokumente sowie allen nach der Reduktion übrig gebliebenen Merkmalen. Diese Merkmale werden dabei in dem Vektorraummodell als Dimensionen bezeichnet (Salton, Wong und Yang 1975, S. 613).

Korde und Mahender (2012, S. 87) stellen das Vektorraummodell wie folgt vor:

$$\begin{pmatrix} D_i & T_1 & T_2 & \cdots & T_t & C_i \\ D_1 & w_{11} & w_{21} & \cdots & w_{t1} & C_j \\ D_2 & w_{12} & w_{22} & \cdots & w_{t2} & C_k \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ D_n & w_{1n} & w_{2n} & \cdots & w_{tn} & C_n \end{pmatrix} \quad (4)$$

wobei D_i alle Dokumente, $T_1 \dots T_t$ alle Merkmalsdimension und C_i die zu den Dokumenten zugeordneten Kategorien repräsentieren. Falls die Dokumente jedoch noch keine zugeordnete Kategorie besitzen, entfällt C_i . Außerdem wird für jedes Merkmal t in dem Dokument n eine Gewichtung w_{tn} berechnet. Die Gewichtung kann beispielsweise mittels bekannter Methoden wie:

- **Binärgewicht**
Gemäß Ignatov und Mihalcea (2017, S. 123) handelt es sich bei dieser Gewichtung um die einfachste, da nur unterschieden wird, ob ein Merkmal in einem Dokument vorkommt oder nicht. Falls es vorkommt wird das Gewicht mit 1 bestimmt, andernfalls mit 0.
- **Merkmalshäufigkeit**
Es wird berechnet, wie oft ein Wort in dem Dokument vorkommt.
- **TF-IDF**
Gemäß Ingersoll, Morton und Farris (2013, S. 182) berechnet TF-IDF die Wichtigkeit eines Merkmals in einem Dokument, in Abhängigkeit davon, wie oft ein Merkmal in allen Dokumenten vorkommt. Dabei wird die Merkmalshäufigkeit durch die Anzahl an Dokumenten, in dem das Merkmal außerdem existiert, geteilt (Ignatov und Mihalcea 2017, S. 123-124).

berechnet werden.

4.3.4 Klassifikation

Im Anschluss kann des Vektorraummodell mit einem überwachten, semi-überwachten oder unbeaufsichtigten Klassifikationsalgorithmus genutzt werden, um die Kategorie für Dokumente zu bestimmen (Korde und Mahender 2012, S. 87). Ikonomakis, Kotsiantis und Tampakas (2005, S. 971) nennen dabei Naïve Bayes, Support Vector Machines, Nearest Neighbor Classifier und den Entscheidungsbaum Classifier als die bekanntesten Algorithmen für die Textklassifikation. Da sich diese Algorithmen dem überwachten Lernen zuordnen lassen ist es notwendig, dass das neue Dokument vorab den Schritt der Merkmalsextraktion durchläuft, damit die Schnittmenge derer Merkmale und der des Vektorraummodells verglichen werden können (Sebastiani 1999).

In diesem Unterabschnitt wird der Naïve Bayes Algorithmus beispielhaft genauer erläutert.

Naïve Bayes

Bei Naïve Bayes handelt es sich gemäß Aggarwal und Zhai (2012, S. 182) um einen der einfachsten und am meisten genutzten Textklassifikationsalgorithmus. Er basiert auf dem Satz von Bayes aus der Wahrscheinlichkeitstheorie, der die Berechnung der bedingten Wahrscheinlichkeit beschreibt (Ignatov und Mihalcea 2017, S. 124).

Gemäß Ingersoll, Morton und Farris (Ingersoll, Morton und Farris 2013, S. 202-204) wird bei der Klassifizierung von neuen Dokumenten die Entscheidung, welche Kategorie diesem zugeordnet werden soll, anhand von Wahrscheinlichkeiten aus dem Vektorraummodell abgeleitet. Dabei werden die Beziehungen zwischen den Dokumentenmerkmalen und deren Kategorien, sowie die Beziehung der Kategorien zum kompletten Vektorraum betrachtet. Diese Beziehungen werden als bedingte Wahrscheinlichkeiten wie folgt berechnet:

- **P (Merkmal | Kategorie)**
Dieser Wahrscheinlichkeitswert wird für jedes Merkmal anhand aller Kategorien berechnet. Dabei wird die Anzahl des Merkmalvorkommens in allen Dokumenten einer Kategorie durch die gesamte Anzahl dieser Kategorie geteilt. Ausgehend von einem Vektorraummodell mit drei Dokumenten welchen die Kategorie A zugewiesen ist, würde beispielsweise für ein Merkmal w, welches nur in einem der drei Dokumente vorkommt, ein Wahrscheinlichkeitswert von 33% berechnet werden.

- **P (Kategorie)**

Dieser Wert berechnet, wie wahrscheinlich es ist, dass eine Kategorie einem neuen Dokument zugewiesen wird. Dabei wird die Anzahl des Vorkommens einer Kategorie A durch die Gesamtzahl an Dokumenten bzw. Kategorien gerechnet. Wenn beispielsweise ein Vektorraum mit drei Dokumenten gegeben ist, wobei zwei Dokumente der Kategorie A und ein Dokument der Kategorie B zugeordnet sind, beträgt die Wahrscheinlichkeit, dass ein neues Dokument der Kategorie A zugeordnet wird, 66%.

Auf der Grundlage dieser Wahrscheinlichkeitswerte, lässt sich für ein neues Dokument die folgende Formel anhand jeder Kategorie berechnen:

$$P(Kategorie_c | Dokument_d) = \frac{P(Dokument_d | Kategorie_c) \cdot P(Kategorie_c)}{P(Dokument_d)} \quad (5)$$

wobei mit „Dokument“ in diesem Falle die Merkmale W des neuen, zu kategorisierenden Dokumentes gemeint sind, wodurch sich $P(Dokument | Kategorie)$ gemäß Lewis (1998) aus der Multiplikation aller im neuen Dokument enthaltenen $P(Merkmal | Kategorie)$ Wahrscheinlichkeiten wie folgt berechnen lässt:

$$P(Dokument_d | Kategorie_c) = \prod_{j=1}^d P(Merkmal_j | Kategorie_c) \quad (6)$$

Der Wert $P(Dokument_d)$ aus Formel (5) lässt sich gemäß Lewis (1998) mittels der Formel:

$$P(Dokument_d) = \sum_{k=1}^C P(Dokument_d | Kategorie_k) \cdot P(Kategorie_k) \quad (7)$$

berechnen, wobei dabei alle Ergebnisse von $P(Dokument_d | Kategorie_k) \cdot P(Kategorie_k)$ für jede Kategorie summiert werden. Dieser Wert ist für die Normalisierung der jeweiligen Ergebnisse pro Kategorie notwendig. Anhand der berechneten normalisierten Wahrscheinlichkeitswerte für das neue Dokument pro Kategorie, kann bestimmt werden, welche Kategorie sich dem Dokument am wahrscheinlichsten zuordnen lässt.

Beispiel

Der Vorgang lässt sich anhand eines Beispiels verdeutlichen. Gegeben ist ein Vektorraummodell bestehend aus sieben Dokumenten und vier Merkmalsdimensionen. Jedes Dokument enthält also maximal vier verschiedene Merkmale, wobei diese w1, w2, w3 und w4 sein können. Zusätzlich ist jedem Trainingsdokument eine der beiden Kategorien A oder B zugeordnet. Dieses Vektorraummodell ist grafisch durch Tabelle 2 dargestellt.

Trainings-dokument	w1	w2	w3	w4	Kategorie
1	1	1	1	1	A
2	1	1	0	0	A
3	1	1	0	0	A
4	1	0	1	0	A
5	0	1	0	1	B
6	1	0	1	1	B
7	0	1	1	1	B

Tabelle 2 Beispiel Textklassifizierung Vektorraummodell. (Quelle: Eigene Tabelle)

Um das Beispiel einfach zu halten, wird von einer Binärgewichtung für die jeweiligen Merkmale ausgegangen. Anhand der Ausgangswerte lassen sich die jeweiligen Wahrscheinlichkeiten berechnen. Aufgrund der Tatsache, dass vier von sieben Trainingsdokumenten der Kategorie A angehören, beträgt die Kategorie Wahrscheinlichkeit $P(A) = \frac{4}{7} = 0,57$. Des Weiteren lässt sich berechnen, wie wahrscheinlich es ist, dass das Merkmal w1 in Dokumenten vorkommt, welche der Kategorie B angehören. Da nur eins der drei als B kategorisierten Dokumente das Merkmal w1 beinhaltet, ergibt sich eine Wahrscheinlichkeit von $\frac{1}{3} = 0,33$. Die Berechnungen für alle Möglichkeiten ergeben die Resultate in Tabelle 3.

	Kategorie A	Kategorie B
$P(\text{Kategorie})$	0,57	0,43
$P(w_1 \text{Kategorie})$	1,00	0,33
$P(w_2 \text{Kategorie})$	0,75	0,67
$P(w_3 \text{Kategorie})$	0,50	0,67
$P(w_4 \text{Kategorie})$	0,25	1,00

Tabelle 3 Beispiel Textklassifikation Wahrscheinlichkeitstabelle. (Quelle: Eigene Tabelle)

Die Resultate aus Tabelle 3 werden für die Klassifizierung neuer und unbekannter Dokumente genutzt. Unter der Annahme, dass für ein neues Dokument D, welches die Merkmale w1, w3 und w4 enthält, eine Kategorie bestimmt werden soll, werden anhand der Formeln (5) und (6) die Wahrscheinlichkeiten für beide Kategorien A und B berechnet.

$$P(K = A|D) = \frac{(1 \cdot (1 - 0,75) \cdot 0,5 \cdot 0,25) \cdot 0,57}{P(D)} = \frac{0,018}{P(D)}$$

$$P(K = B|D) = \frac{(0,33 \cdot (1 - 0,67) \cdot 0,67 \cdot 1) \cdot 0,43}{P(D)} = \frac{0,031}{P(D)}$$

Die Subtraktionen $(1 - 0,75)$ und $(1 - 0,67)$ liegen darin begründet, dass das Merkmal w2 nicht in dem Dokument D vorkommt und dementsprechend die Gegenteile, der in Tabelle 3 resultierenden Ergebnisse, berechnet werden müssen.

Die berechneten Ergebnisse werden wie folgt normalisiert:

$$P(D) = 0,018 + 0,031 = 0,049$$

$$P(K = A|D) = \frac{0,018}{0,049} = 0,37$$

$$P(K = B|D) = \frac{0,031}{0,049} = 0,63$$

Das Ergebnis drückt also aus, dass Dokument D mit einer Wahrscheinlichkeit von 37% der Kategorie A und mit einer Wahrscheinlichkeit von 63% der Kategorie B angehört.

5 Automatische Schlüsselwort-Extraktion

Um möglichst effizient einen Überblick über die Inhalte einer Vielzahl an Dokumenten zu erhalten bedarf es eines automatisierten Systems, das in der Lage ist nur deren relevantere Informationen zu extrahieren. Da dies im praktischen Teil dieser Bachelorarbeit eine große Rolle spielt, wird in diesem Kapitel auf die automatisierte Schlüsselwort-Extraktion eingegangen.

Dieses Kapitel ist in vier Teile gegliedert, wobei im ersten Teil die automatische Schlüsselwort-Extraktion definiert wird. Der zweite Teil verschafft einen Überblick über die aktuell angewandten methodischen Ansätze der Schlüsselwort-Extraktion. Darauf folgt die Prozessbeschreibung des allgemeinen Vorgehens. Zum Schluss wird der theoretische Aufbau dreier unterschiedlicher Schlüsselwort-Extraktionsmethoden beschrieben, welche auch im praktischen Teil dieser Ausarbeitung angewandt werden.

5.1 Definition

Beim Verfahren der automatisierten Schlüsselwort-Extraktion werden Schlüsselworte (englisch keywords) und Schlüsselwortphrasen (englisch keyword phrases) ohne manuellem Zutun aus einem Textdokument ermittelt. Diese drücken die Hauptthemen des Dokumentes aus (Zhang, et al. 2008, S. 1169). Taeho (2018, S. 320) beschreibt die Schlüsselwort-Extraktion als ein Verfahren, bei dem jene wichtigen Wörter aus einem Text extrahiert werden, die seinen gesamten Inhalt repräsentieren. Hierbei ist es wichtig, das Extraktionsverfahren von dem Bestimmungsverfahren abzugrenzen, da bei ersterem ein Schlüsselwort anhand von im Ausgangstext befindlichen Wörtern bestimmt wird und bei letzterem durch ein klassifiziertes Vokabular an Begriffen (Beliga 2014, Witten, et al. 1999).

Ziel ist es somit, einen Satz an Schlüsselwörtern zu extrahieren die sich auf das im Dokument diskutierte Hauptthema beziehen. Die ermittelten Schlüsselwörter können im Anschluss dafür verwendet werden, aus einer großen Anzahl an Dokumenten schnell und akkurat einzelne Dokumente herauszufiltern. Des Weiteren können Schlüsselwörter für zahlreiche weitere Textverarbeitungsaufgaben verwendet werden, beispielsweise:

- Zur Erstellung von Textzusammenfassungen (Zhang, Zincir-Heywood und Milios 2004)
- Textkategorisierung (Hulth und Megyesi 2006)
- Opinion Mining (Berend 2011)
- Dokumentenindexierung (Gutwin, et al. 1999)

5.2 Methodologie

In diesem Abschnitt wird ein Überblick über aktuelle Schlüsselwort-Extraktionsmethoden vermittelt. Dabei können diese anfangs grob in die Kategorien „überwacht“ (englisch supervised) und „unbeaufsichtigt“ (englisch unsupervised) eingeteilt werden (Beliga 2014, S. 1).

Überwachter Ansatz

Das Ziel von überwachten Schlüsselwort-Extraktionsmethoden ist es, ein Modell anhand von mehreren bereits manuell mit Schlüsselworten versehenen Trainingsdokumenten zu erstellen. Dieses Modell wird im Anschluss genutzt, um Schlüsselwortphrasen in neuen Dokumenten zu ermitteln. Hierfür ist es notwendig, vorab Merkmale für jeden potentiellen Kandidaten zu bestimmen, was auf der Grundlage von beispielsweise statistischen (Witten, et al. 1999, Frank, et al. 1999), linguistischen (Yih, Goodman und Carvalho. 2006) oder auf

externen Ressourcen basierenden (Medelyan, Frank und Witten 2009) Verfahren durchgeführt werden kann (Hasan und Ng 2014). Die Qualität der Ergebnisse ist abhängig vom Trainingsdatensatz und funktioniert aufgrund dessen meist nur für ein textuelles Themengebiet. Dies bedeutet, dass ein auf wissenschaftlichen Dokumenten trainiertes Modell bei Texten über beispielsweise Kinofilme wahrscheinlich keine guten Schlüsselwortphrasen finden wird (Bennani-Smires, et al. 2018, S. 1).

Unbeaufsichtigter Ansatz

Der unbeaufsichtigte Ansatz der Schlüsselwort-Extraktion benötigt keinerlei Trainingsdaten, da er alle benötigten Informationen in dem betrachteten Textdokument selbst findet. Dabei kann die Erkennung von Schlüsselwortphrasen anhand von beispielsweise statistischen (Campos, Mangaravite, et al. 2018a), linguistischen (Tomoko und Hurst 2003) oder auf Graphen basierenden (Mihalcea und Tarau 2004) Verfahren ermittelt werden. Der Vorteil dieses Ansatzes ist seine vom Themenbereich des Textes unabhängige Anwendbarkeit und das hierzu keine Vorarbeit benötigt wird (Bennani-Smires, et al. 2018, S. 1).

Beide Ansätze nutzen verschiedene Methoden, die gemäß Zhang et al. (2008, S. 1170-1171) in vier Kategorien: Statistisch, Sprachlich, Maschinelles Lernen und Andere eingeteilt werden können. Unter dem Gesichtspunkt der aktuellen Arbeiten zu diesem Thema hat Beliga (2014, S. 2 ff) erkannt, dass die auf Graphen basierenden Methoden auch unter den Hauptkategorien erwähnt werden sollten. Abbildung 5 zeigt eine daraus resultierende Übersicht der verschiedenen algorithmischen Verfahren und derer Methoden, welche einzeln oder in Kombination angewandt werden.

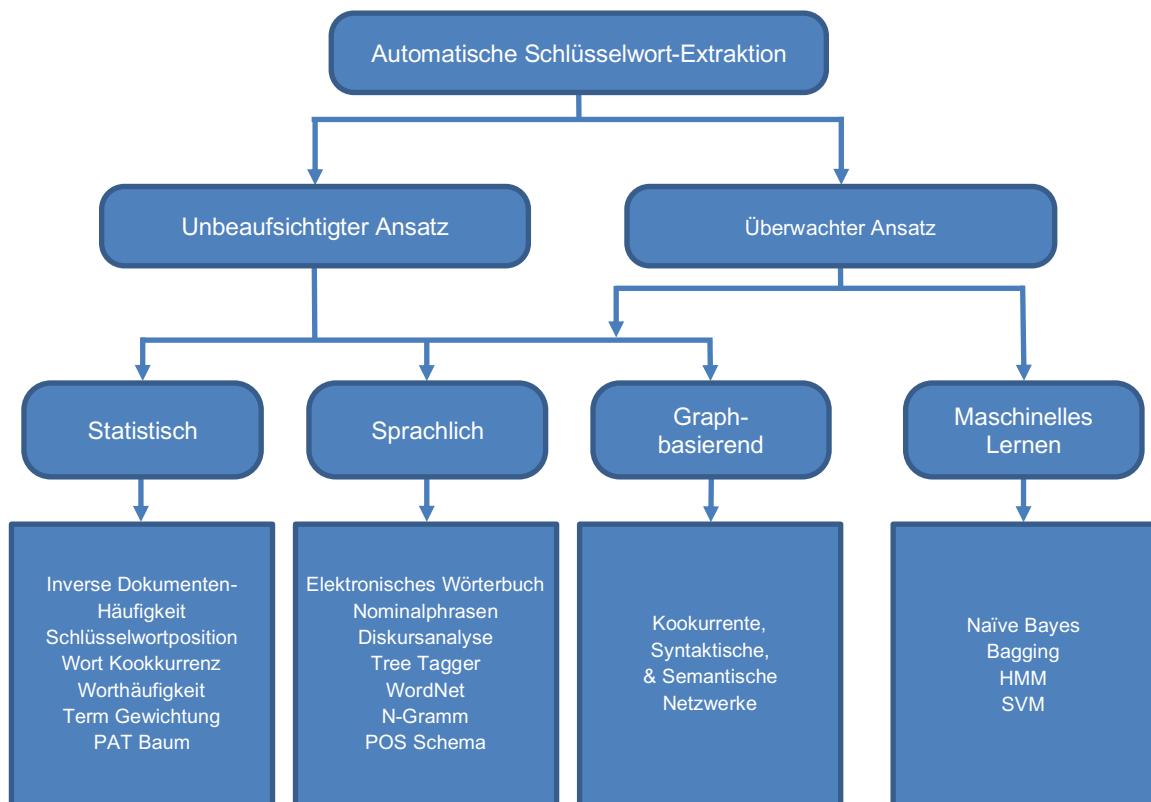


Abbildung 5 Kategorisierung der automatischen Schlüsselwort-Extraktion. (Quelle: Eigene Darstellung in Anlehnung an Bharti, Babu und Jena (2017, S. 2))

Statistisch

Gemäß Bharti, Babu und Jena (2017, S. 2) beinhalten die statistischen Verfahren einfache Methoden, die keine Trainingsdaten benötigen und unabhängig von der Sprache sowie dem Textthema angewandt werden können. Dabei lassen sich Werte wie zum Beispiel die Worthäufigkeit (Luhn 1957), die inverse Dokumentenhäufigkeit (Salton, Yang und Yu 1975), die Wort Kookkurrenz (Matsuo und Ishizuka 2004) und die Position eines Wortes im Dokument ermittelt. Anhand dieser Kennzahlen kann eine Liste mit Schlüsselwörtern erstellt werden. Nachteil dieser statistisch gestützten Methoden ist, dass beispielsweise in Fachtexten des medizinischen Bereiches das wichtigste Schlüsselwort nur einmal vorkommen kann. Dieses wird jedoch später aufgrund der niedrigen Kennzahlen nicht als wichtiges Schlüsselwort eingestuft (Chen und Lin 2010, S. 1928).

Sprachlich

Bei diesen Methoden werden gemäß Zhang (2008, S. 1170) die sprachlichen Merkmale von Wörtern, Wortphrasen, Sätzen und Dokumenten zur Erkennung und Extraktion von Schlüsselwörtern genutzt. Hierfür werden hauptsächlich lexikalische Analysen (Barzilay und Elhadad 1997), syntaktische Analysen (Hulth 2003) und Diskursanalysen (Salton, Singhal, et al. 1997) angewandt, wobei diese sehr komplex sind. Die dabei für die lexikalische Analyse verwendeten Ressourcen sind unter anderen das elektronische Wörterbuch, Tree-Tagger, WordNet, N-Gramm und Part-Of-Speech Muster. Die syntaktische Analyse wird anhand von Nominalphrasen-Chunks durchgeführt (Bharti, Babu und Jena 2017, S. 2).

Graphbasierend

Gemäß Hasan und Ng (2014, S. 1265) wird die Wichtigkeit von Schlüsselwort-Kandidaten häufig anhand deren Beziehung zu anderen Schlüsselwort-Kandidaten definiert. Dabei ist ein Kandidat genau dann interessant, wenn er im Zusammenhang zu einer großen Anzahl an Kandidaten steht, die bereits als wichtig eingestuft wurden. Die Idee der graphbasierten Verfahren liegt darin, dass auf Grundlage des vorliegenden Textes ein Graph erstellt wird, anhand dessen mittels der Anwendung von verschiedenen Methoden eine Rangordnung der Schlüsselwort-Kandidaten erstellt werden kann. Jeder Kandidat stellt dabei, wie in Abbildung 6 dargestellt, einen Knoten dar. Diese sind anhand von Beziehungsmerkmalen, repräsentiert durch Kanten und deren Gewichtung, miteinander verbunden sind.

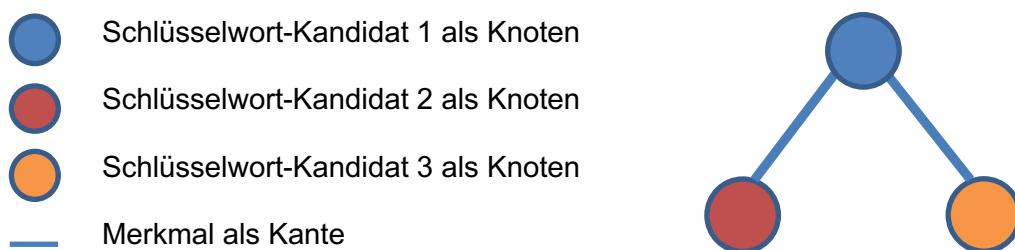


Abbildung 6 Graphbasierende Verfahren. (Quelle: Eigene Darstellung)

Grundlegend können Graphen aus gerichteten oder ungerichteten Kanten bestehen. Gerichtete Kanten werden als Pfeile dargestellt und drücken aus, dass beispielsweise Knoten A auf Knoten B verweist. Anhand der Arbeit von Mihalcea und Tarau (2004) wurde jedoch gezeigt, dass die Nutzung von gerichteten gegenüber ungerichteten Graphen keinen entschiedenen Einfluss auf das Ergebnis der Schlüsselwort-Extraktion hat. Daher wird bei den graphenbasierten Methoden im Regelfall von ungerichteten Graphen ausgegangen.

Kantenmerkmale ergeben sich beispielsweise durch die Kookkurrenz (Mihalcea und Tarau, TextRank: Bringing Order into Texts 2004, Matsuo und Ishizuka 2004) oder der semantischen Beziehung (Grineva, Grinev und Lizorkin 2009) zwischen Wörtern und drücken die Gewichtung der Kanten aus. Der Rang eines Kandidaten wird im Anschluss mittels einer rekursiven Berechnung, anhand der Gewichtung seiner anliegenden Kanten sowie der Ränge der Nachbarschaftsknoten, bestimmt (Hasan und Ng 2014, S. 1265).

Maschinelles Lernen

Da die Methoden des maschinellen Lernens immer auf der Grundlage von bereits manuell mit Schlüsselworten versehenen Trainingsdokumenten basieren, werden sie den beaufsichtigten Ansätzen untergeordnet (Turney 1999, S. 2). Mithilfe von maschinellem Lernen wird anhand dieser Daten ein Trainingsmodell aufgebaut. Bekannte Methoden hierfür sind Support Vector Machine (SVM) (Zhang, et al. 2006), Naïve Byers (Frank, et al. 1999) und Bagging (Hulth 2003). Das entstandene Modell wird im Anschluss genutzt um Schlüsselwörter aus neuen Dokumenten zu extrahieren (Zhang, et al. 2008, S. 1170).

5.3 Prozessbeschreibung

In diesem Abschnitt werden die Prozessschritte sowie die Evaluationsmethodik für die automatische Schlüsselwort-Extraktion beschrieben. Der Prozess kann gemäß Taeho (2018, S. 320) sowie Hasan und Ng (2014, S. 1263) in zwei Schritte unterteilt werden. Der erste Schritt dient der Ermittlung aller potentiellen Wörter und Phrasen, welche als Schlüsselwort-Kandidaten infrage kommen könnten. Im zweiten Schritt erfolgt die Extraktion der tatsächlichen Schlüsselwörter aus den Kandidaten unter Verwendung von Schlüsselwort-Extraktionsmethoden.

5.3.1 Ermittlung der Schlüsselwort-Kandidaten

Wie zuvor erwähnt, erfolgt im ersten Schritt der automatisierten Schlüsselwort-Extraktion typischerweise eine Tokenisierung (Vergleich Unterabschnitt 3.1 Tokenisierung) aller Worte und Phrasen, die als potentielle Schlüsselwort-Kandidaten infrage kommen. Dabei kann je nach Bedarf, auf verschiedene Heuristiken zurückgegriffen werden, um diese Kandidaten so gut wie möglich einzuschränken. Die folgenden heuristischen Verfahren werden hierbei häufig angewandt (Hasan und Ng 2014, S. 1263):

- Entfernung von Stoppwörtern (Vergleich Unterabschnitt 3.2 Stoppwörter) (Liu, et al. 2009b)
- Filterung von Wortarten mithilfe von Part-of-Speech (Vergleich Unterabschnitt 3.3 Wortartenbestimmung) (Mihalcea und Tarau, TextRank: Bringing Order into Texts 2004, Liu, Li, et al. 2009b, Florescu und Caragea 2017)
- Extraktion von N-Grammen (Vergleich Unterabschnitt 3.4 N-Gramm) (Witten, et al. 1999, Hulth 2003)

In Abhängigkeit von beispielsweise der Länge des Textes oder der zu untersuchenden textuellen Thematik eignen sich die verschiedenen Verfahren allein oder in Kombination, um eine Liste an geeigneten Schlüsselwort-Kandidaten zu erhalten. Darüber hinaus existieren weitere Verfahren (Huang, et al. 2006, Newman, et al. 2012), welche für verschiedene Textquellen-Arten konzipiert wurden.

5.3.2 Extraktion der Schlüsselwörter

Wie bereits erwähnt, wird im zweiten Schritt der automatisierten Schlüsselwort-Extraktion mithilfe verschiedener Methoden bestimmt, bei welchen zuvor extrahierten Kandidaten es sich tatsächlich um Schlüsselwörter handelt. Hierbei werden, wie in Abschnitt 5.2 beschrieben, Verfahren angewandt, die auf statistischen, sprachlichen, graphenbasierenden oder maschinell lernenden Methoden aufbauen (Hasan und Ng 2014, S. 1263 ff). Das Ergebnis

dieser Verfahren ist eine Liste der wichtigsten ermittelten Schlüsselwörter (Taeho 2018, S. 320).

5.3.3 Evaluation

Um Extraktionsverfahren überprüfen und vergleichen zu können ist es notwendig einheitliche Tests durchzuführen. Zu diesem Zweck wurden verschiedene Datensets entwickelt, welche sowohl Testdokumente als auch deren bereits richtig bestimmte Schlüsselwörter beinhalten. Diese Sets lassen sich thematisch einer Quelle und statistisch in die Anzahl an Dokumenten, die durchschnittliche Anzahl an Tokens pro Dokument und die Anzahl an vorbestimmten Schlüsselwörtern pro Dokument kategorisieren (Hasan und Ng 2014, S. 1263). Um einen Überblick über die verschiedenen Sets zu erhalten, werden diese in Tabelle 4 aufgelistet.

Quelle	Datenset/Ersteller	Statistik		
		Dokumente	Tokens/ Dokument	Schlüssel- worte/ Dokument
Paper Abstracts	<i>Inspect (Hulth 2003)</i>	2000	< 200	10
Wissenschaftliche Paper	<i>NUS corpus (Nguyen und Kan 2007)</i>	211	≈ 8000	11
	<i>citeulike.org (Medelyan, Frank und Witten 2009)</i>	180	-	5
	<i>SemEval-2010 (Kim, et al. 2010)</i>	284	> 5000	15
	<i>Schutz2008 / -</i>	-	≈ 2500	-
	<i>KDD (Gollapalli und Caragea 2014)</i>	834	-	4
	<i>WWW (Gollapalli und Caragea 2014)</i>	1350	-	5
Technische Reports	<i>NZDL (Witten, et al. 1999)</i>	1800	-	-
News Artikel	<i>DUC-2001 (Wan und Xiao 2008)</i>	308	≈ 900	8
	<i>500N-KPCrowd / -</i>	-	≈ 300	-
	<i>Reuters corpus (Hulth und Megyesi 2006)</i>	12848	-	6
	<i>Yih et al. (2006)</i>	828	-	-
Webseiten	<i>Hammouda et al. (2005)</i>	312	≈ 500	-
	<i>Blogs (Grineva, Grinev und Lizorkin 2009)</i>	252	≈ 1000	8
	<i>ICSI (Liu, et al. 2009a)</i>	161	≈ 1600	4
Emails	<i>Enron corpus (Dredze, et al. 2008)</i>	14659	-	-
Live Chats	<i>Libraby of Congress (Kim und Baldwin 2012)</i>	15	-	10

Tabelle 4 Evaluation Datensets. (Quelle: Eigene Tabelle in Anlehnung an Hasan und Ng (2014, S. 1263))

Um ein Schlüsselwort-Extraktionsverfahren zu bewerten, wird gemäß Hasan und Ng (2014, S. 1267 ff) in zwei Schritten vorgegangen.

Schritt 1: Untersuchung der Übereinstimmung

In diesem Schritt wird anhand der vordefinierten Schlüsselwörter eines Datensets eine Übereinstimmung mit den gefundenen Schlüsselwörtern ermittelt. Dabei muss vorab definiert werden, ab wann eine Übereinstimmung anerkannt wird. Eine exakte Übereinstimmung würde ausschließen, dass ein Schlüsselwort als richtig angesehen wird, wenn es sich um eine Variante des vordefinierten Schlüsselwortes handelt. Beispielsweise würde die Schlüsselwortphrase „neuronales Netzwerk“ als falsch angesehen werden, wenn sie mit der gesuchten Phrase „künstliches neuronales Netzwerk“ verglichen wird. Ein weiteres Problem der Vergleichbarkeit kann durch morphologische Variationen von Schlüsselwörtern entstehen, wobei El-Beltagy und Ahmed (2010, S. 191) dies mithilfe von Verfahren der Stammwortreduktion lösen konnten.

Schritt 2: Bewertung

Das daraus resultierende Ergebnis wird mittels der Metriken Precision (P), Recall (R) und F-Score (F) bewertet. Gemäß Turney (1997, S. 7) erfolgt die Berechnung der Precision (P) und Recall (R) Metriken anhand einer Konfusionsmatrix, dargestellt in Abbildung 7.

	Gefunden (G)	Nicht gefunden (NG)
Relevant (R)	Richtig Positiv (RP)	Falsch Negativ (FN)
Nicht relevant (NR)	Falsch Positiv (FP)	Richtig Negativ (RN)

Abbildung 7 Konfusionsmatrix. (Quelle: Eigene Darstellung in Anlehnung an Turney (1997, S. 7))

Die Begriffe der Matrix werden, in Bezug auf die Evaluation gefundener Schlüsselwörter, wie folgt definiert:

- R: Es handelt sich um ein relevantes Schlüsselwort
- NR: Es handelt sich nicht um ein relevantes Schlüsselwort
- G: Die Methode hat ein Schlüsselwort gefunden
- NG: Die Methode hat kein Schlüsselwort gefunden

- RP: Die Methode hat das relevante Schlüsselwort gefunden
- FP: Die Methode hat ein Schlüsselwort gefunden das irrelevant ist
- FN: Die Methode hat ein relevantes Schlüsselwort nicht gefunden
- RN: Die Methode hat ein irrelevantes Schlüsselwort nicht gefunden

Auf Grundlage dieser Matrix können alle gefundenen Schlüsselwörter mit den Vordefinierten verglichen und in eines der 4 Felder RP, FP, FN oder RN kategorisiert werden. So wird die Berechnung der oben genannten Metriken wie folgt möglich:

- **Precision (P)**

Der Wert drückt aus, wie viel Prozent der durch die Methode gefundenen Schlüsselwörter tatsächlich relevant sind. Die Berechnung erfolgt mittels der Formel:

$$P = \frac{RP}{RP + FP} \quad (8)$$

- **Recall (R)**

Der Wert drückt aus, wie viel Prozent aller relevanten Schlüsselwörter durch die Methode gefunden wurden. Die Berechnung erfolgt mittels der Formel:

$$R = \frac{RP}{RP + FN} \quad (9)$$

- **F-Score (F)**

Der F-Score ist ein harmonischer Mittelwert von Recall (R) und Precision (P). Die Berechnung erfolgt durch die Formel:

$$F = \frac{2 \cdot P \cdot R}{P + R} \quad (10)$$

5.4 Schlüsselwort-Extraktionsverfahren

Es existiert bereits eine Vielzahl an Schlüsselwort-Extraktionsverfahren, die anhand von unterschiedlichen Textarten bewertet wurden. Diese Textquellen reichen von langen wissenschaftlichen Veröffentlichungen bis zu Kurzfassungen und E-Mails. Dabei haben sich vier textbezogene Faktoren herauskristallisiert, welche einen Einfluss auf die Schwierigkeit der Schlüsselwort-Extraktion haben. Bei diesen Faktoren handelt es sich um die Textlänge, die strukturelle Konsistenz, die Themenwechsel und die Themenkorrelation. Da sich diese Faktoren in den verschiedenen Textarten unterscheiden, stellt sich bei den existierenden Verfahren oft eine Stärke für den einen oder anderen Bereich heraus (Hasan und Ng 2014, S. 1262-1264).

In Tabelle 5 wird eine Auswahl bisheriger Verfahren anhand ihrer Publikationen aufgelistet und nach Methoden kategorisiert.

Publikation	Verfahren			
	Statistisch	Sprachlich	Graph-basierend	Maschinelles Lernen
Dennis (1967)		✓		
Salton und Buckley (1991)		✓		
Cohen (1995)	✓			
Salton et al. (1997)		✓		
Chien (1997)	✓			
Ohsawa, Benson und Yachida (1998)	✓			
Hovy und Lin (1998)		✓		
Fukumoto et al. (1998)	✓			
Mani und Maybury (1999)		✓		
Witten et al. (1999)	✓	✓		✓
Frank et al. (1999)				✓
Barzilay et al. (1999)		✓		
Turney (1999)	✓			✓
Conroy und O'leary (2001)				✓
Humphreys (2002)		✓		
Ramos (2003)	✓			
Hulth (2003)		✓		✓
Matsuo und Ishizuka (2004)	✓			
Erkan und Radev (2004)		✓	✓	
Van der Plas et al. (2004)		✓		
Mihalcea (2004)			✓	
HaCohen-Kerner et al. (2005)				✓
Zhang, et al. (2006)				✓
Ercan und Cicekli (2007)		✓		✓
Wan und Xiao (2008)	✓		✓	
Litvak und Last (2008)			✓	✓
Zhang et al. (2008)				✓
El-Beltagy und Ahmed (2010)	✓			
Nguyen und Luong (2010)				✓
Bougouin, Boudin und Daille (2013)			✓	
Sterckx et al. (2015)			✓	
Thomas, Bharti und Babu (2016)	✓	✓		
Florescu und Caragea (2017)	✓	✓	✓	
Boudin (2018)			✓	
Campos et al. (2018a)	✓			

Tabelle 5 Schlüsselwort-Extraktionsverfahren. (Quelle: Eigene Tabelle in Anlehnung an Bharti, Babu und Jena (2017, S. 3))

Um ein Verständnis für die Anwendung der vorab definierten Prozessschritte zu erhalten werden in diesem Abschnitt drei Schlüsselwort-Extraktionsverfahren genauer beschrieben, die auch im praktischen Teil dieser Bachelorarbeit angewandt werden. Im Anschluss wird eine Übersicht über die Evaluationsergebnisse aktueller Verfahren vermittelt.

5.4.1 YAKE!

Gemäß Campos et al. (2018a, S. 684) handelt es sich bei YAKE! um ein unbeaufsichtigtes und auf statistischen Textmerkmalen aufbauendes Schlüsselwort-Extraktionsverfahren welches in der Lage ist, sowohl aus Einzel- als auch aus mehreren Wörtern bestehende Begriffe als Schlüsselworte aus einem Textdokument zu identifizieren.

In diesem Unterabschnitt wird anhand der wissenschaftlichen Publikation von Campos et al. (2018a, S. 684-691) die Funktionsweise von YAKE! erklärt. Hierzu werden die einzelnen Schritte (1) Textvorverarbeitung, (2) Merkmalsextraktion, (3) Gewichtung und (4) Schlüsselwortauswahl einzeln betrachtet. Im letzten Punkt wird auf die durch den Herausgeber evaluierten Ergebnisse eingegangen.

Schritt 1: Textvorverarbeitung

Am Anfang werden alle sich in dem vorliegenden Dokument befindlichen Worte tokenisiert. Getrennt dazu werden in einer gesonderten Liste 1, 2 und 3- Gramme gespeichert, wobei jene welche mit einem Stoppwort beginnen oder enden nicht berücksichtigt werden. Diese Liste wird jedoch erst ab Schritt 4 wieder benötigt. Schritt 2 und 3 werden anhand der reinen tokenisierten Wörter durchgeführt.

Schritt 2: Merkmalsextraktion

Um jedes der vorab tokenisierten Worte besser bewerten zu können, werden für diese jeweils fünf Merkmale ermittelt. Einige dieser Merkmale sind dabei besonders auf die westliche Sprache abgestimmt, wobei dies nicht von vornherein ausschließt, dass diese auch für andere Sprachen bestimmt werden können. Dies gilt zum Beispiel für die Großschreibung, die in westlichen Sprachen eine Wichtigkeit für Worte signalisiert.

- **Großschreibung (W_{Case})**

Da man davon ausgehen kann das Worte die mit einem Großbuchstaben beginnen tendenziell eine höhere Bedeutung haben, werden diese, mit Ausnahme von Wörtern die am Anfang eines Satzes stehen, bei der Bestimmung dieses Merkmals genauer betrachtet. Zur Berechnung von W_{Case} wird dabei die folgende Formel angewandt:

$$W_{Case} = \frac{\max(TF(U(w)), TF(A(w)))}{\log_2(TF(w))} \quad (11)$$

wobei $TF(U(w))$ die Anzahl, wie oft das Wort w mit einem Großbuchstaben beginnt, $TF(A(w))$ die Anzahl wie oft das Wort w als Akronym (Wörter, die nur aus Großbuchstaben bestehen) erkannt wurde und $TF(w)$ die gesamte Häufigkeit des Vorkommens des Wortes w repräsentiert. Dabei errechnet sich W_{Case} also aus dem Maximalwert von $TF(U(w))$ oder $TF(A(w))$ geteilt durch den Logarithmus von $TF(w)$.

- **Wort Position ($W_{Position}$)**

Unter der Annahme das Schlüsselworte in Sätzen, die am Anfang von Texten stehen eine höhere Bedeutung haben, wird bei diesem Merkmal der Wert

$W_{Position}$ errechnet, welcher umso geringer ist, je öfter ein Wort am Anfang eines Textes erscheint. Zur Berechnung von $W_{Position}$ wird die folgende Formel angewandt:

$$W_{Position} = (\log_2(\log_2(2 + Median(Sen_w)))) \quad (12)$$

wobei Sen_w die Positionen aus der Anzahl von Sätzen, in welchen das Wort w vorkommt, als geordnete Liste repräsentiert, aus welcher mit der Median Funktion der mittlere Wert errechnet wird. Dieser wird mit der Zahl 2 addiert, damit garantiert werden kann, dass das Endergebnis $W_{Position}$ größer als 0 ist. Eine Logarithmusberechnung gefolgt von einer erneuten Logarithmusberechnung ergibt das Ergebnis $W_{Position}$.

- **Worthäufigkeit (W_{Freq})**

Da angenommen wird, dass wiederholt in einem Text auftretende Worte eine höhere Bedeutung haben, wird die Häufigkeit des Vorkommens eines Wortes mit Wert W_{Freq} ausgedrückt.

Zur Berechnung von W_{Freq} wird dabei die folgende Formel angewandt:

$$W_{Freq} = \frac{TF(w)}{MeanTF + 1 \cdot \sigma} \quad (13)$$

wobei $TF(w)$ die Anzahl des Vorkommens des Wortes w, $MeanTF$ den Durchschnitt aller Worthäufigkeiten und σ deren Standardabweichung repräsentiert. W_{Freq} ergibt sich also aus der Dividende von $TF(w)$ und $MeanTF + 1 \cdot \sigma$, womit Werteverfälschungen durch eine hohe Worthäufigkeit in langen Texten vermieden werden. Diese Herangehensweise verfolgt das Ziel jene Wörter zu bewerten, welche über dem Mittelwert aller liegen.

- **Wortverwandtschaft zum Kontext (W_{Rel})**

Auf der Grundlage der Berechnung des W_{Rel} Merkmals wird ermittelt, in wieweit ein Wort den Eigenschaften eines Stopworts gleichkommt und somit eine geringe Bedeutung hat. Hierzu werden Wörter, die sich in einer vordefinierten Anzahl n links bzw. rechts neben dem Kandidatenwort befinden, genauer betrachtet. Es wird angenommen, dass sich die Wahrscheinlichkeit erhöht, dass das Kandidatenwort einem Stopwort gleicht, umso mehr sich die auf beiden Seiten umliegenden Begriffe über den kompletten Text unterscheiden. Zur Berechnung von W_{Rel} wird dabei die folgende Formel angewandt:

$$W_{Rel} = \left(0.5 + \left(\left(WL \cdot \frac{TF(w)}{MaxTF} \right) + PL \right) \right) + \left(0.5 + \left(\left(WR \cdot \frac{TF(w)}{MaxTF} \right) + PR \right) \right) \quad (14)$$

wobei $TF(w)$ die Anzahl des Vorkommens des Wortes w, $MaxTF$ die maximale Worthäufigkeit aller Worte, PL / PR jeweils das Verhältnis zwischen der Anzahl unterschiedlicher Worte auf der linken / rechten Seite und der maximalen Worthäufigkeit aller Worte ($MaxTF$) und WL / WR jeweils das Verhältnis zwischen der Anzahl unterschiedlicher Worte und der Gesamtanzahl an Wörtern auf der linken / rechten Seite repräsentiert. Anhand dieser Formel errechnet sich ein Wert, welcher mit der Bedeutungslosigkeit des Wortes ansteigt.

- **Wort in verschiedenen Sätzen (WDifSentence)**

Dieses Merkmal beschreibt, in wie vielen verschiedenen Sätzen sich ein Wort befindet. Zur Berechnung von $W_{DifSentence}$ wird dabei die folgende Formel angewandt:

$$W_{DifSentence} = \frac{SF(w)}{\#Sentences} \quad (15)$$

wobei $SF(w)$ die Anzahl der Sätze, in denen das Wort w vorkommt und $\#Sentences$ die Gesamtanzahl der Sätze repräsentiert.

Schritt 3: Gewichtung

Um eine individuelle Gewichtung für die einzelnen Wörter zu erhalten und diese damit vergleichbar zu machen, werden die aus Schritt 2 ermittelten Merkmale zu dem Wert $S(w)$ zusammengefasst. Zur Berechnung von $S(w)$ wird die folgende Formel angewandt:

$$S(w) = \frac{W_{Rel} \cdot W_{Position}}{W_{Case} + \frac{W_{Freq}}{W_{Rel}} + \frac{W_{DifSenence}}{W_{Rel}}} \quad (16)$$

Grundlegend steigt die Bedeutung eines Wortes mit der Höhe des $S(w)$ -Wertes. Interessant hierbei ist die Herangehensweise mittels der Division von W_{Freq} und $W_{DifSenence}$ durch W_{Rel} . Das hohe Aufkommen eines Wortes im Allgemeinen oder in unterschiedlichen Sätzen kann anhand eines hohen W_{Rel} -Wertes, welcher mit der Bedeutungslosigkeit steigt, gut relativiert werden. So können aufgrund eines wahrscheinlich geringen daraus resultierenden $S(w)$ -Wertes belanglose Wörter gut erkannt werden.

Schritt 4: Schlüsselwortauswahl

Im letzten Schritt wird ein abschließender Wert für jeden Schlüsselwortkandidaten berechnet. Dabei werden vorab keine Kandidaten, aufgrund eines schlechten $S(w)$ Gewichtungswertes, ausgeschlossen. Das bedeutet, dass Schlüsselwörter unabhängig von ihrem Auftreten als wichtig oder unwichtig eingestuft werden können. Um die Schlüsselwortauswahl abzuschließen, wird zur Berechnung des $S(kw)$ -Wertes folgende Formel für jeden Kandidaten angewandt:

$$S(kw) = \frac{\prod_{w \in kw} S(w)}{TF(kw) + (1 + \sum_{w \in kw} S(w))} \quad (17)$$

wobei $\prod_{w \in kw} S(w)$ im Zähler, die in Schritt 3 berechneten Gewichtungswerte $S(w)$ für die aus bis zu drei Wörtern bestehenden Schlüsselwortkandidaten zusammen multipliziert. Dieser Wert wird dann durch die im Nenner gebildete Summe geteilt. Diese setzt sich aus der im Text vorkommenden Häufigkeit $TF(kw)$ sowie der Aufsummierung $(1 + \sum_{w \in kw} S(w))$ der Gewichtungswerte $S(w)$ des aus bis zu drei Wörtern bestehenden Schlüsselwortkandidaten kw , zusammen. Auf diese Weise wird verhindert, dass 2 und 3-Gramm Kandidaten nicht mittels im Zähler durchgeföhrter Multiplikation bevorteilt werden.

Zum Schluss werden ähnliche Kandidaten durch Berechnung der Levenshtein distance (1966) herausgefiltert. Dabei wird, sobald der gemessene Wert zwischen zwei Schlüsselworten eine bestimmte Grenze überschreitet, das Schlüsselwort mit dem höchsten $S(kw)$ -Wert eliminiert. Das Ergebnis ist eine Liste von 1, 2 oder 3-Gramm Schlüsselwörtern, wobei jene mit einem geringeren $S(kw)$ -Wert als wichtiger eingestuft werden.

Evaluation

Die Evaluation von YAKE! erfolgt auf Grundlage der exakten Übereinstimmung der Top 10 gefundenen Schlüsselwörtern, mit den im jeweiligen Datensatz vorbestimmten Schlüsselwörtern. Dabei werden, wie in Unterabschnitt 5.3.3 beschrieben, die Werte Precision (P), Recall (R) und F-Score (F) berechnet. Wie in Tabelle 6 sichtbar haben Campos et al. (2018a) einen Vergleich zwischen ihrer und vier weiteren unbeaufsichtigten Methoden (Mihalcea und Tarau 2004, Wan und Xiao 2008, Rose, et al. 2010) anhand von drei unterschiedlichen Datensets (Vergleich Tabelle 4) durchgeführt.

Metrik	SemEval2010			Schutz2008			500N-KPCrowd		
	P %	R %	F %	P %	R %	F %	P %	R %	F %
YAKE!	15,3	10,3	12,3	21,7	5,8	9,1	25,1	6,3	10,1
TextRank	10,1	6,7	8,1	19,8	5,2	8,2	26,5	6,3	10,3
TF.IDF	3,6	2,3	2,8	10	2,8	4,3	22,3	6,0	9,5
SingleRank	3,5	2,2	2,7	8,2	2,4	3,7	19,0	5,4	8,4
RAKE	0,7	0,4	0,5	1,3	0,4	0,6	12,0	3,8	5,8

Tabelle 6 Evaluation YAKE!. (Quelle: Eigene Tabelle in Anlehnung an Campos et al. (2018a))

Die in Tabelle 6 grün markierten Felder repräsentieren die höchsten und damit besten Werte für die jeweiligen Metriken und Datensets. Die Ergebnisse spiegeln wider, dass YAKE! sehr gute Werte erzielen konnte. TextRank erzielte im 500N-KPCrowd Datensatz dennoch etwas bessere Resultate, wobei der Unterschied minimal ist. Des Weiteren lässt sich erkennen, dass die Score-Werte stark von der durchschnittlichen Dokumentenlänge abhängen. Alle Methoden konnten in kürzeren Texten bessere Ergebnisse erreichen.

5.4.2 KEA

Gemäß Witten et al. (1999, S. 254) handelt es sich bei KEA um ein überwachtes und auf linguistischen, statistischen und maschinell lernenden Algorithmen aufbauendes Schlüsselwort-Extraktionsverfahren. Dabei wird nach dem Ansatz von Turney (1999) verfahren, bei welchem ein Modell aus positiven und negativen Schlüsselwortbeispielen erstellt wird. Anhand dieses Modells können dann Schlüsselworte aus neuen Dokumenten erkannt werden (L. Liu 2008). KEA hat sich in den vergangenen Jahren zu einer der wichtigsten überwachten Anwendungen in diesem Bereich entwickelt, welche zum Vergleich gegen neuere Verfahren herangezogen wird (Beliga 2014, Bharti, Babu und Jena 2017). In diesem Unterabschnitt wird anhand der wissenschaftlichen Publikation von Witten et al. (1999, S. 254–255) die Funktionsweise von KEA erklärt. Grundlegend kann das Verfahren in die folgenden zwei Hauptschritte unterteilt werden.

1. Training: Erstellung eines Modells anhand von Dokumenten deren Schlüsselworte bereits bekannt sind.
2. Extraktion: Extraktion von Schlüsselwortphrasen anhand des zuvor erstellten Modells.

Dabei geht beiden Schritten die Ermittlung der Schlüsselwort-Kandidaten und Merkmalextraktion voraus. Daher gliedert sich dieser Unterabschnitt in die folgenden vier Schritte.

Schritt 1: Ermittlung der Schlüsselwort-Kandidaten

Am Anfang müssen die Schlüsselwort-Kandidaten aus dem Text herausgefiltert werden. Um dies zu erreichen, wird in den folgenden drei Schritten vorgegangen.

- **Textreinigung**

In diesem Schritt wird der Text tokenisiert. Hierbei werden alle Nummern, Klammern und Satzzeichen durch Wortabtrennung ersetzt und mittels eines Bindestriches verbundene Wörter getrennt. Außerdem erfolgt eine Löschung aller Apostrophe und Zeichenfolgen, welche keine Buchstaben beinhalten.

- **Erkennung von Schlüsselwort-Kandidaten**

Die Erkennung von Schlüsselwort-Kandidaten erfolgt bei KEA nach drei Regeln. Diese besagen, dass die Kandidaten:

- auf drei Wörter in der Länge begrenzt sind
- niemals Wörter sein können, die immer mit einem Großbuchstaben beginnen
- niemals mit einem Stoppwort anfangen oder enden

- **Stammformreduktion und Groß- und Kleinschreibung**

Im letzten Schritt wird die Liste an Schlüsselwort-Kandidaten dupliziert. An einer Liste erfolgt eine iterative Stammformreduktion nach Lovins (1968). Diese wird an den Schlüsselwort-Kandidaten so oft durchgeführt, bis keine Veränderungen mehr auftreten. Um den User nach der Evaluierung die originalen und nicht auf ihre Stammform reduzierten Schlüsselwörter präsentieren zu können, bleiben diese in der zweiten Liste in ihrer Ursprungsform erhalten. Falls bestimmte Wörter unterschiedliche Groß- und Kleinschreibungsmerkmale aufweisen wird die Variante präsentiert, die am häufigsten in der Liste auftritt.

Schritt 2: Merkmalsextraktion

Nachdem die Schlüsselwort-Kandidaten ermittelt wurden, werden für diese jeweils 2 Merkmale ermittelt. Die Berechnung dieser Merkmale wird in diesem Abschnitt beschrieben.

- **TFxIDF**

Wie bereits von Salton und Buckley (1988) beschrieben, handelt es sich bei TFxIDF um ein effektives statistisches Verfahren, bei welchem die Wichtigkeit von Schlüsselwörtern anhand ihres Vorkommens berechnet werden kann. Die Berechnung erfolgt mittels der Formel:

$$TFxIDF = \frac{freq(P, D)}{size(D)} \cdot -\log_2 \frac{df(P)}{N} \quad (18)$$

wobei sich $TFxIDF$ aus der Multiplikation der normalisierten Häufigkeit TF und der inversen Dokumentenhäufigkeit IDF einer Schlüsselwortphrase ergibt. Dabei wird die normalisierte Schlüsselworthäufigkeit:

$$TF = \frac{freq(P, D)}{size(D)} \quad (19)$$

mittels der Division aus der Anzahl $freq()$ wie oft eine Schlüsselwortphrase P in einem Dokument D vorkommt und der Gesamtanzahl $size()$ an Wörtern in einem Dokument D kalkuliert. Die inverse Dokumentenhäufigkeit einer Phrase:

$$IDF = -\log_2 \frac{df(P)}{N} \quad (20)$$

berechnet sich aus dem negierten Logarithmus des Quotienten der Gesamtzahl $df()$ der zu untersuchenden Dokumente, welche die Schlüsselwortphrase P enthalten und der Gesamtanzahl N aller Dokumente. Das Ergebnis von IDF repräsentiert also die Wichtigkeit einer Phrase, anhand der, durch die Multiplikation mit TF, häufig auftretende Ausdrücke wie "ist", "von" und "das" schlechter bewertet werden (Salton und Buckley 1988, S. 516 - 517).

- **Distanz zum ersten Vorkommen einer Schlüsselwortphrase**

Um zu erkennen, an welcher Stelle sich Schlüsselworte befinden, wird der Wert des ersten Vorkommens einer Phrase FO bestimmt. Die Berechnung erfolgt anhand der Formel:

$$FO = \frac{precede(P)}{size(D)} \quad (21)$$

wobei $precede(P)$ die Anzahl der Wörter, die der ersten Schlüsselwortphrase P vorangehen und $size(D)$ die gesamte Anzahl an Wörtern in einem Dokument D repräsentiert. Aufgrund der Division von $precede(P)$ und $size(D)$ wird ein Wert zwischen 0 und 1 berechnet, welcher die Distanz vom Anfang des Dokumentes bis zum ersten Vorkommen der Schlüsselwortphrase ausdrückt.

- **Diskretisierung**

Zum Schluss werden die beiden Merkmale in Zahlenbereiche diskretisiert, so dass sie als nominale Werte behandelt werden können. Dieser Vorgang wird anhand der Methode von Fayyad und Irani (1993) durchgeführt, wobei ein Set an Zahlen in n Bereiche geteilt wird. Hierfür wird vorab eine Schrittgröße pro Bereich ermittelt. Dies erfolgt anhand der Formel:

$$s = \frac{size(d)}{b} \quad (22)$$

wobei $size(d)$ die Anzahl an Nummern und b die Menge an Bereichen repräsentiert. Das aufsteigend sortierte Set an Zahlen wird im Anschluss in seine Bereiche aufgeteilt, wobei jeder Bereich die Spanne der in der Höhe der Schrittgröße aufeinander folgenden Zahlen beinhaltet.

Schritt 3: Training Aufbau eines Modells

In diesem Schritt wird anhand von Trainingsdokumenten, für welche bereits die Schlüsselworte bekannt sind, das Trainingsmodell aufgebaut. Das Trainingsmodell dient dem Zweck, im Nachhinein Schlüsselwörter automatisch zu identifizieren. Für den Aufbau werden für alle Trainingsdokumente die Schlüsselwortkandidaten mit ihren Merkmalen wie vorab beschrieben identifiziert. Alle Phrasen, die nur einmal in dem Dokument vorkommen werden gelöscht um das Trainingsset schlank zu halten. Je nachdem, ob es sich bei den übrig gebliebenen Schlüsselwortkandidaten um eines der tatsächlichen Schlüsselworte aus den Trainingsdokumenten handelt, werden diese mit dem booleschen Wert true oder false markiert. Das aus diesem Vorgang entstandene binäre Merkmal wird als Klassenmerkmal bezeichnet. Kea nutzt im Anschluss die maschinelle Lerntechnik Naïve Bayes (Vergleich Unterabschnitt 4.3.4), um ein Modell zu erstellen, welches aufgrund der berechneten diskretisierten Merkmalswerte die Klasse voraussagt. Dafür lernt das Modell anhand welcher Merkmalskombinationen es sich bei einem Schlüsselwortkandidaten wahrscheinlich um ein Schlüsselwort handelt.

Schritt 4: Extraktion neuer Schlüsselwortphrasen

Für die Extraktion von Schlüsselwortphrasen aus einem neuen Textdokument werden wie vorab beschrieben alle Schlüsselwortkandidaten sowie deren Merkmale bestimmt. Anhand des Trainingsmodells wird daraufhin die Gesamtwahrscheinlichkeit jedes Kandidaten vorausgesagt, woraus sich die potenziell besten Schlüsselwortphrasen erkennen lassen.

Bei der Anwendung des Naïve Bayes Algorithmus an einem Schlüsselwort werden die zwei Wahrscheinlichkeitswerte $P[ja]$ und $P[nein]$ anhand der Formeln:

$$P[ja] = \frac{Y}{Y + N} \cdot P_{TFxIDF}[t | ja] \cdot P_{distance}[d | ja] \quad (23)$$

und

$$P[nein] = \frac{N}{Y + N} \cdot P_{TFxIDF}[t | nein] \cdot P_{distance}[d | nein] \quad (24)$$

berechnet. Dabei beschreibt Y die Anzahl der als *true* markierten und N die Anzahl der als *false* markierten Schlüsselwörter aus den Trainingsdokumenten, in welchen die tatsächlichen Schlüsselwörter bereits vorab definiert wurden. Anhand der Division von Y (bzw. N) durch die Gesamtanzahl der Schlüsselwortkandidaten $Y + N$ berechnet sich die prozentuale Wahrscheinlichkeit für das Auftreten oder Nichtauftreten von Schlüsselwortphrasen in einem Dokument. Des Weiteren repräsentiert t den Merkmalswert TFxIDF und d den Merkmalswert der Distanz zum ersten Vorkommen einer Schlüsselwortphrase. P_{TFxIDF} und $P_{distance}$ beschreiben die Wahrscheinlichkeit, dass die Schlüsselwortphrase P aufgrund ihrer Merkmale t und d ein Schlüsselwort (*ja*) oder kein Schlüsselwort (*nein*) ist.

Aus den berechneten Wahrscheinlichkeitswerten $P[ja]$ und $P[nein]$ wird zum Schluss anhand der Formel:

$$p = \frac{P[ja]}{P[ja] + P[nein]} \quad (25)$$

ein Vergleichswert ermittelt mithilfe dessen sich die potenziell besten Schlüsselwortphrasen erkennen lassen.

Beispielmodell

Um den Vorgang zu verdeutlichen beschreiben Witten et al. (1999, S. 9-11) den Prozess mittels eines praktischen Beispiels. Hierbei wird das Trainingsmodell anhand von Tabelle 7, Tabelle 8 und Tabelle 9 dargestellt.

Merkmal	Diskretisierungsspannen				
	1	2	3	4	5
TFxIDF	< 0.0031	[0.0031, 0.0045)	[0.0045, 0.013)	[0.013, 0.033)	≥ 0.033
Distanz	< 0.0014	[0.0014, 0.017)	[0.017, 0.081)		≥ 0.081

Tabelle 7 Diskretisierungstabelle. (Quelle: Eigene Tabelle in Anlehnung an Witten et al. (1999, S. 10))

In Tabelle 7 wurde das TFxIDF Merkmal in fünf und das Distanzmerkmal in vier Spannen diskretisiert.

Merkmale	Merkmalswerte	Diskretisierungsspannen				
		1	2	3	4	5
TFxIDF	$P[TFxIDF ja]$	0.2826	0.1002	0.2986	0.1984	0.1182
	$P[TFxIDF nein]$	0.8609	0.0548	0.0667	0.0140	0.0036
Distanz	$P[Distanz ja]$	0.1952	0.3360	0.2515	0.2173	
	$P[Distanz nein]$	0.0194	0.0759	0.01789	0.7333	

Tabelle 8 Klassenmerkmalsgewichtung. (Quelle: Eigene Tabelle in Anlehnung an Witten et al. (1999, S. 10))

Anhand der Werte aus Tabelle 7 lassen sich, wie in Tabelle 8 zu sehen, neun Merkmalsgewichte für positive Beispiele (ja) sowie neun Merkmalsgewichte für negative Beispiele (nein) bestimmen.

Klasse	Trainingswortphrasen	Trainingswahrscheinlichkeit
ja	493	$P(ja) = Y/(Y+N) = 0.0044$
nein	112183	$P(nein) = N/(Y+N) = 0.9956$

Tabelle 9 Trainingswahrscheinlichkeit. (Quelle: Eigene Tabelle in Anlehnung an Witten et al. (1999, S. 10))

In Tabelle 9 wird die Anzahl der positiven (ja) und negativen (nein) Trainingswortphrasen dargestellt. Die berechnete Trainingswahrscheinlichkeit drückt dabei aus wo hoch die prozentuale Chance ist, dass es sich bei einem Schlüsselwortkandidaten im Trainingsset um ein tatsächliches Schlüsselwort handelt oder nicht.

Für das Beispiel wird angenommen, dass zwei Wortphrasen aus einem Text untersucht werden, welche die folgenden berechneten TFxIDF und Distanzmerkmale aufweisen:

Phrase	TFxIDF	Distanz	Diskretisierungsspannen	
1	0.0189	0.0254	TFxIDF = 4	Distanz = 3
2	0.0021	0.0025	TFxIDF = 1	Distanz = 2

Tabelle 10 Merkmale Beispielwortphrasen. (Quelle: Eigene Tabelle)

Anhand der Merkmalswerte werden diese der jeweiligen Diskretisierungsspanne mithilfe von Tabelle 7 zugeordnet. Die benötigten Merkmalsgewichtungen aus Tabelle 8 ergeben sich durch die zuvor ermittelten Diskretisierungsspannen. Als Folge dessen lassen sich mithilfe der Trainingswahrscheinlichkeiten aus Tabelle 9, die zwei Wahrscheinlichkeitswerte $P[ja]$ und $P[nein]$ gemäß der vorab genannten Formel berechnen.

$$P1[ja] = 0.0044 \cdot 0.1984 \cdot 0.2515 = 0.00021954944$$

$$P1[nein] = 0.9956 \cdot 0.0140 \cdot 0.1789 = 0.00249357976$$

$$P2[ja] = 0.0044 \cdot 0.2826 \cdot 0.3360 = 0.00041779584$$

$$P2[nein] = 0.9956 \cdot 0.8609 \cdot 0.0759 = 0.06505480384$$

Daraus ergeben sich die folgenden Vergleichswerte:

$$p1 = \frac{0.00021954944}{0.00021954944 + 0.00249357976} = 0.080921115$$

$$p2 = \frac{0.00041779584}{0.00041779584 + 0.06505480384} = 0,006381231875$$

woraus erkennbar ist, dass es sich bei Wortphrase 1 ($p1$), aufgrund des höheren Ergebnisses wahrscheinlicher um ein Schlüsselwort handelt als Wortphrase 2 ($p2$).

Evaluation

Für die Evaluation von KEA haben Witten et al. (1999, S. 11 ff) 1800 technische Berichte über das Thema Informatik genutzt, welchen vorab Schlüsselwörter zugewiesen wurden. Anhand dieser Dokumente wurden drei Tests bestimmt um zu ermitteln, (1) wie effektiv der KEA Algorithmus im Allgemeinen ist, (2) welchen Effekt eine mengenmäßige Veränderung der Trainingsdokumente hat und (3) inwieweit sich die Performance bei der Nutzung von kurzen Texten verändert. Zur Messung ihrer Ergebnisse nutzen Witten et al. (1999, S. 11 ff), anders als bei herkömmlichen Evaluationsmethoden, die durchschnittliche Anzahl an Schlüsselwortübereinstimmungen zwischen den gefundenen und den vorab bestimmten.

1. Gesamteffektivität

Für die Ermittlung der Gesamteffektivität werden 50 Trainingsdokumente, 500 Testdokumente und 100 Korpusdokumente genutzt. Die Auswertung erfolgt anhand der Top 5, 10, 15 und 20 durch KEA gefundenen Schlüsselwörter.

Top Schlüsselwörter	Durchschnittliche Übereinstimmung mit Autoren Schüsselwörtern	Durchschnittliche Gesamtübereinstimmung
5	0,93	20,67%
10	1,39	30,89%
15	1,68	37,33%
20	1,88	41,78%

Tabelle 11 KEA Gesamteffektivität. (Quelle: Eigene Tabelle in Anlehnung an Witten et al. (1999, S. 13))

Die Ergebnisse der Evaluation werden in Tabelle 11 dargestellt. Die zweite Spalte beschreibt die durchschnittliche Übereinstimmung zwischen den mittels KEA gefundenen und den durch die Autoren benannten Schlüsselwörtern. Die Dokumente enthalten durchschnittlich 4,5 im Text vorkommende Schlüsselwörter. Spalte drei berechnet sich aus den jeweiligen Werten von Spalte zwei geteilt durch 4,5. Wie rauszusehen, steigt die Übereinstimmung mit der Anzahl der betrachteten Top Schlüsselwörter. Die größte Effektivitätssteigerung erscheint mit ca. 10% zwischen der Auswahl der Top 5 und Top 10.

2. Anzahl der Trainingsdokumente

In diesem Test wird ermittelt, welchen Einfluss die Anzahl der genutzten Trainingsdokumente auf das Endergebnis hat. Es werden dafür 100 Korpus und 500 Testdokumente genutzt.

Trainings-Dokumente	Durchschnittliche Übereinstimmung Top 5 Schlüsselwörter	Durchschnittliche Übereinstimmung Top 15 Schlüsselwörter
0	0,684	1,266
1	0,717	1,301
5	0,819	1,508
10	0,840	1,542
20	0,869	1,625
50	0,898	1,650
100	0,908	1,673

Tabelle 12 KEA Effekt Anzahl Trainingsdokumente. (Quelle: Eigene Tabelle in Anlehnung an Witten et al. (1999, S. 16))

Das in Tabelle 12 dargestellte Ergebnis zeigt, dass eine stetige Steigerung bei der Anwendung von bis zu 50 Trainingsdokumenten erkennbar ist. Daraus lässt sich schließen, dass gute Resultate bereits mit einer relativ kleinen Anzahl an Trainingsdokumenten erreicht werden können.

3. Dokumentenlänge

In diesem Test wird der Einfluss der Textlänge auf das Ergebnis evaluiert. Dafür werden 100 Korpus, 110 Trainings- und 429 Testdokumente genutzt. Die Testdokumente werden hierbei, wie in Tabelle 13 dargestellt, zum einen anhand ihres vollen Textes und zum anderen anhand ihrer Abstracts untersucht.

Dokumentenlänge	Durchschnittliche Übereinstimmung Top 5 Schlüsselwörter	Durchschnittliche Übereinstimmung Top 15 Schlüsselwörter
Voller Text	0,909	1,712
Abstracts	0,655	1,025

Tabelle 13 KEA Effekt Dokumentenlänge. (Quelle: Eigene Tabelle in Anlehnung an Witten et al. (1999, S. 16))

Erkennbar ist, dass KEA bei der Auswertung von kurzen Texten weniger korrekte Schlüsselwörter extrahiert. Dementsprechend scheint diese Methode bei längeren Texten besser zu funktionieren.

5.4.3 PositionRank

Gemäß Florescu und Caragea (2017, S. 1105) handelt es sich bei PositionRank um ein auf Graphen basierendes und unbeaufsichtigtes Schlüsselwort-Extraktionsverfahren. Dabei werden Schlüsselwort-Kandidaten anhand ihrer linguistischen Textmerkmale gefiltert und durch den auf Graphen basierenden Google PageRank (Brin und Page 1998) Algorithmus als Schlüsselwörter identifiziert. Die für die Identifikation benötigten Graphenmerkmale resultieren aus den einzelnen Textpositionen sowie gemeinsamen Relationen der Kandidaten. PositionRank konnte gegenüber vergleichbaren graphbasierenden Verfahren (Mihalcea und Tarau, TextRank: Bringing Order into Texts 2004, Wan und Xiao 2008) das Ergebnis der Extraktion von Schlüsselwörtern aus Forschungsberichten um 29% steigern (Florescu und Caragea 2017, S. 1105).

In diesem Unterabschnitt wird die Funktionsweise von PositionRank auf der Grundlage der dazugehörigen wissenschaftlichen Publikation von Florescu und Caragea (2017, S. 1105–1115) erklärt. Dazu werden die einzelnen Schritte (1) Textvorverarbeitung, (2) Graphenerstellung, (3) Rangermittlung und (4) Schlüsselwortauswahl einzeln betrachtet.

Schritt 1: Textvorverarbeitung

Im ersten Schritt wird der Text tokenisiert und mit POS Tags versehen. Um später ein übermäßiges Anwachsen an Knotenpunkten und Kanten zu vermeiden, wird keine N-Gramm Sequenzierung vorgenommen. Eventuelle Schlüsselwortphrasen werden in einem Nachbearbeitungsprozess gefunden.

Schritt 2: Graphenerstellung

Anhand der POS Tags werden in diesem Schritt alle Tokens gefiltert und als Knotenpunkte V dem ungerichteten Graphen $G = (V, E)$ hinzugefügt, wobei jene ab diesem Moment als Schlüsselwort-Kandidaten $v \in V$ angesehen werden. Der Filter wird vorab bestimmt, so dass beispielsweise nur Adjektive, Pronomen und Substantive betrachtet werden. Die sich nun auf dem Graph befindenden Knotenpunkte v werden immer dann durch Kanten $(v_i, v_j) \in E$ verbunden, wenn sich diese im ursprünglichen Text innerhalb eines vorab bestimmten Abstandsfensters von n Worten befinden. Für jede Kante $(v_i, v_j) \in E$ wird ein Gewicht bestimmt. Dieses basiert auf der Anzahl des gemeinsamen Auftretens der mittels der Knoten v_i und v_j repräsentierten Kandidaten innerhalb eines vorab bestimmten Abstandsfensters von n Worten. Dementsprechend würde beispielsweise der Satz:

„Google zielt darauf ab, den Nutzern die Suchergebnisse nach Relevanz sortiert zu liefern.“

durch eine Filterung von Substantiven und einem Abstandswert von drei Worten, den in Abbildung 8 dargestellten Graphen ergeben, wobei die Gewichtung für alle Kanten 1 beträgt.

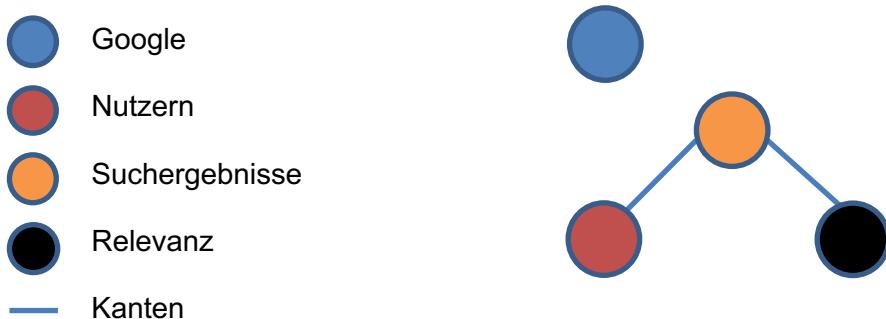


Abbildung 8 PositionRank Graphenerstellung. (Quelle: Eigene Darstellung)

Schritt 3: Rangermittlung

Im folgenden Schritt wird ein Score-Wert für jeden Knotenpunkt ermittelt, wodurch sich eine Rangfolge der Schlüsselwort Kandidaten ergibt. Bevor diese Ermittlung erfolgen kann, bedarf es der Normalisierung der Kantengewichtungen sowie der Berechnung der normalisierten Wortpositionsgewichtungen.

- **Normalisierung der Kantengewichtung**

Für die Normalisierung der Kantengewichtung wird eine Beziehungsmatrix M erstellt. Die darin enthaltenen Elemente $m_{ij} \in M$ repräsentieren die Gewichtswerte aller Kanten (v_i, v_j) . Falls zwischen den Knoten v_i und v_j keine Beziehung bestehen sollte, wird dem Element m_{ij} eine 0 zugewiesen. Daraufhin wird Matrix M in eine normalisierte Form \tilde{M} , bestehend aus den Elementen $\tilde{m}_{ij} \in \tilde{M}$ umgewandelt, wofür folgende Formel angewandt wird:

$$\widetilde{m_{ij}} = \frac{m_{ij}}{\sum_{j=1}^V m_{ij}} \quad (26)$$

wobei V einen Vektor repräsentiert, der alle Knoten v_i beinhaltet. Der Gewichtungswert m_{ij} wird also mittels der Summe aller Gewichtungswerte berechnet, welche vom Knotenpunkt v_i ausgehen.

- **Berechnung der normalisierten Wortpositionsgewichtung**

Um eine Gewichtung für die Positionen eines Kandidaten im Text zu ermitteln, wird der normalisierte Vektor \tilde{P} berechnet. Hierfür wird vorab die Gewichtung der Positionen p_i für jeden Kandidaten v_i im Text ermittelt. Wenn derselbe Kandidat mehrmals im Text erscheint, werden alle Gewichte summiert. Wenn also beispielsweise ein Wort an der vierten, zehnten und fünfzehnten Stelle in einem Text auftaucht, wird folgendes berechnet:

$$p_i = \frac{1}{4} + \frac{1}{10} + \frac{1}{15} = \frac{5}{12} = 0,4167$$

Der normalisierte Vektor \tilde{P} ergibt sich im Anschluss aus der Formel:

$$\begin{aligned} \tilde{P} = \left[\tilde{p}_i = \frac{p_i}{p_i + p_k + \dots + p_V}, \tilde{p}_k = \frac{p_k}{p_i + p_k + \dots + p_V}, \dots, \tilde{p}_V \right. \\ \left. = \frac{p_V}{p_i + p_k + \dots + p_V} \right] \end{aligned} \quad (27)$$

wobei $\tilde{p}_i \in \tilde{P}$.

- **Score Berechnung**

Basierend auf den berechneten Gewichtungswerten kann der Kandidatenscore $S(v_i)$ anhand einer leicht abgewandelten Form des PageRank (Brin und Page 1998) Algorithmus errechnet werden. Dies erfolgt gemäß der folgenden Formel:

$$S(v_i) = (1 - \alpha) \cdot \tilde{p}_i + \alpha \cdot \sum_{v_j \in Adj(v_i)} \widetilde{w_{ji}} \cdot S(v_j) \quad (28)$$

wobei \tilde{p}_i das normalisierte Wortpositionsgewicht für den Kandidaten v_i und $\widetilde{w_{ji}}$ das normalisierte Kantengewicht zwischen dem Kandidaten v_j und einem verbundenen Kandidaten v_i repräsentiert. Der Dämpfungsfaktor α , welcher zwischen 0 und 1 gesetzt werden kann hat die Aufgabe, die Wahrscheinlichkeit des Sprunges von einem Knotenpunkt zu einem anderen zufälligen Knotenpunkt zu integrieren, um nicht in einem Knotenstrang hängen zu bleiben. Dieser Faktor wird im Regelfall mit 0.85 bestimmt (Brin und Page 1998, S. 109). Um die Scores und Gewichtungen aller miteinander verbundenen Kandidaten einfließen zu lassen, wird anhand des Formelteiles $\sum_{v_j \in Adj(v_i)} \widetilde{w_{ji}} \cdot S(v_j)$ rekursiv über alle Verbindungen und deren Unterverbindungen zu dem Kandidaten v_i iteriert, um die Summe aller Produkte aus Gewichtungen und Scores zu bilden. Wie in Abbildung 9 schemenhaft dargestellt, berechnet sich v_i aus der Summe der blauen, roten und schwarzen Verbindungen. Der Endknoten des blauen Verbindungsstrangs wird dabei aus der Summe der grünen und lila Verbindungen ermittelt.

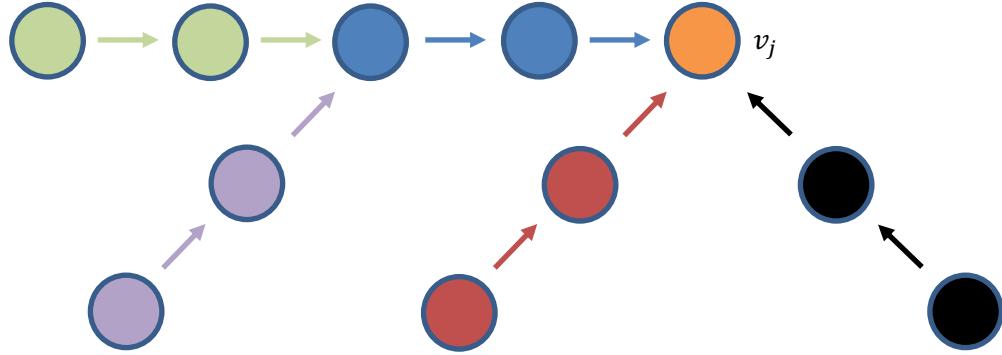


Abbildung 9 Iterative Scoreberechnung. (Quelle: Eigenen Darstellung)

Um eine sehr lange und zum Schluss wertlose Berechnung zu vermeiden, wird dieser Vorgang pro ausgehenden Strang abgebrochen, sobald die Differenz zwischen zwei aufeinanderfolgenden Iterationen weniger als 0,001 beträgt oder eine Anzahl von 100 Iterationen erreicht ist.

Schritt 4: Schlüsselwortauswahl

Alle in dem Text direkt aufeinander folgenden Kandidaten werden zu Phrasen zusammengefasst. Des Weiteren müssen diese Phrasen einem vorab bestimmten regulären Ausdruck gleichen. Beispielsweise nur Phrasen in der Form (Adjektiv)*(Substantiv) werden als Schlüsselwortphrasen anerkannt. Dementsprechend lassen sich beliebige n-Gramme former. Zum Schluss werden die Scores aller individuellen Wörter in der Phrase addiert. Die Wichtigkeit der Kandidaten steigt somit mit dem Score, woraus sich eine Liste der wichtigsten Schlüsselwörter ergibt.

Evaluation

Die in Tabelle 14 präsentierte Evaluation von PositionRank wurde gemäß Florescu und Caragea (2017, S. 1110–1113) mittels des Vergleiches von fünf unbeaufsichtigten Methoden (Mihalcea und Tarau 2004, Wan und Xiao 2008, Liu, Huang, et al. 2010) und anhand von drei unterschiedlichen Datensätzen (Vergleich Tabelle 4) durchgeführt. Alle Datensätze beinhalten wissenschaftliche Publikationen, wobei für diese Tests nur deren Abstracts und die dazugehörigen Schlüsselwörter in Betracht gezogen wurden. Die in Unterabschnitt 5.3.3 beschriebenen Metriken Precision (P), Recall (R) und F-Score (F) wurden anhand der Top 2, 4, 6 und 8 durch die Methoden gefundenen Schlüsselwörter gemessen.

Daten-set	Methoden	Top2			Top4			Top6			Top8		
		P %	R %	F %	P %	R %	F %	P %	R %	F %	P %	R %	F %
KDD	<i>PositionRank</i>	11.1	5.6	7.3	10.8	11.1	10.6	9.8	15.3	11.6	9.2	18.9	12.1
	<i>TF-IDF</i>	10.5	5.2	6.8	9.6	9.7	9.4	9.2	13.8	10.7	8.7	17.4	11.3
	<i>TextRank</i>	8.1	4.0	5.3	8.3	8.5	8.1	8.1	12.3	9.4	7.6	15.3	9.8
	<i>Single Rank</i>	9.1	4.6	6.0	9.3	9.4	9.0	8.7	13.1	10.1	8.1	16.4	10.6
	<i>Expand Rank</i>	10.3	5.5	6.9	10.4	10.7	10.1	9.2	14.5	10.9	8.4	17.5	11.0
	<i>TPR</i>	9.3	4.8	6.2	9.1	9.3	8.9	8.8	13.4	10.3	8.0	16.2	10.4
WWW	<i>PositionRank</i>	11.3	5.3	7.0	11.3	10.5	10.5	10.8	14.9	12.1	9.9	18.1	12.3
	<i>TF-IDF</i>	9.5	4.5	5.9	10.0	9.3	9.3	9.6	13.3	10.7	9.1	16.8	11.4
	<i>TextRank</i>	7.7	3.7	4.8	8.6	7.9	8.0	8.1	12.3	9.8	8.2	15.2	10.2
	<i>Single Rank</i>	9.1	4.2	5.6	9.6	8.9	8.9	9.3	13.0	10.5	8.8	16.3	11.0
	<i>Expand Rank</i>	10.4	5.3	5.7	10.4	10.6	10.1	9.5	14.7	11.2	8.6	17.7	11.2
	<i>TPR</i>	8.8	4.2	5.5	9.6	8.9	8.9	9.5	13.2	10.7	9.0	16.5	11.2
Nguyen	<i>Position Rank</i>	10.5	5.8	7.3	10.6	11.4	10.7	11.0	17.2	13.0	10.2	21.1	13.5
	<i>TF-IDF</i>	7.3	4.0	5.0	9.5	10.3	9.6	9.1	14.4	10.9	8.9	18.9	11.8
	<i>TextRank</i>	6.3	3.6	4.5	7.4	7.4	7.2	7.8	11.9	9.1	7.2	14.8	9.4
	<i>Single Rank</i>	9.0	5.2	6.4	9.5	9.9	9.4	9.2	14.5	11.0	8.9	18.3	11.6

	<i>Expand Rank</i>	9.5	5.3	6.6	9.5	10.2	9.5	9.1	14.4	10.8	8.7	18.3	11.4
	<i>TPR</i>	8.7	4.9	6.1	9.1	9.5	9.0	8.8	13.8	10.5	8.8	18.0	11.5

Tabelle 14 Evaluation PositionRank. (Quelle: Eigene Tabelle in Anlehnung an Florescu und Caragea (2017, S. 1113))

Die in Tabelle 14 grün markierten Felder repräsentieren die höchsten und damit besten Werte für die jeweiligen Metriken und Datensets. PositionRank konnte bei allen Messungen sehr gute Ergebnisse erzielen. ExpandRank zeigte ebenfalls gute Resultate, wobei angemerkt werden muss, dass diese rechenintensive Methode externe Informationen aus textuell ähnlichen Dokumenten nutzt. PositionRank benötigt ausschließlich die im Dokument vorhandenen Informationen um geeignete Schlüsselwörter zu extrahieren. Wie bereits erwähnt, haben sich die Tests nur auf die jeweiligen Abstracts aus den Datensätzen bezogen. Daher unterscheidet sich das Verhalten der Methoden für jeden Datensatz nur minimal. Eine Steigerung der Scores ist nur anhand der Anzahl der ermittelten Top Schlüsselwörter erkennbar.

5.4.4 Evaluationsergebnisse aktueller Verfahren

Obwohl durch die in Tabelle 4 gezeigten Datensets Evaluationsvergleiche zwischen allen Schlüsselwort-Extraktionsmethoden möglich sind, wird in wissenschaftlichen Publikationen (Beliga 2014, Hasan und Ng 2014, Bharti, Babu und Jena 2017, Gupta 2017) welche einen thematischen Überblick verschaffen gar nicht oder nur auf vereinzelte Ergebnisse eingegangen. In Tabelle 15 sind, gemäß den Angaben von Hasan und Ng (2014, S. 1267 ff), die besten Scores für gängige Datensets gelistet, anhand welcher zwei Feststellungen getroffen wurden. Zum einen lässt sich mithilfe von Tabelle 4 erkennen, dass die Precision- (P), Recall- (R) und F-Scores (F) bei längeren Dokumenten abnehmen. Des Weiteren zeichnet sich ab, dass unbeaufsichtigte Ansätze (Liu, et al. 2009b, Grineva, Grinev und Lizorkin 2009, Wan und Xiao 2008) bessere Ergebnisse erzielen als überwachte.

Datenset	Schlüsselwort-Extraktionsmethode	Score		
		P	R	F
<i>Abstracts (Inspec)</i>	Topic clustering (Liu, et al. 2009b)	35.0	66.0	45.7
<i>Blogs</i>	Topic community detection (Grineva, Grinev und Lizorkin 2009)	35.1	61.5	44.7
<i>News (DUC-2001)</i>	Graph-based ranking for extended neighborhood (Wan und Xiao 2008)	28.8	35.4	31.7
<i>Papers (SemEval-2010)</i>	Statistical, semantic and distributional features (Lopez und Romary 2010)	27.2	27.8	27.5

Tabelle 15 Top Score Evaluation 2014. (Quelle: Eigene Tabelle in Anlehnung an Hasan und Ng (2014, S. 1268))

6 Praktische Ausarbeitung

Aufbauend auf dem vorab theoretisch erarbeiteten Grundlagenwissen wird in diesem Kapitel die praktische Ausarbeitung des entwickelten Verfahrens beschrieben. Diese beinhaltet den kompletten Prozess dessen Zielsetzung es ist, aus Texten eine Visualisierung für die Analyse konvergierender Forschungsgebiete zu erstellen. Hierbei lässt sich der Prozessablauf wie in Abbildung 10 dargestellt in fünf Teilprozesse gliedern, welche in diesem Kapitel in den gleichnamigen Abschnitten bzw. Unterabschnitten genauer definiert werden.

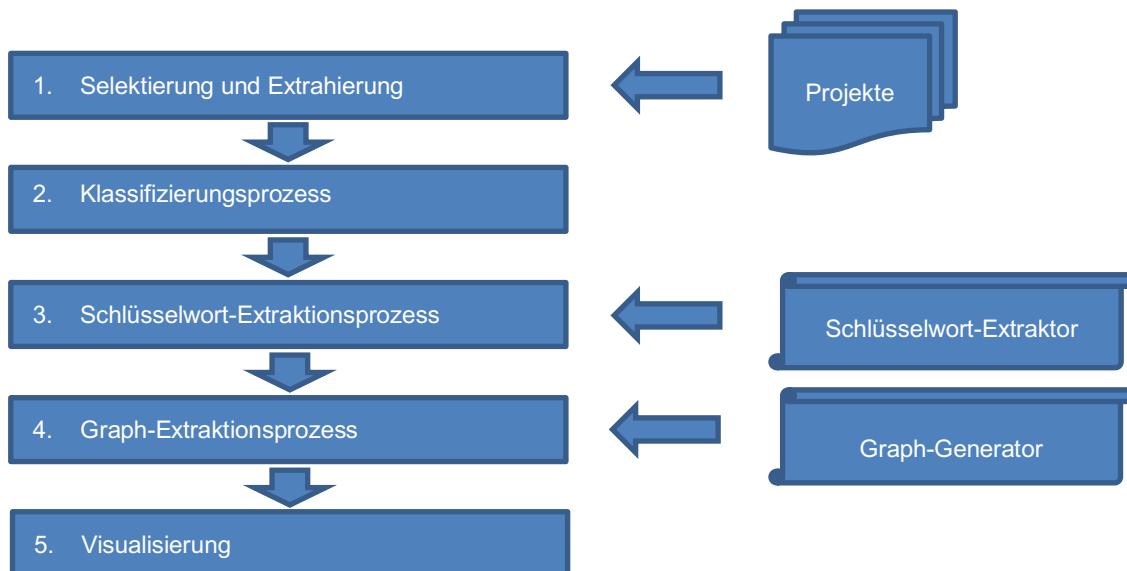


Abbildung 10 Prozess der praktischen Ausarbeitung. (Quelle: Eigene Darstellung)

Im ersten Teilprozess werden die zu untersuchenden Projektdaten selektiert und deren benötigte Informationen extrahiert. Anhand dieser Informationen wird im zweiten Schritt eine Klassifizierung der Projekte durchgeführt. Mithilfe einer Schlüsselwort-Extraktorklasse erfolgt daraufhin die Extraktion der Projektschlüsselwörter. Hierfür werden vorab die einzelnen Schlüsselwort-Extraktionsverfahren überprüft, um das Verfahren mit dem qualitativ besten Ergebnis für den Prozess zu nutzen. Daraufhin erfolgt die Erstellung eines Graphen mithilfe der dafür implementierten Graph-Generatorklasse. Die Umsetzung der Prozessschritte eins bis vier werden in der Programmiersprache Python realisiert. Das daraus resultierende Ergebnis wird abschliessend visualisiert und am Ende dieses Kapitels präsentiert. Alle Informationen für die Prozessschritte werden in den Abschnitten Daten, automatische Textklassifizierung, automatische Schlüsselwort-Extraktion, Graphaufbau und Visualisierung genauer erläutert.

Die Resultate der Prozessschritte eins bis drei können anhand eines Beispielprojektes im Anhang A exemplarisch nachvollzogen werden.

6.1 Daten

In diesem Abschnitt wird die für diese Arbeit verwendete Datenquelle sowie der Vorgang der Datenextraktion und Selektion erläutert.

6.1.1 Quelle

Wie bereits vorab beschrieben, werden im praktischen Teil dieser Arbeit zukünftig anlaufende Forschungsprojekte betrachtet¹. Dafür werden Projekte die der Informationsdienst

¹ Da der Bearbeitungsbeginn dieser Bachelorarbeit auf den 03.12.2018 datiert ist, werden alle Forschungsprojekte mit einem späteren Startdatum als zukünftig anlaufend betrachtet.

der Gemeinschaft für Forschung und Entwicklung (CORDIS) über seine Onlineplattform zur Verfügung stellt untersucht. CORDIS dient als primäre öffentliche Informationsquelle für Projekte, welche durch die EU in Rahmenprogrammen für Forschung und Innovation finanziert werden. Dafür stellt CORDIS über seine Onlineplattform ein umfassendes und strukturiertes Register über alle Projektinformationen wie beispielsweise Projektdatenblätter, Teilnehmer, Berichte, Leistungen und Links zu frei zugänglichen Veröffentlichungen in englischer Sprache zur Verfügung (European Union, About CORDIS). Das aktuelle EU-Rahmenprogramm für Forschung und Innovation heißt Horizon 2020 und umfasst mit einem Finanzbudget von fast 80 Milliarden Euro über einen Zeitraum von 2014 bis 2020 die bisher höchste Fördersumme (European Union, What is Horizon 2020?). Die Horizon 2020 Projekte dienen für diese Arbeit als Datenquelle, wobei diese als Datenset von CORDIS zum Download zur Verfügung gestellt werden. Insgesamt umfasst dieses Set rund 20000 Dateien im XML Format. Diese beinhalten die folgenden projektbezogenen Informationen: Projekt-ID, Projektakronym, Projektstatus, Förderprogrammbezeichnung, Projekttitle, Zielsetzung, Startdatum, Enddatum, Gesamtkosten, Koordinator, Koordinator Land, Teilnehmer, Teilnehmer Länder sowie gegebenenfalls Referenzen zu bestehenden Zwischen- und Abschlussberichten (European Union, CORDIS - EU research projects under Horizon 2020 (2014-2020)).

6.1.2 Selektierung und Extrahierung

Um die für die Analyse benötigten Daten für zukünftige Forschungsprojekte zu erhalten bedarf es einer Extrahierung und Selektierung der rund 20000 XML Dateien. Hierfür müssen im ersten Schritt die relevanten Informationen aus den Dateien ausgelesen werden. Da in dieser Arbeit der Fokus auf noch nicht begonnene Projekte gelegt wird, erfolgt die Textanalyse anhand des Projekttitles und der Projektzielsetzung. Des Weiteren wird das Projektstartdatum, Projektenddatum und die Projekt-ID für die weitere Bearbeitung benötigt. Die Selektierung der Daten erfolgt anhand des Projektstartdatums, wobei nur Projekte betrachtet werden sollen, welche ab dem 01.01.2019 beginnen. Die Extrahierung und Selektierung der Daten wird mittels der in Abbildung 11 dargestellten Python Funktion realisiert.

```

1 def xmlDatenExtrahierenUndSelektieren( verzeichnis , fruehstesProjektStartDatum = '2019-01-01' ) :
2     from xml.dom import minidom
3     import os
4
5     daten = []
6     for dateiname in os.listdir( verzeichnis ) :
7         if dateiname.endswith( '.xml' ) :
8             xmlDok = minidom.parse( verzeichnis + dateiname )
9         else:
10             continue
11
12         projektID      = xmlDok.getElementsByTagName( 'rcn' )[0].firstChild.data
13         projektTitel    = xmlDok.getElementsByTagName( 'title' )[0].firstChild.data
14         projektZiel     = xmlDok.getElementsByTagName( 'objective' )[0].firstChild.data
15         projektStart    = xmlDok.getElementsByTagName( 'startDate' )[0].firstChild.data
16         projektEnde     = xmlDok.getElementsByTagName( 'endDate' )[0].firstChild.data
17
18         if projektStart >= fruehstesProjektStartDatum :
19             daten.append({
20                 'id' : projektID
21                 'titel' : projektTitel,
22                 'zielbeschreibung' : projektZiel,
23                 'startDatum' : projektStart,
24                 'endDatum' : projektEnde
25             })
26
27     return daten

```

Abbildung 11 Datenextrahierung und Selektierung. (Quelle: Eigene Darstellung)

Die Funktion benötigt zur Ausführung die zwei Übergabeparameter *verzeichnis* und *fruehstesProjektStartDatum*, wobei Ersterer den Pfad zum Horizon 2020 Datensatz und Letzterer das früheste Startdatum der zu betrachtenden Projekte repräsentiert. Des Weiteren verwendet die Funktion die Module *minidom* und *os*, welche am Anfang importiert werden. Das *minidom* Modul wird dabei für die Extrahierung der Informationen aus den jeweiligen XML Dateien und das *os* Modul für die Referenzierung auf die Daten aus dem vorab beschriebenen Verzeichnispfad benötigt. Mithilfe der in Zeile sechs implementierte *for*-Schleife wird durch alle Dateien aus dem Verzeichnispfad iteriert. Diese werden mittels des *minidom* Modules eingelesen und der Variable *xmlDok* als Objekt übergeben. Aufgrund der umschließenden *if-else*-Anweisung wird sichergestellt, dass nur Daten im XML Format betrachtet werden. Daraufhin erfolgt die Extrahierung der benötigten Informationen aus der *xmlDok* Variable, wobei diese durch das jeweilige XML Tag identifiziert und ihren Variablen zugewiesen werden. Um im Nachhinein einen einfachen Zugriff auf die Informationen zu gewährleisten, werden die Variablen als Schlüsselwort-Wert-Paare in einem Dictionary zusammengefasst und der Listenvariable *daten* angehängt. Dabei wird mittels der umschließenden *if*-Anweisung sichergestellt, dass nur die Daten der Projekte der Listenvariable angehängt werden, die im Vergleich zu dem *fruehstesProjektStartDatum* Übergabeparameter denselben oder einen späteren Projektstart aufweisen.

Für diese Arbeit wird der Übergabeparameter *fruehstesProjektStartDatum* mit dem Wert 2019-01-01 bestimmt, infolgedessen 979 Datensätze aus den ursprünglich rund 20000 betrachtet werden.

6.2 Automatische Textklassifizierung

Um die zuvor extrahierten und selektierten Projektdaten effizient in Kategorien einzuteilen wird eine automatische Textklassifizierung durchgeführt. In diesem Abschnitt wird das praktische Vorgehen für die automatischen Textklassifizierung erläutert. Zu diesem Zweck erfolgt im ersten Teil eine Beschreibung des dafür verwendeten Python Tools, daraufhin wird auf den praktischen Klassifizierungsprozess eingegangen.

6.2.1 Tool

Da die von CORDIS zur Verfügung gestellten Projektdaten nicht vorab in Kategorien eingeteilt wurden, wird in dieser Arbeit auf die Natural Language Understanding Klassifizierungslösung von IBM Watson (IBM, Watson Natural Language Classifier) zurückgegriffen. Diese ist in der Lage einem Text eine bereits zur Verfügung stehende Kategorie zuzuweisen. Die von IBM vordefinierten 23 Hauptkategorien sind dabei in einer hierarchischen Struktur mit einer Tiefe von bis zu fünf Ebenen aufgeschlüsselt und erreichen damit eine Gesamtanzahl von 1037 verschiedenen Kategorisierungsmöglichkeiten (IBM, Natural Language Understanding Kategorieklassifizierung). Um einen Überblick über alle Hauptkategorien zu erhalten, werden diese in Abbildung 12 dargestellt.



Abbildung 12 IBM Kategorien. (Quelle: Eigene Darstellung)

Die hierarchische Struktur der Kategorien wird in den folgenden Beispielen verdeutlicht:

Business and Industrial / Energy / Renewable Energy / Hydroelectric Energy

Business and Industrial / Energy / Renewable Energy / Solar Energy

Business and Industrial / Energy / Renewable Energy / Wind Energy

Technology and Computing / Hardware / Computer / Desktop Computer

Technology and Computing / Hardware / Computer / Portable Computer

Technology and Computing / Hardware / Computer / Portable Computer / Tablet

wobei jede Ebene durch ein / getrennt wird.

IBM macht keine genaueren Angaben über verwendete Klassifizierungsvorgehen oder Algorithmen. Daher wird in Kapitel 4 ein allgemeines Grundverständnis für die automatische Textklassifizierung vermittelt.

6.2.2 Klassifizierungsprozess

Für die Klassifizierung bietet IBM ein in dieser Arbeit genutztes Application Programm Interface (API) für Python an, welches anhand von Texten Kategorien bestimmt (IBM, Natural Language Understanding Categories). Die Klassifizierung der vorab extrahierten und selektierten Projektdaten wird mittels der in Abbildung 13 dargestellten Python Funktion realisiert.

```

1 def projekteKategorisieren( daten , version , apiSchluessel , watsonURL ) :
2     from watson_developer_cloud import NaturalLanguageUnderstandingV1
3     from watson_developer_cloud.natural_language_understanding_v1 import Features, CategoriesOptions
4
5     nlus = NaturalLanguageUnderstandingV1(
6         url = watsonURL,
7         version = version,
8         iam_apikey = apiSchluessel
9     )
10
11    for idx , projekt in enumerate( daten ) :
12        text = projekt[ 'titel' ] + ' ' + projekt[ 'zielbeschreibung' ]
13        kategorien = nlus.analyze(
14            text = text,
15            features = Features( categories=CategoriesOptions() ).get_result()
16
17        kategorie = kategorien[ 'categories' ][0][ 'label' ]
18
19        kategorieArray = kategorie.split( ' ' )[1:]
20        dictKategorieEbenen = dict( ( 'Ebene' + str( i + 1 ) , k ) for i , k in enumerate( kategorieArray ) )
21
22        daten[idx][ 'kategorie' ] = dictKategorieEbenen
23
24    return daten

```

Abbildung 13 Daten Klassifizierung. (Quelle: Eigene Darstellung)

Die Funktion benötigt zur Ausführung die vier Übergabeparameter *daten*, *version*, *apiSchluessel* und *watsonURL*. Der Parameter *daten* repräsentiert dabei die vorab extrahierten und selektierten Projektdaten. Die restlichen drei Parameter *version*, *apiSchluessel* und *watsonURL* werden für die Authentifizierung des API Services benötigt und sind durch eine Registrierung auf der IBM Cloud Plattform erhältlich (IBM, IBM Cloud Registrierung). Des Weiteren verwendet die Funktion die Module *NaturalLanguageUnderstandingV1*, *Features* und *CategoriesOptions*, wobei das erste Modul für die Authentifizierung des API Services und die anderen Module für die Konfiguration der Klassifizierung benötigt werden. In den Zeilen fünf bis neun erfolgt die Authentifizierung des Natural Language Understanding Services, welcher als Objekt der Variable *nlus* zugewiesen wird.

Daraufhin werden die in der Listenvariable *daten* hinterlegten Projektdaten mittels einer *for*-Schleife iteriert. Mithilfe der *enumerate* Funktion kann neben den Projektdaten auch der Index der Listenvariable durchlaufen werden, welcher als *idx* bezeichnet wird. Innerhalb der Schleife wird die *text* Variable durch den jeweiligen Projekttitel und der Projektzielbeschreibung zusammengesetzt. Diese wird im Anschluss gemeinsam mit den importierten *Features* und *CategoriesOptions* Modulen an die *analyze* Methode des *nlus*-Objektes als Parameter übergeben. Die Resultate der *analyze* Methode sind die drei relevantesten und auf bis zu fünf Ebenen verfeinerten Kategorien der vorab übergebenen *text* Variable. Diese werden der *kategorien* Variable zugewiesen, wobei aus dieser direkt im Anschluss die relevante Kategorie in die *kategorie* Variable extrahiert wird. Da die Kategorie als zusammenhängender String mit dem / Zeichen als Ebenentrennung angegeben wird, erfolgt in Zeile 19 eine Aufteilung der einzelnen Ebenen in die *kategorieArray* Variable. Diese wird daraufhin in einer *for*-Schleife iteriert, wobei die einzelnen Ebenen als Schlüsselwort-Wert-Paare in einem Dictionary zusammengefasst werden. Dementsprechend wird beispielsweise der Kategoriestring:

Science / Biology / Zoology

in ein Dictionary mit den Schlüsselwort-Wert-Paaren:

```
{ 'Ebene1' : 'Science' , 'Ebene2' : 'Biology' , 'Ebene3' : 'Zoology' }
```

umgewandelt. Dieses Dictionary wird im Anschluss dem richtigen Projektdatensatz anhand des Indexwertes *idx* der *daten* Listenvariable übergeben.

Obwohl eine Evaluation der Ergebnisse aufgrund von fehlenden Testdaten nicht möglich ist, wird in dieser Arbeit die Annahme getroffen, dass der IBM Natural Language Understanding Klassifizierungsalgorithmus für jeden Text die korrekte Kategorie definiert.

6.3 Automatische Schlüsselwort-Extraktion

Um für die zuvor extrahierten, selektierten und kategorisierten Projektdaten effizient Schlüsselwörter zu erkennen, wird eine automatische Schlüsselwort-Extraktion durchgeführt. In diesem Abschnitt wird das praktische Vorgehen für die automatische Schlüsselwort-Extraktion erläutert. Dafür wird im ersten Unterabschnitt auf den Aufbau der Schlüsselwort-Extraktorklasse eingegangen. Der zweite Unterabschnitt beinhaltet eine Beschreibung der Verfahrensimplementierung. Darauf folgt die Erläuterung des Schlüsselwort-Extraktionsprozesses. Zum Schluss werden die Evaluationsergebnisse der Verfahren präsentiert.

6.3.1 Schlüsselwort-Extraktorklasse

In diesem Unterabschnitt wird die Implementierung der Schlüsselwort-Extraktorklasse beschrieben. Mithilfe dieser wird während des Schlüsselwort-Extraktionsprozesses ein Objekt instanziert, welches das jeweilige Schlüsselwort-Extraktionsverfahren ausführt. Die in Abbildung 14 dargestellte Klasse beinhaltet den Konstruktor *__init__* sowie die Methoden *setText*, *schluesselwoerterExtrahieren*, *stammformreduktion*, *yake*, *keaLernprozess*, *kea* und *positionRank*. In diesem Unterabschnitt wird detaillierter auf die Implementierung des Konstruktors sowie der ersten drei vorab genannten Methoden eingegangen. Die in Abbildung 14 dargestellten Methoden ohne Körper werden im folgenden Unterabschnitt beschreiben.

```

1 | class SchluesselwortExtraktor :
2 |
3 |     def __init__( self ,
4 |                     stoppwortListe = '/Pfad/Zu/SWListe/stoppwortliste.txt' ,
5 |                     pfadTrainingsProjekte = '/Pfad/Zu/TrainingsProjekten' ,
6 |                     pfadKeaModell = '/Pfad/Zu/Ausgabeordner/model.pickle' ,
7 |                     pfadSchluesselwoerter = '/Pfad/Zu/Schluesselwoerter/schluesselwoerter.txt' ,
8 |                     pfadDokumentenHaeufigkeit = '/Pfad/Zu/Ordner/KandidatenDokumentenHaeufigkeit.tsv.gz' ,
9 |                     anzahlSchluesselwoerter = 10 ) :
10 |
11 |         import spacy
12 |
13 |         self.anzahlSchluesselwoerter = anzahlSchluesselwoerter
14 |         self.nlp = spacy.load( 'en_core_web_lg' )
15 |         self.stoppwortListe = stoppwortListe
16 |         self.pfadTrainingsProjekte = pfadTrainingsProjekte
17 |         self.pfadKeaModell = pfadKeaModell
18 |         self.pfadSchluesselwoerter = pfadSchluesselwoerter
19 |         self.pfadDokumentenHaeufigkeit = pfadDokumentenHaeufigkeit
20 |
21 |     def setText( self , text ) :
22 |         self.text = text
23 |
24 |     def schluesselwoerterExtrahieren( self , extraktionsVerfahren ) :
25 |
26 |         if extraktionsVerfahren == 'YAKE' :
27 |             return self.yake()
28 |         if extraktionsVerfahren == 'KEA' :
29 |             return self.kea()
30 |         if extraktionsVerfahren == 'POSITIONRANK' :
31 |             return self.positionRank()
32 |
33 |     def stammformreduktion( self , daten ):
34 |
35 |         for i in daten :
36 |             schluesselwort = daten[ i ]
37 |             doc = self.nlp( schluesselwort )
38 |             schluesselwort = ''.join( [ token.lemma_ for token in doc ] )
39 |             daten[ i ] = schluesselwort
40 |
41 |         return daten
42 |
43 |     def yake( self ) :
44 |         ...
45 |
46 |     def keaLernprozess( self ) :
47 |         ...
48 |
49 |     def kea( self ) :
50 |         ...
51 |
52 |     def positionRank( self ) :
53 |         ...
54 |
55 | ...
56 |
57 | ...
58 |
59 | ...
60 |
61 | ...
62 |
63 | ...
64 |
65 | ...
66 |
67 | ...
68 |
69 | ...
70 |
71 | ...
72 |
73 | ...
74 |
75 | ...
76 |
77 | ...
78 |
79 | ...
80 |
81 | ...
82 |
83 | ...
84 |
85 | ...
86 |
87 | ...
88 |
89 |
90 |
91 |
92 |
93 |
94 |
95 |
96 |
97 |
98 |
99 |
100 |
101 |
102 |
...

```

Abbildung 14 Implementierung Schlüsselwort-Extraktorklasse. (Quelle: Eigene Darstellung)

__init__

Der Konstruktor wird bei jeder Instanziierung eines neuen Objektes der Schlüsselwort-Extraktorklasse aufgerufen und fordert optional als Übergabeparameter *stoppwortListe*, *pfadTrainingsProjekte*, *pfadKeaModell*, *pfadSchluesselwoerter*, *pfadDokumentenHaeufigkeit* und *anzahlSchluesselwoerter*. Diese werden für die Ausführung der Schlüsselwort-Extraktionsverfahren benötigt und in deren Erläuterungen genauer definiert. Im Körper des Konstruktors werden die Übergabeparameter den gleichnamigen Variablen zugewiesen, wobei diese durch die vorstehende *self* Bezeichnung in allen Klassenmethoden welche *self* als Parameter übergeben bekommen zur Verfügung stehen. Des Weiteren erfolgt der Import von *spacy*, eine Open-Source-Softwarebibliothek für die Verarbeitung von natürlicher Sprache (ExplosionAI GmbH, spaCy), durch die daraufhin ein englisches Sprachmodell geladen und der Variable *nlp* zugewiesen wird.

setText

Die Methode `setText` benötigt als Übergabeparameter den Text aus welchem die Schlüsselwörter extrahiert werden sollen. Dieser wird der Variable `text` zugewiesen.

schluesselwoerterExtrahieren

Die Methode `schluesselwoerterExtrahieren` benötigt als Übergabeparameter den Namen des gewünschten Schlüsselwort-Extraktionsverfahrens als String. In Abhängigkeit davon ob der Name YAKE, KEA oder POSITIONRANK übergeben wurde, wird daraufhin die entsprechende Methode aufgerufen.

stammformreduktion

Die Methode `stammformreduktion` erwartet als Übergabeparameter `daten`, ein Dictionary mit Schlüsselwort-Wert-Paaren, wobei der Wert jeweils ein Schlüsselwort repräsentiert. Der Parameter wird im Anschluss in einer `for`-Schleife durchlaufen. Auf der Grundlage des im Konstruktor initialisierten `nlp`-Objektes wird das jeweilige Schlüsselwort auf seine Stammform reduziert. Da es sich bei dem Schlüsselwort auch um eine Phrase aus zusammenhängenden Wörtern handeln kann, erfolgt dieser Vorgang iterativ pro Wort. Das auf seine Stammform reduzierte Schlüsselwort wird im Anschluss wieder der vorherigen Position im `daten` Dictionary zugewiesen. Nachdem die `for`-Schleife den kompletten Datensatz durchlaufen hat wird dieser zurückgegeben.

6.3.2 Schlüsselwort-Extraktionsverfahren

In dieser Arbeit werden in der praktischen Ausarbeitung die drei bereits in Abschnitt 5.4 beschriebenen Schlüsselwort-Extraktionsverfahren YAKE!, KEA und PositionRank für die Extrahierung von Schlüsselwörtern verwendet. In diesem Unterabschnitt wird auf die Anwendung dieser drei bereits als Open Source Python Module zur Verfügung stehenden Verfahren eingegangen. Campos, et al. (2018a, S. 688) (2018b, S. 806) benennen in ihren Publikationen die Quelle für eine implementierte Python Bibliothek mit YAKE! (Campos, Mangaravite, et al. 2018c) welche in dieser Arbeit genutzt wird. Die Python Implementatioen von KEA und PositionRank stammen aus der in der wissenschaftlichen Publikation (2016) von Florian Boudin ausgeführten python keyphrase extraction (pke) Kollektion (Boudin 2018b). Die folgenden Methoden sind Teil der vorab definierten Schlüsselwort-Extraktorklasse.

YAKE!

Die in Abbildung 15 dargestellte Methode wird für die Schlüsselwort-Extraktion mit dem Verfahren YAKE! aufgerufen. Folgend wird die Funktionsweise dieser definiert.

```

42 |     def yake( self ):
43 |         import yake
44 |
45 |         extractor = yake.KeywordExtractor(
46 |             lan = 'en',
47 |             n = 3 ,
48 |             dedupLim = 0.8 ,
49 |             windowsSize = 2 ,
50 |             top = self.anzahlSchluesselwoerter )
51 |
52 |         schluesselwoerter = extractor.extract_keywords( self.text )
53 |         schluesselwoerter = list( ( y , x ) for x , y in schluesselwoerter )
54 |
55 |         return schluesselwoerter

```

Abbildung 15 Implementierung YAKE! Schlüsselwort-Extraktion. (Quelle: Eigene Darstellung)

Im ersten Schritt erfolgt der Import der `yake`-Bibliothek. Daraufhin wird das KeywordExtractor-Objekt initialisiert und der Variable `extractor` zugewiesen. Die dafür übergebenen Parameter lassen sich wie folgt beschreiben:

- **lan**
Der Parameter `lan` drückt die Sprache aus, in welcher der zu bearbeitende Text verfasst ist. Die Bezeichnung `en` steht für Englisch.
- **n**
Der Parameter `n` steht für das N in N-Gramme und beschreibt die maximale Länge der zu ermittelnden Schlüsselwortphrasen. In dieser konkreten Implementierung können die Schlüsselwortphrasen aus maximal drei Wörtern bestehen.
- **dedupLim**
Der Parameter `dedupLim` gibt den Schwellenwert für die Berechnung der Levenshtein distance an, der wie bereits durch Campos et. al. (2018a, S. 688) für die Evaluation von YAKE! bestimmt, mit 0.8 angegeben wird.
- **windowSize**
Der Parameter `windowSize` gibt den Wort-Fenster-Wert für die Links- und Rechtsberechnung der Wortverwandtschaft zum Kontext (WRel) an, welcher in dieser Implementierung mit zwei bestimmt wird.
- **top**
Der Parameter `top` gibt die maximale Anzahl der relevantesten Schlüsselwörter an die durch die Extraktion zurückgegeben werden. Dieser wird mittels der mit `self` aufgerufenen Variable `anzahlSchluessewoerter` bestimmt.

Im Anschluss werden die Schlüsselwörter anhand der `extract_keywords` Methode aus dem Text des Übergabeparameters `text` extrahiert und als Tupelliste der Variable `schluessewoerter` zugewiesen. Dabei geht die Methode nach den in Unterabschnitt 5.4.1 beschriebenen Schritten vor. Die jeweiligen Tupel beinhalten den berechneten Relevanzwert und das Schlüsselwort. Um eine einheitliche Bearbeitung im Nachhinein zu gewährleisten, werden die Inhalte der Tupel unter Einsatz einer `for`-Schleife andersherum angeordnet und zum Schluss zurückgegeben.

KEA

Wie bereits in Unterabschnitt 5.4.2 erläutert, handelt es sich bei KEA um ein überwachtes und auf maschinell lernende Algorithmen aufbauendes Schlüsselwort-Extraktionsverfahren, welches in zwei Hauptschritte unterteilt wird. Der erste Schritt, die Erstellung eines Trainingsmodells anhand von bereits mit Schlüsselwörtern versehenen Dokumenten, wird in Abbildung 16 dargestellt. Darauf folgt der in Abbildung 17 präsentierte zweite Schritt, der Schlüsselwörter mithilfe des zuvor erstellten Trainingsmodells extrahiert. Bei der Implementierung von KEA wurde in dieser Arbeit gemäß der Ausführung von Boudin (pke 1.8 documentation) vorgegangen. Im Folgenden wird die Funktionsweise des Lernprozesses für die Erstellung des Trainingsmodells beschreiben.

```

57 def keaLernprozess( self ) :
58     from pke import compute_document_frequency
59     import pke
60
61     compute_document_frequency(
62         input_dir = self.pfadTrainingsProjekte ,
63         output_file = self.pfadDokumentenHaeufigkeit ,
64         extension = 'txt' ,
65         language = 'en' ,
66         normalization = 'stemming' ,
67         stoplist = self.stopwortListe ,
68         n = 3 )
69
70     dokumentenHaeufigkeit = pke.load_document_frequency_file(
71         input_file = self.pfadDokumentenHaeufigkeit ,
72         delimiter = '\t' )
73
74     pke.train_supervised_model(
75         input_dir = self.pfadTrainingsProjekte ,
76         reference_file = self.pfadSchluesselwoerter ,
77         model_file = self.pfadKeaModell ,
78         extension = 'txt' ,
79         language = 'en' ,
80         normalization = 'stemming' ,
81         df = dokumentenHaeufigkeit ,
82         model = pke.supervised.Kea() )

```

Abbildung 16 Implementierung KEA Lernprozess. (Quelle: Eigene Darstellung)

Im ersten Schritt erfolgt der Import der *pke*-Bibliothek und der *compute_document_frequency* Funktion. Daraufhin wird direkt die *compute_document_frequency* Funktion aufgerufen, welche die Dokumentenhäufigkeit aller potenziellen Schlüsselwortkandidaten ermittelt und in einer komprimierten Datei im *tsv.gz* Format speichert. Die dafür übergebenen Parameter lassen sich gemäß Boudin (Document frequency counts) wie folgt definieren:

- **input_dir**
Der Parameter *input_dir* gibt den Pfad zu den Trainingstexten an, die getrennt von den dazugehörigen Schlüsselwörtern in einem Ordner in einzelnen Dateien abgelegt sind. Dieser wird mittels der mit *self* aufgerufenen Variable *pfadTrainingsProjekte* bestimmt.
- **output_file**
Der Parameter *output_file* gibt den Speicherpfad und Namen der vorab benannten komprimierten *tsv.gz* Datei an. Dieser wird mithilfe der mit *self* aufgerufenen Variable *pfadDokumentenHaeufigkeit* bestimmt.
- **extension**
Der Parameter *extension* beschreibt die Dateiendung der Trainingsdaten, auf welche der zuvor definierte *input_dir* Parameter verweist. Da es sich bei den Trainingsdaten um *txt* Dateien handelt, wird *extension* dementsprechend bestimmt.
- **language**
Der Parameter *language* drückt die Sprache aus, in welcher die Trainingstexte verfasst wurden. Die Bezeichnung *en* steht für Englisch.
- **normalization**
Der Parameter *normalization* gibt das Verfahren für die Normalisierung der Schlüsselwortkandidaten an. Die Normalisierung in dieser Implementierung wird

anhand einer Stammformreduktion nach Porter (1980) durchgeführt. Dafür wird der Parameter mit dem String *stemming* bestimmt.

- **stoplist**
Der Parameter *stoplist* gibt den Pfad zur Stopwortliste an. Dieser wird mittels der mit *self* aufgerufenen Variable *stopwortListe* bestimmt.
- **n**
Der Parameter *n* steht für das N in N-Gramme und beschreibt die maximale Länge der Schlüsselwortkandidaten. In dieser konkreten Implementierung können die Schlüsselwortkandidaten aus maximal drei Wörtern bestehen.

Im Anschluss werden die zuvor ermittelten und als *tsv.gz* Datei gespeicherten Dokumentenhäufigkeiten der Variable *dokumentenHaeufigkeit* unter Einsatz des Methodenaufrufes *load_document_frequency_file* zugewiesen. Da die Dokumentenhäufigkeiten als tab-sprung-separierte-Werte in der Datei gespeichert werden, wird der Parameter *delimiter*, welcher die Art der Wertetrennung angibt, mit *\t* bestimmt. Der Parameter *input_file* entspricht der vorab definierten Beschreibung von *output_file*. Zum Schluss wird die Methode *train_supervised_model* aufgerufen, um das Trainingsmodell wie in Unterabschnitt 5.4.2 erläutert zu erstellen und zu speichern. Die dafür übergebenen Parameter lassen sich gemäß Boudin (Training supervised models) wie folgt definieren, wobei bereits vorab genannte Parameter nicht erneut erwähnt werden:

- **reference_file**
Der Parameter *reference_file* gibt den Dateipfad zu den für die Trainingsdokumente vorab definierten Schlüsselwörtern an. Die Schlüsselwörter befinden sich in einer Testdatei, welche durch die mit *self* aufgerufene Variable *pfadSchluesSelwoerter* referenziert wird.
- **model_file**
Der Parameter *model_file* gibt den Speicherpfad und Namen des erstellten Trainingsmodells an. Das Modell wird dabei im pickle Dateiformat abgespeichert. Der Parameter wird mittels der mit *self* aufgerufenen Variable *pfadKeaModell* bestimmt.
- **df**
Der Parameter *df* gibt die Dokumentenhäufigkeit der Schlüsselwortkandidaten an und wird anhand der vorab definierten Variable *dokumentenHaeufigkeit* bestimmt.
- **model**
Dem Parameter *model* wird eine Kea Instanz übergeben, welche für die Erstellung des Trainingsmodells notwendig ist.

Nachdem das Trainingsmodell erstellt wurde kann durch den Aufruf der in Abbildung 17 dargestellten Methode die Extrahierung der Schlüsselwörter mittels des in Unterabschnitt 5.4.2 beschriebenen KEA Verfahrens durchgeführt werden. Im Folgenden wird die Funktionsweise erläutert.

```

84 | def kea( self ) :
85 |     import pke
86 |
87 |     extractor = pke.supervised.Kea()
88 |
89 |     extractor.load_document(
90 |         input = self.text ,
91 |         language= 'en' ,
92 |         normalization = None)
93 |
94 |     extractor.candidate_selection( stoplist = self.stoppwortListe )
95 |     df = pke.load_document_frequency_file( input_file = self.pfadDokumentenHaeufigkeit )
96 |     extractor.candidate_weighting( model_file = self.pfadKeaModell , df = df )
97 |
98 |     schlüsselwoerter = extractor.get_n_best( n = self.anzahlSchlüsselwoerter )
99 |
100| return schlüsselwoerter

```

Abbildung 17 Implementierung KEA Schlüsselwort-Extraktion. (Quelle: Eigene Darstellung)

Im ersten Schritt erfolgt der Import der *pke*-Bibliothek. Daraufhin wird das KEA-Objekt instanziert und der Variable *extractor* übergeben. Unter Einsatz der *load_document* Methode wird der Text geladen, für welchen die Schlüsselwörter extrahiert werden sollen. Daraufhin werden die Schlüsselwortkandidaten durch die *candidate_selection* Methode selektiert. Damit die Gewichtung der Schlüsselwortkandidaten erfolgen kann, müssen die zuvor ermittelten und in der *tsv.gz* Datei gespeicherten Dokumentenhäufigkeiten der Variable *df* mittels des Methodenaufrufes *load_document_frequency_file* zugewiesen werden. Mithilfe dieser und des Trainingsmodells kann nun die Gewichtung der Schlüsselwortkandidaten mithilfe der Methode *candidate_weighting* bestimmt werden. Zum Schluss werden die *n* besten Schlüsselwörter extrahiert und zurückgegeben, wobei der Parameter *n* in diesem Fall die Anzahl der zu ermittelnden Schlüsselwörter definiert.

PositionRank

Die in Abbildung 18 dargestellte Methode wird für die Schlüsselwort-Extraktion mit dem in Unterabschnitt 5.4.3 beschriebenen Verfahren PositionRank aufgerufen. Im Folgenden wird die Funktionsweise auf der Grundlage von Boudin (PositionRank) erläutert.

```

102 | def positionRank( self ) :
103 |     import pke
104 |
105 |     pos = { 'NOUN' , 'PROPN' , 'ADJ' }
106 |     grammar = 'NP: {<ADJ>*<NOUN|PROPN>+}'
107 |
108 |     extraktor = pke.unsupervised.PositionRank()
109 |
110 |     extraktor.load_document( input = self.text , language = 'en' , normalization = None )
111 |     extraktor.candidate_selection( grammar = grammar , maximum_word_number = 3 )
112 |     extraktor.candidate_weighting( window = 10 , pos = pos )
113 |
114 |     schlüsselwoerter = extraktor.get_n_best( n = self.anzahlSchlüsselwoerter )
115 |
116 | return schlüsselwoerter

```

Abbildung 18 Implementierung PositionRank Schlüsselwort-Extraktion. (Quelle: Eigene Darstellung)

Im ersten Schritt erfolgt der Import der *pke*-Bibliothek. Daraufhin werden die Wortarten für die Filterung der Wörter bestimmt und der Variable *pos* übergeben. Im Anschluss erfolgt die Definition des grammatischen Aufbaues der zu ermittelnden Schlüsselwörter. Die Bezeichnung *NP: {<ADJ>*<NOUN|PROPN>+}* beschreibt dabei eine Substantivphrase (*NP*), die aus keinem, einem oder mehreren vorstehenden Adjektiven (*ADJ*) und darauffolgenden oder alleinstehenden Substantiven (*NOUN*) bzw. Eigennamen (*PROPN*) gebildet wird. Daraufhin wird ein PositionRank-Objekt instanziert und der Variable *extraktor*

zugewiesen. Unter Einsatz der *load_document* Methode wird der Text geladen, für welchen die Schlüsselwörter extrahiert werden sollen. Etwas anders als in Unterabschnitt 5.4.3 erläutert werden die Wörter für die Erstellung des Graphen durch die Methode *candidate_selection* eingegrenzt, da mittels der vorab definierten Variable *grammar* bereits festgelegt wurde, welche zusammenhängenden Schlüsselwortphrasen extrahiert werden sollen. Auf Grundlage des *maximum_word_number* Parameters wird daraufhin eine Selektion von maximal Trigrammen durchgeführt.

Im Anschluss erfolgt mithilfe des Methodenaufrufes *candidate_weighting* die Kandidaten-gewichtung durch den PageRank Algorithmus. In dem für die Berechnung des jeweiligen Wort Scores erstellten Graph werden die Knoten (nur Adjektive und Substantive bzw. Eigennamen) immer dann miteinander verbunden, sobald diese in einem Fenster von 10 Wörtern innerhalb eines Satzes gemeinsam vorkommen. Boudin (PositionRank) setzt einen Wortfensterwert von 10 in seiner Implementierung als Standard. Gemäß Florescu und Caragea (2017, S. 1110) wirkt sich dieser in einer Spanne von 2 bis 10 nur unwesentlich auf das Ergebnis aus. Des Weiteren wird der Dämpfungsfaktor innerhalb der pke-Bibliothek wie bereits von Brin und Page (1998, S. 109) suggeriert, mit 0.85 bestimmt. Zum Schluss werden die n besten Schlüsselwörter extrahiert und zurückgegeben.

6.3.3 Schlüsselwort-Extraktionsprozess

Die Schlüsselwort-Extraktion der vorab extrahierten, selektierten und kategorisierten Projektdaten wird mittels der in Abbildung 13 dargestellten Funktion realisiert.

```

1  def schluesselwoerterExtrahieren( daten , verfahren = 'YAKE' ) :
2      from SchluesselwortExtraktor import SchluesselwortExtraktor
3
4      extraktor = SchluesselwortExtraktor()
5
6      if verfahren == 'KEA' :
7          extraktor.keaLernprozess()
8
9      for idx , projekt in enumerate( data ) :
10         text = projekt[ 'titel' ] + ' ' + projekt[ 'zielsetzung' ]
11         extraktor.setText( text )
12
13         schluesselwoerterListe = extraktor.schluesselwoerterExtrahieren( verfahren )
14         dictSchluesselwoerter = dict( ( i , s[0] ) for i , s in enumerate( schluesselwoerterListe ) )
15         dictSchluesselwoerter = extraktor.stammformreduktion( dictSchluesselwoerter )
16
17         data[idx][ 'schluesselwoerter' ] = dictSchluesselwoerter
18
19     return daten

```

Abbildung 19 Implementierung Schlüsselwort-Extraktionsprozess. (Quelle: Eigene Darstellung)

Die Funktion benötigt zur Ausführung die zwei Übergabeparameter *daten* und *verfahren*, wobei *verfahren* optional ist. Der Parameter *daten* repräsentiert die vorab extrahierten, selektierten und kategorisierten Projektdaten. *Verfahren* benennt das für die Extraktion angewandte Verfahren, wobei *KEA*, *YAKE* und *POSITIONRANK* zur Auswahl stehen. Im ersten Schritt wird das im Unterabschnitt 6.3.1 beschriebene *SchluesselwortExtraktor* Modul importiert. Mithilfe dessen wird ein gleichnamiges Objekt instanziert und der Variable *extraktor* zugewiesen. Falls die *verfahren* Variable mit *KEA* definiert wurde, erfolgt die Erstellung des Trainingsmodells unter Einsatz des Methodenaufrufes *keaLernprozess*.

Daraufhin werden die in der Listenvariable *daten* hinterlegten Projektdaten mittels einer *for*-Schleife iteriert. Mithilfe der *enumerate* Funktion kann neben den Projektdaten auch der Index der Listenvariable durchlaufen werden, welcher als *idx* bezeichnet wird. Innerhalb der Schleife wird die *text* Variable anhand des jeweiligen Projekttitels und der Projektzielbeschreibung zusammengesetzt. Diese wird im Anschluss der *setText* Methode als Parameter

übergeben. Daraufhin erfolgt die Extrahierung der Schlüsselwörter mittels des mit der Variable *verfahren* beschriebenen Verfahrens, wobei das Resultat als Liste an Tupeln der Variable *schluesselwoerterListe* zugewiesen wird. Die jeweiligen Tupel beinhalten den berechneten Relevanzwert und das Schlüsselwort. Um im Nachhinein eine einfache Handhabung zu gewährleisten, werden die Schlüsselwörter der Tupel durch eine *for*-Schleife als Schlüsselwort-Wert-Paare in einem Dictionary zusammengefasst, wobei die Schlüsselwörter die Werte repräsentieren. Bevor die Schlüsselwörter zum Schluss dem richtigen Projektdatensatz der *daten* Listenvariable zugeordnet werden erfolgt eine Stammformreduktion, damit diese im Nachhinein über alle Projekte vergleichbar sind.

6.3.4 Evaluation

In diesem Unterabschnitt wird auf die Evaluation der drei Schlüsselwort-Extraktionsverfahren eingegangen. Diese erfolgt im Selbstversuch, wobei dafür für insgesamt 90 Projekte die Schlüsselwörter manuell vorab bestimmt wurden. Davon werden 50 für die Erstellung des KEA Trainingsmodells und 40 für die Evaluation genutzt. Die Auswahl der Anzahl an Trainingsdokumenten für KEA liegt darin begründet, dass Witten, et al. (1999, S.15) in ihrer KEA Evaluation gute Ergebnisse ab einem Trainingset von 50 Dokumenten erzielen konnten. Die Parameter der einzelnen Verfahren wurden gemäß der Implementierung aus Unterabschnitt 6.3.2 gesetzt. Als Stopwortliste dient die der YAKE! Bibliothek (Campos, Mangaravite, et al., YAKE! StopwordsList). Für die 40 Evaluationsdokumente wurden die insgesamt 472 Schlüsselwörter maximal als Trigramme vorab bestimmt, woraus sich ein Durchschnitt von 11,8 Schlüsselwörtern pro Dokument ergibt. Die Auswertung erfolgt unter Betrachtung der besten 6, 8, 10, 12 und 14 extrahierten Schlüsselwörtern pro Verfahren. Die Schlüsselwörter wurden dabei nur aufgrund einer exakten Übereinstimmung mit den Vorbestimmten als gefunden deklariert. Die Berechnung der Evaluationsergebnisse Precision (P), Recall (R) und F-Score (F) erfolgte wie in Unterabschnitt 5.3.3 erläutert. Die Resultate werden in Tabelle 16 dargestellt.

Verfahren	Top 6			Top 8			Top 10			Top 12			Top 14		
	P %	R %	F %	P %	R %	F %	P %	R %	F %	P %	R %	F %	P %	R %	F %
YAKE!	25,00	12,71	16,85	23,13	15,68	18,69	19,25	16,31	17,66	18,75	19,07	18,91	18,21	21,61	19,77
KEA	7,08	3,60	4,78	7,50	5,08	6,06	7,25	6,14	6,65	6,67	6,78	6,72	6,79	8,05	7,36
Position-Rank	37,50	19,07	25,28	34,38	23,31	27,78	31,25	26,48	28,67	27,92	28,39	28,15	25,89	30,72	28,10

Tabelle 16 Evaluation Implementierung. (Quelle: Eigene Tabelle)

Anhand der Evaluationsergebnisse lässt sich erkennen, dass PositionRank in diesem Selbstversuch das effektivste Verfahren ist. Die grün markierten Felder drücken die jeweiligen Topwerte pro Schlüsselwortanzahl aus. Dabei konnte der höchste F-Score-Wert (harmonischer Mittelwert von Recall (R) und Precision (P)) bei der Betrachtung der relevantesten 10 Schlüsselwörtern mit 28,67 % durch PositionRank erzielt werden. Bei den vorab bestimmten Schlüsselwörtern handelte es sich im Regelfall um zusammenhängende Phrasen, wodurch KEA, welches oft nur einzelne Worte als Schlüsselwörter erkannte, geringe Resultate erzielte. In Abbildung 20 werden die Resultate über eine Spanne der relevantesten 2 bis 20 Schlüsselwörter in den jeweiligen Precision, Recall und F-Score Diagramm betrachtet.

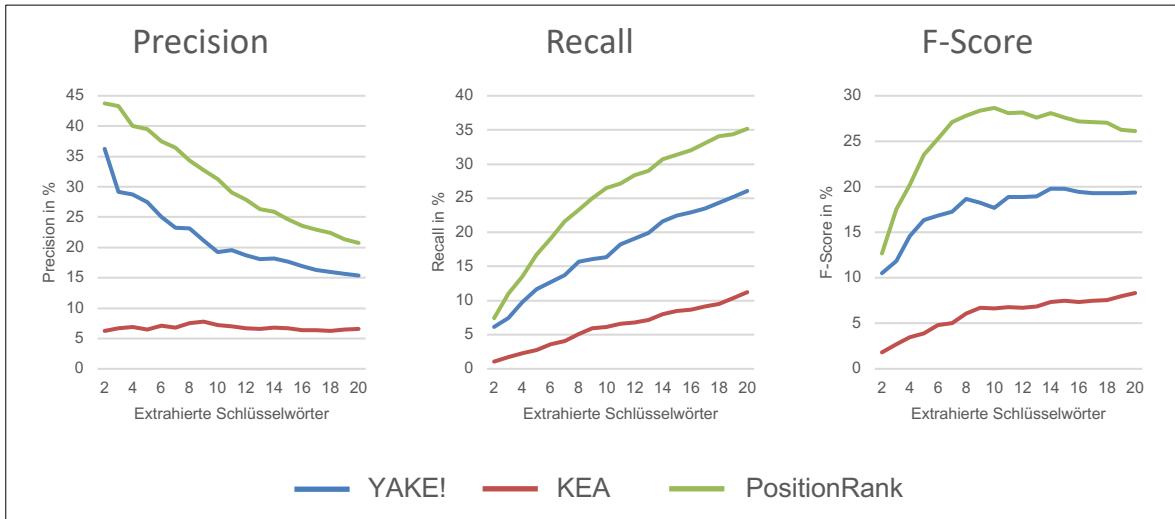


Abbildung 20 Evaluation der Implementierung. (Quelle: Eigene Darstellung)

Dabei ist zu erkennen, dass der Precision-Wert, der ausdrückt wie viel Prozent der durch die Methode gefundenen Schlüsselwörtern tatsächlich relevant sind, bei YAKE! und PositionRank mit der Anzahl an Schlüsselwörtern stark abfällt. Der Wert von KEA bleibt hingegen relativ stabil. Der Recall-Wert, welcher ausdrückt wie viel Prozent aller relevanten Schlüsselwörter mittels der Methode gefunden wurden, steigt bei allen drei Verfahren mit der Anzahl an Schlüsselwörtern. In dem F-Score Diagramm ist wie bereits zuvor in Tabelle 16 zu erkennen, dass der höchste Wert von Position Rank bei der Betrachtung der 10 relevantesten Schlüsselwörter erreicht wurde. Im Anschluss fallen die Werte ab, da der Precision-Wert schneller fällt, als der Recall-Wert steigt. Aufgrund dessen weist diese Schlüsselwortanzahl für PositionRank den besten Ausgleich zwischen der Menge an extrahierten Schlüsselwörtern und deren Gesamtrelevanz auf. Der F-Score von KEA steigt bis zum Schluss aufgrund des nahezu gleichbleibenden Precision-Wertes und des steigenden Recalls.

6.4 Graphaufbau

Um die zuvor kategorisierten und mit Schlüsselwörtern versehenen Projektdaten in einem Netzwerk zu visualisieren, muss ein Graph erstellt werden. In diesem Abschnitt wird das praktische Vorgehen für die Erstellung des Graphen erläutert. Dafür wird im ersten Unterabschnitt auf den Aufbau der Graphen-Generatorklasse eingegangen. Der zweite Unterabschnitt beinhaltet die Beschreibung des Graphen-Erstellungsprozesses.

6.4.1 Graph-Generatorklasse

In diesem Unterabschnitt wird die Implementierung der Graph-Generatorklasse beschrieben. Mithilfe dieser wird ein Objekt instanziert, welches die Kategorien und Schlüsselwörter der Projekte in einem Graphen zusammenfassen. Die Klasse beinhaltet den Konstruktor `__init__` sowie die Methoden `filterSchlüsselwoerter`, `schlüsselwoerterZuGraph`, `kategorienZuGraph` und `schreibeGraphInGmlDatei` und wird in Abbildung 21 dargestellt. Die Graphenerstellung wird mithilfe der NetworkX Bibliothek für Python durchgeführt. Diese eignet sich grundlegend zur Erstellung, Bearbeitung und Untersuchung der Struktur, Dynamik und Funktion komplexer Netzwerke (NetworkX developers). Dafür wird vor der Klassendefinition das NetworkX Modul `nx` importiert. Folgend wird detaillierter auf die Implementierung des Konstruktors sowie der Methoden eingegangen.

```

1 import networkx as nx
2
3 class GraphGenerator :
4
5     def __init__( self , daten ) :
6
7         self.G = nx.Graph()
8         self.daten = daten
9
10    def filterSchluesselwoerter( self ) :
...
11
12    def schluesselwoerterZuGraph( self ) :
...
13
14    def kategorienZuGraph( self ) :
...
15
16    def speicherGraphInGmlDatei( self , dateiname = 'graph.gml' ) :
...

```

Abbildung 21 Implementierung Graph-Generatorklasse. (Quelle: Eigene Darstellung)

__init__

Der Konstruktor wird bei jeder Instanziierung eines neuen Objektes der Graph-Generator-Klasse aufgerufen und fordert als Übergabeparameter die vorab erstellte Listenvariable *daten*, welche die einzelnen Projektedaten enthält. Im Körper des Konstruktors wird der Übergabeparameter der gleichnamigen Variablen zugewiesen. Des Weiteren wird ein Graph-Objekt mithilfe des *nx* Modules instanziert und der Variable *G* übergeben.

filterSchluesselwoerter

Die Methode *filterSchluesselwörter* erstellt eine Gesamtliste der relevanten Schlüsselwörter aller Projekte. Dafür werden alle Schlüsselwörter herausgefiltert, die insgesamt nur einmal über alle Projekte hinweg auftreten und denselben Namen wie eine Kategorie haben. Die Methode wird in Abbildung 22 dargestellt und im Folgenden erklärt.

```

10 def filterSchluesselwoerter( self ) :
11
12     alleKat = []
13     alleSchluesselw = []
14
15     for projekt in self.daten :
16
17         alleKat.extend( map( lambda kategorie : kategorie , projekt[ 'kategorie' ].values() ))
18         alleSchluesselw.extend( map( lambda sw : sw , projekt[ 'schluesselwoerter' ].values() ))
19
20     alleSchluesselw = [ x for x in alleSchluesselw if x not in alleKat ]
21     gefilterteSchluesselw = set( [ sw for sw in alleSchluesselw if alleSchluesselw.count( sw ) > 1 ] )
22
23     return gefilterteSchluesselw

```

Abbildung 22 Implementierung filterSchluesselwoerter. (Quelle: Eigene Darstellung)

Im ersten Schritt erfolgt die Initialisierung der Variablen *alleKat*, *alleSchluesselw* und *gefilterteSchluesselw*. Daraufhin werden die in der Listenvariable *daten* hinterlegten Projektdaten durch eine *for*-Schleife iteriert. Unter Einsatz der *lambda* Funktion werden innerhalb der Schleife aus allen Projekten die Kategorien und Schlüsselwörter den Listenvariablen *alleKat* und *alleSchluesselw* übergeben. Im Anschluss erfolgt die erste Filterung der Schlüsselwörter. Dabei werden nur die Schlüsselwörter erneut an die *alleSchluesselw* Variable übergeben, die sich nicht in der Listenvariable *alleKat* befinden. Daraufhin erfolgt der zweite Filterschritt, wobei nur die Schlüsselwörter der Variable *gefilterteSchluesselw* zugewiesen werden, die insgesamt mehr als einmal über alle Projekte hinweg auftreten. Diese Variable wird zum Schluss als Return-Wert zurückgegeben.

schlüsselwoerterZuGraph

Die Methode *schlüsselwoerterZuGraph* fügt dem im Konstruktor erstellten Graph-Objekt alle Schlüsselwörter mit ihren Verbindungen zu der letzten Kategorieebene des eigenen Projektes zu. Die Schlüsselwörter und Kategorieebenen werden dabei als Knoten und ihre Verbindung als Kanten bezeichnet. Die Kanten haben ein Gewicht, mit dem die Stärke der Beziehung zwischen zwei Knoten dargestellt werden kann. Umso öfter also ein Schlüsselwort in Verbindung mit einer Kategorieebene zusammen auftritt, desto stärker wird ihre Verbindung. Des Weiteren wird für jede Kante ein Attribut mit dem Namen der Grundkategorieebene des verbundenen Kategorie Knotens bestimmt. Anhand dessen können die Verbindungen während der Visualisierung entsprechend der Kategorie eingefärbt werden. Die Methode wird in Abbildung 23 dargestellt und im Folgenden erklärt.

```

25 | def schlüsselwoerterZuGraph( self ) :
26 |
27 |     gefilterteSw = self.filterSchlüsselwoerter()
28 |
29 |     for projekt in self.daten :
30 |
31 |         anzahlEbenen = len( projekt[ 'kategorie' ] )
32 |         letzteKat = projekt[ 'kategorie' ][ 'Ebene' ] + str( anzahlEbenen )
33 |         ersteKat = projekt[ 'kategorie' ][ 'Ebene1' ]
34 |
35 |         relevanteSw = [ sw for sw in projekt[ 'schlüsselwoerter' ].values() if sw in gefilterteSw ]
36 |
37 |         for sw in relevanteSw :
38 |
39 |             if self.G.has_edge( letzteKat , sw ) :
40 |                 self.G[ letzteKat ][ sw ][ 'weight' ] += 1
41 |
42 |             else :
43 |                 self.G.add_path( [ letzteKat , sw ] , Kategorie = ersteKat , weight = 1 )

```

Abbildung 23 Implementierung *schlüsselwoerterZuGraph*. (Quelle: Eigene Darstellung)

Im ersten Schritt wird mithilfe der vorab beschriebenen Methode *filterSchlüsselwoerter* die Liste der gefilterten Schlüsselwörter der Variable *gefilterteSw* übergeben. Daraufhin werden die in der Listenvariable *daten* hinterlegten Projektdaten mittels einer *for*-Schleife iteriert. Dabei wird die Anzahl der Kategorieebenen, die letzte Kategorieebene und die erste Kategorieebene des jeweiligen Projektes den Variablen *anzahlEbenen*, *letzteKat* und *ersteKat* zugewiesen. Im Anschluss erfolgt mithilfe der *gefilterteSw* Variable die Sortierung der relevantesten Schlüsselwörter, wobei nur jene Schlüsselwörter des Projektes beachtet werden, welche in der gefilterten Schlüsselwortliste vorkommen. Die daraus resultierende Listenvariable *relevanteSw* wird daraufhin mithilfe einer *for*-Schleife iteriert. Dadurch wird jedes Schlüsselwort darauf geprüft, ob es bereits eine Verbindung mit der letzten Kategorieebene aufweist. Diese Prüfung erfolgt mit der Graph Methode *has_edge*. Im Falle einer bereits bestehenden Verbindung wird der bestehende Gewichtungswert um eins addiert. Wenn noch keine Verbindung besteht, werden die Knotenpunkte zu dem Graphen hinzugefügt und die Verbindung aufgebaut. Dabei werden die Kantenattribute *Kategorie* mit der ersten Kategorieebene und *weight* mit eins bestimmt.

kategorieZuGraph

Die Methode *kategorieZuGraph* fügt dem im Konstruktor erstellten Graph-Objekt alle Kategorieebenen mit ihren Verbindungen zu der nächst höheren Kategorieebene des jeweiligen Projektes zu. Auch in diesem Fall werden die Kategorieebenen als Knoten und ihre Verbindung als Kanten bezeichnet. Damit die Verbindungen der Kategorieebenen bei der Visualisierung als die stärksten Stränge sichtbar sind, werden ihnen die vorab maximal erreichte Gewichtung, erhöht um zwei, zugewiesen. Die Methode wird in Abbildung 24 dargestellt und im Folgenden erklärt.

```

45 | def kategorieZuGraph( self ) :
46 |
47 |     maxGewicht = max( nx.get_edge_attributes( self.G , 'weight' ).values() )
48 |
49 |     for projekt in self.daten :
50 |
51 |         anzahlKategorie = len( projekt[ 'kategorie' ] )
52 |
53 |         if anzahlKategorie > 1 :
54 |
55 |             for i in range( 1 , anzahlKategorie ) :
56 |                 aktuelleKat = projekt[ 'kategorie' ][ 'Ebene' + str( i ) ]
57 |                 folgendeKat= projekt[ 'kategorie' ][ 'Ebene' + str( i + 1 ) ]
58 |
59 |                 if not self.G.has_edge( aktuelleKat , folgendeKat ) :
60 |                     self.G.add_path(
61 |                         [ aktuelleKat , folgendeKat ] ,
62 |                         Kategorie = projekt[ 'kategorie' ][ 'Ebene1' ] ,
63 |                         weight = maxGewicht + 2 )

```

Abbildung 24 Implementierung kategorieZuGraph. (Quelle: Eigene Darstellung)

Im ersten Schritt wird mithilfe der *max* Funktion und der *get_edge_attributes* Methode, der bisher vergebene maximale Kanten-Gewichtungswert ermittelt und der Variable *maxGewicht* zugewiesen. Daraufhin werden die in der Listenvariable *daten* hinterlegten Projektdaten durch eine *for*-Schleife iteriert. Im Anschluss wird die Anzahl der Kategorieebenen des jeweiligen Projektes ermittelt und der Variable *anzahlKategorie* übergeben. Falls das Projekt in mehr als eine Kategorieebene klassifiziert wurde, werden in einer *for*-Schleife alle von der Ersten bis zur Vorletzten vergebenen Ebenennummern iteriert. Die aktuelle und die folgende Kategorieebene werden den Variablen *aktuelleKat* und *folgendeKat* zugewiesen. Im Anschluss wird mit der *has_edge* Methode des Graphen überprüft, ob die beiden Kategorien bereits als Knoten vorhanden und mittels einer Kante verbunden sind. Falls dies nicht der Fall ist, werden die beiden Knoten und ihre Verbindung dem Graphen hinzugefügt, wobei das Kantenattribut *Kategorie* mit der ersten Kategorieebene und das Kantenattribut *weight* mit dem *maxGewicht* erhöht um zwei, bestimmt wird.

speicherGraphInGmlDatei

Die Methode *schreibeGraphInGmlDatei* schreibt, nachdem der Graph erstellt wurde, diesen in eine Datei im *Gml* Format. Diese Datei kann daraufhin für die Visualisierung des Netzwerkes aus Knoten und Kanten genutzt werden. Die Methode wird in Abbildung 25 dargestellt und im Folgenden erklärt.

```

65 | def speicherGraphInGmlDatei( self , dateiname ) :
66 |
67 |     unrelevanteKnoten = [ node for node , degree in self.G.degree() if degree == 1 ]
68 |     self.G.remove_nodes_from( unrelevanteKnoten )
69 |
70 |     nx.write_gml( self.G , dateiname )

```

Abbildung 25 Implementierung schreibeGraphInGmlDatei. (Quelle: Eigene Darstellung)

Die Methode benötigt als Übergabeparameter den Dateinamen bzw. Pfad für die Speicherung des Graphen. Die Dateiendung muss dabei mit *.gml* angegeben werden. Bevor die Speicherung stattfindet, erfolgt eine Bereinigung des Graphen anhand der Knotengrade. Der Grad eines Knotens repräsentiert die Anzahl an anliegenden Kanten. Da in dieser Bachelorarbeit nur die Beziehung zwischen verschiedenen Forschungsgebieten untersucht wird, werden alle Knoten, die nur eine Verbindung zu einer Kategorie haben, eliminiert. Dafür werden die Grade aller Knoten iteriert und überprüft. Falls ein Grad nur eins beträgt, wird der dazugehörige Knoten von dem Graphen entfernt.

6.4.2 Graph-Erstellungsprozess

Die Graphenerstellung der vorab kategorisierten und mit Schlüsselwörtern versehenen Projektdaten wird durch die in Abbildung 26 dargestellte Funktion realisiert und im Folgenden erläutert.

```
1 | def erstelleGraph( daten , speicherortGraph = 'graph.gml' ) :  
2 |     from GraphGenerator import GraphGenerator  
3 |  
4 |     graphGenerator = GraphGenerator(daten)  
5 |  
6 |     graphGenerator.schluesselwoerterZuGraph()  
7 |     graphGenerator.kategorienZuGraph()  
8 |     graphGenerator.schreibeGraphInGmlDatei( speicherortGraph )
```

Abbildung 26 Implementierung Graph-Erstellungsprozess. (Quelle: Eigene Darstellung)

Die Funktion benötigt zur Ausführung die zwei Übergabeparameter *daten* und *speicherortGraph*, wobei Letzterer optional ist. Der Parameter *daten* repräsentiert hierbei die vorab extrahierten, selektierten, kategorisierten und mit Schlüsselwörtern versehenen Projektdaten. Der Parameter *speicherortGraph* repräsentiert den Dateinamen bzw. den Pfad für die Speicherung des Graphen. Die Dateiendung muss dabei mit *.gml* angegeben werden. Im ersten Schritt wird das vorab beschriebene GraphGenerator Modul importiert. Mit dessen Hilfe wird ein gleichnamiges Objekt instanziert und der Variable *graphGenerator* zugewiesen, wobei eine Übergabe der Listenvariable *daten* erfolgt. Daraufhin werden die Schlüsselwörter und Kategorien dem Graphen hinzugefügt. Zum Schluss erfolgt die Speicherung des Graphen an dem vorab festgelegten Speicherort.

6.5 Visualisierung

Der letzte Schritt dieser praktischen Ausarbeitung ist die Visualisierung des zuvor erstellten Graphen. Zu diesem Zweck wird zuerst auf das dafür verwendete Tool eingegangen, im Weiteren erfolgt die Beschreibung der Ergebnisausarbeitung und Präsentation.

6.5.1 Tool

Die Visualisierung des Graphennetzwerkes wird mit Gephi, einer der führenden Open-Source Visualisierungs- und Explorationssoftware, durchgeführt ([Gephi.org](#), The Open Graph Viz Platform). Die Software wurde von Bastian, Heymann und Jacomy (2009) entwickelt und wird zum freien Download und Nutzen online zur Verfügung gestellt ([Gephi.org](#), [Download](#)).

Gephi bietet die Möglichkeit, die Grapheninformationen aus Dateiformaten wie beispielsweise GDF, GraphML, GML und CSV einzulesen. Des Weiteren ist eine Anbindung an relationale Datenbanken möglich. Für die Anzeige und Erkundung des Netzwerkgraphen verwendet Gephi eine 3D-Render-Engine, um eine Bearbeitung in Echtzeit zu ermöglichen. Dabei bietet sich dem Benutzer die Möglichkeit, mit der Graphendarstellung zu interagieren, um anhand von Manipulation der Strukturen, Formen und Farben verborgene Muster sichtbar zu machen. Neben der Option, dem Netzwerk durch die Anwendung von Layoutalgorithmen, wie beispielsweise ForceAtlas2 (Jacomy, et al. 2014), Fruchterman Reingold (Fruchterman und Reingold 1991), OpenOrd (Martin, et al. 2011) und Yifan Hu (Hu 2005), eine Form zu geben, ermöglicht es die Filterung der Knoten und Kanten, basierend auf der Netzwerkstruktur oder den Daten, einen guten Überblick auf die zu untersuchende Darstellung zu erhalten. Des Weiteren ist Gephi in der Lage, dass dargestellte Netzwerk im PDF oder PNG Format zu exportieren ([Gephi.org](#), Features).

6.5.2 Ergebnisse

Die Erstellung des Graphen erfolgt mittels der vorab beschriebenen Prozessschritte. Aufgrund der Evaluationsresultate aus Unterabschnitt 6.3.4, wird PositionRank für die Schlüsselwort-Extraktion, mit einer Anzahl von jeweils 10 Schlüsselwörtern pro Projekt, genutzt. Infolge dessen, dass während des Graphen-Erstellungsprozesses jeder Verbindungskante die jeweilige Hauptkategorie zugewiesen wurde, lassen sich diese durch Gephi in Abhängigkeit der Kategorie einfärben. Ein visueller Gesamtüberblick über alle vorkommenden Hauptkategorien und ihre Verzweigungen wird in Abbildung 27 dargestellt.

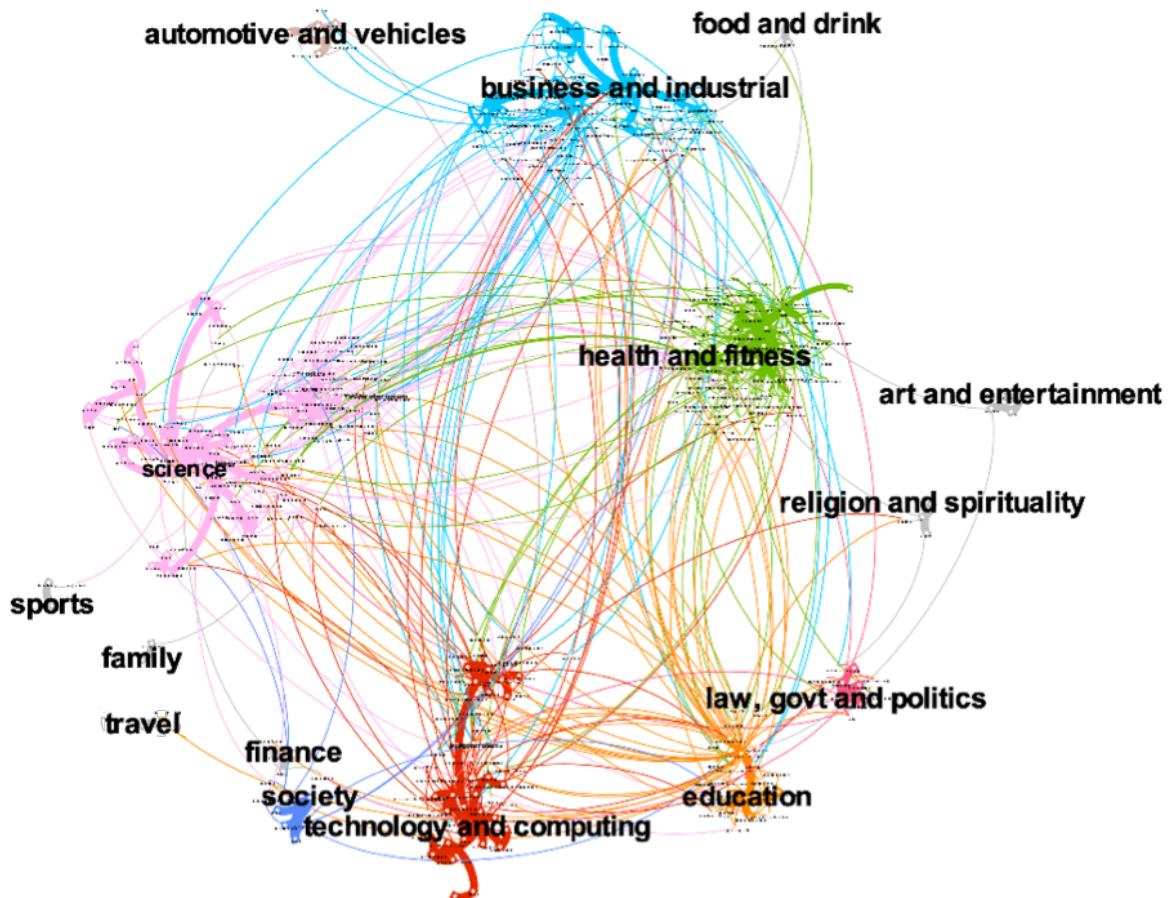


Abbildung 27 Visualisierung Kategorie Übersicht. (Quelle: Eigene Darstellung)

Für die Erstellung dieser Visualisierung wurden die Hauptkategorien, mithilfe des OpenOrd Algorithmus in alleinstehende Cluster separiert. Im Anschluss wurden die Labels der Hauptkategorieebenen vergrößert dargestellt. Die dickeren Verbindungen stellen hierbei die Kanten zwischen den jeweiligen Kategorieebenen der jeweiligen Hauptkategorie dar. Die dünnen sichtbaren Verbindungen, welche sich zwischen den einzelnen Kategorieclustern aufspannen, verbinden Schlüsselwörter zwischen den Kategorien.

Des Weiteren lässt sich durch Gephi der prozentuale Anteil der Verbindungen pro Kategorie bestimmen. In Tabelle 17 werden die Hauptkategorien mit ihren prozentualen Anteilen und Farben dargestellt.

Farbe	Kategorie	Anteil in %
pink	Science	26,64
green	Health and Fitness	21,46
blue	Business and Industrial	17,35
orange	Technology and Computing	14,07
yellow	Education	10,93
red	Law, Govt and Politics	2,46
light blue	Society	2,05
brown	Automotive and Vehicles	1,23
grey	7 Weitere	3,81

Tabelle 17 Visualisierung Kategorie Übersicht. (Quelle: Eigene Tabelle)

Insgesamt wurden 15 der 23 zu Verfügung stehenden Hauptkategorien mittels die automatischen Textklassifizierung vergeben. Als thematisch relevanste Hauptkategorien, mit einem Anteil von über 15 %, lassen sich Science, Health and Fitness sowie Business and Industrial bestimmen. Diese wurden in verschiedenen Variationen in einem Netzgraphen gegenübergestellt. Die Ergebnisse können im Anhang wie folgt eingesehen werden:

- | | |
|---|----------------|
| 1. Health and Fitness und Science | Anhang B und C |
| 2. Health and Fitness und Business and Industrial | Anhang D und E |
| 3. Science und Business and Industrial | Anhang F und O |
| 4. Alle drei Kategorien | Anhang H |

Für die Erstellung dieser Visualisierungen wurden die jeweiligen Hauptkategorien herausgefiltert und durch Anwendung des ForceAtlas2 Algorithmus dargestellt. Des Weiteren erfolgte eine automatische Erweiterung der Knotenabstände um zu verhindern, dass die Knotenbeschriftungen sich überschneiden. Die in den Anhängen B, D und F dargestellten Graphen zeigen alle verbundenen Kategorien- und Schlüsselwörterknoten der jeweiligen Hauptkategorie an. Infolge dessen, dass bereits bei der Graphenerstellung alle Knoten die nur eine Verbindung haben herausgefiltert wurden, sind nur Knoten mit mindestens zwei Verbindungen sichtbar.

Die in den Anhängen C, E, O und H dargestellten Graphen wurden vorab so gefiltert, dass nur die Kategorien- und Schlüsselwörterknoten der jeweiligen Hauptkategorie angezeigt werden, die drei oder mehr Verbindungen aufweisen. Durch diese Vorgehensweise fällt eine große Anzahl an Knoten weg, so dass die Ansicht etwas übersichtlicher dargestellt wird. Hierbei fällt auf, dass vereinzelt Knotenpunkte mit nur einer oder zwei Verbindungen dargestellt werden, was darin begründet liegt, dass diese Knotenpunkte weitere Verbindungen zu herausgefilterten Kategorien oder Schlüsselwörtern aufweisen. Die Stärke der Verbindung wird durch die Dicke der Kante dargestellt. Umso dicker die Kante, desto öfter wurde vorab die Kategorie/Schlüsselwort Kombination gefunden. Die Kategorie Stränge sind jedoch alle mit der höchsten Verbindungsstärke verbunden, um diese von den Schlüsselwörtern unterschieden zu können. Die Darstellung in Anhang H zeigt die Verbindung aller drei Kategorien. Dabei wurden diese vorab gefiltert, so dass nur die Kategorien- und Schlüsselwörterknoten der jeweiligen Hauptkategorie angezeigt werden, die mehr als drei Verbindungen aufweisen.

7 Science Mapping

In diesem Kapitel wird ein Überblick über den interdisziplinären Bereich des Science Mapping vermittelt. Dieser runden die Arbeit insofern ab, dass sich das in der praktischen Ausarbeitung vorgestellte Verfahren in gewisser Weise in den Bereich des Science Mapping einteilen lässt. Dabei werden die einzelnen Aspekte sowie drei Projekte aus dem Bereich vorgestellt.

Um die Gesamtheit von wissenschaftlichen Feldern zu überblicken ist es, im traditionellen Sinne, die Aufgabe von Wissenschaftlern, Zusammenfassungen zu einem Feld anhand von Literatur zu erstellen. Dies ist eine sehr zeitaufwendige und komplexe Aufgabe, die aufgrund des aktuellen Informationswachstums nur schwer auf dem aktuellen Stand gehalten werden kann. Hinzu kommt die Herausforderung von multidisziplinären Forschungsfeldern, welche von Wissenschaftlern mit dem Fokus auf eine bestimmte Disziplin nicht allumfassend überblickt werden können (Börner, Chen und Boyack 2003, S. 180-181). Konsequenterweise ist das Interesse an der Analyse und an Visualisierungstechniken zur Entdeckung und Darstellung von wissenschaftlichen Feldern und ihren Zusammenhängen in den letzten Jahren stark angewachsen (Chen, Paul und O'Keefe 2001, S.315).

Dieses Kapitel ist in drei Teile gegliedert. Im ersten Teil wird auf die Definition von Science Mapping eingegangen. Im Anschluss werden verschiedene Aspekte beleuchtet, welche für die Betrachtung von Science Mapping wichtig sind. Zum Schluss werden einige Projekte vorgestellt, die sich mit dem Thema beschäftigen.

7.1 Definition

Chaomei Chen (2017, S. 3) definiert Science Mapping als einen generischen Prozess der Analyse und Visualisierung von Domänen, wobei der Fokus auf wissenschaftlichen Disziplinen, Forschungsgebieten oder Themenbereichen zu spezifischen Forschungsfragen liegt. Gemäß Henry Small (1999, S. 800) stellt Science Mapping eine räumliche Beziehung zwischen wissenschaftlichen Fachgebieten, Disziplinen, Feldern, Dokumenten oder Autoren dar, deren kognitive Strukturen und Entwicklungen untersucht werden.

Chen, Dubin und Schultz (2014, S. 1) definieren Science Mapping wie folgt etwas ausführlicher:

„Science mapping is a fast growing interdisciplinary field originated in information science and technology. Science mapping is the development and application of computational techniques to the visualization, analysis, and modeling of a broad range of scientific and technological activities as a whole. This is an interdisciplinary field emerging from traditional library information science in the areas of scientometrics, citation analysis, and computer science in the areas of information visualization, visual analytics, data mining and knowledge discovery.“

7.2 Aspekte

In diesem Abschnitt werden einzelne Aspekte betrachtet, welche für Science Mapping relevant sind. Hierzu gehören die Datenquelle, Analyseeinheiten, Datenvorverarbeitung, Beziehungsmaße, Mapping-Prozesse und die Visualisierungstechniken (Cobo, et al. 2011, S. 1383).

7.2.1 Datenquellen

Gemäß Cobo, et al. (2011, S. 1383) werden für Science Mapping typischerweise wissenschaftliche Arbeiten und Patente genutzt. Diese Dokumente können für die meisten wissenschaftlichen Bereiche anhand von bibliografischen Onlinedatenbanken abgerufen

werden, wobei die Bekanntesten Web of Science, Scopus, Google Scholar und NLM's MEDLINE sind. Cobo, et al. benennen in ihrer Arbeit weitere Quellen, die in Tabelle 18 dargestellt werden.

Typ	Name der Online-Datenbank	Quelle
Wissenschaftliche Arbeiten	<i>Web of Science</i>	https://www.webofknowledge.com
	<i>Scopus</i>	https://www.scopus.com
	<i>Google Scholar</i>	https://scholar.google.com
	<i>NLM's MEDLINE</i>	https://www.ncbi.nlm.nih.gov/pubmed
	<i>arXiv</i>	https://arxiv.org
	<i>CiteSeerX</i>	https://citeseerk.ist.psu.edu
	<i>Digital Bibliography & Library Project</i>	https://dblp.uni-trier.de
	<i>SAO/NASA Astrophysics Data System</i>	https://adswww.harvard.edu
	<i>Science Direct</i>	https://www.sciencedirect.com
Patente	<i>United States Patent and Trademark Office</i>	https://www.uspto.gov

Tabelle 18 Bibliografische Online-Datenbanken. (Quelle: Eigene Tabelle in Anlehnung an Cobo, et al. (2011, S. 1383))

7.2.2 Analyseeinheiten

Gemäß Börner, Chen und Boyack (2003, S. 10) werden üblicherweise Journale, Dokumente, Autoren, beschreibenden Begriffe oder Wörter als die zu analysierenden Einheiten bei der Visualisierung von Wissensdomänen betrachtet, wobei die Auswahl davon abhängig ist, welche Erkenntnisse aus den Daten gewonnen werden sollen. Cobo, et al. (2011, S. 1384) nennen verschiedene Beziehungen, die zwischen Analyseeinheiten ermittelt werden können, wobei die bekanntesten in Tabelle 19 abgebildet sind.

Bibliometrische ² Technik	Analyseeinheiten	Art der Beziehungen
<i>Bibliografische Koppelung</i>	<i>Autoren</i>	<i>Gemeinsame Referenzen zwischen deren Werken</i>
	<i>Dokumente</i>	<i>Gemeinsame Referenzen zwischen Dokumenten</i>
	<i>Journale</i>	<i>Gemeinsame Referenzen zwischen Journals</i>
<i>Co-Autor</i>	<i>Name des Autors</i>	<i>Gemeinsames Auftreten von Autoren</i>
	<i>Zugehöriges Land</i>	<i>Gemeinsames Auftreten von Ländern</i>
	<i>Zugehöriges Institut</i>	<i>Gemeinsames Auftreten von Instituten</i>
<i>Co-Zitationen</i>	<i>Autor Referenzen</i>	<i>Co-Zitationen von Autoren</i>
	<i>Dokument Referenzen</i>	<i>Co-Zitationen von Dokumenten</i>
	<i>Journal Referenzen</i>	<i>Co-Zitationen von Journals</i>
<i>Co-Wörter</i>	<i>Schlüsselwörter oder Begriffe welche aus dem Dokument extrahiert wurden</i>	<i>Gemeinsames Auftreten von Begriffen oder Schlüsselwörtern</i>

Tabelle 19 Bibliometrische Techniken. (Quelle: Eigene Tabelle in Anlehnung an Cobo, et al. (2011, S. 1384))

Bei der Betrachtung der einzelnen Analyseeinheiten und deren Beziehungen lassen sich verschiedene Aspekte eines Forschungsfeldes analysieren (Cobo, et al. 2011, S. 1384).

- **Autoren**

Gemäß Börner, Chen und Boyack (2003, S. 10) lässt sich bei der Betrachtung der Visualisierung von Autoren-Zitationen auf die intellektuelle Struktur eines wissenschaftlichen Feldes schliessen. Glänzel (2001) hat in seiner Arbeit gezeigt, dass durch die Visualisierung von gemeinsam in Dokumenten auftretenden Autoren eine Analyse von sozialen Strukturen in

² Alan Pritchard (1969, S. 349) definiert Bibliometrie als eine Anwendung von mathematischen und statistischen Methoden an Büchern oder anderen Medien.

wissenschaftlichen Feldern möglich ist. Des Weiteren kann die internationale Dimension eines Feldes anhand der Herkunft oder des Institutes der Autoren dargestellt werden (Cobo, et al. 2011, S. 1384). Mit der bibliografischen Koppelung von Autoren werden Beziehungen zwischen Autoren untersucht, welche die selben Werke zitieren (Zhao und Strotmann 2008).

- **Wörter**

Anhand von aus Schlüsselwörtern generierten semantischen Karten, lässt sich das Hauptkonzept sowie die konzeptionelle Struktur eines wissenschaftlichen Feldes analysieren (Börner, Chen und Boyack 2003, S. 10).

- **Dokumente / Journale**

Gemäß Börner, Chen und Boyack (2003, S. 10) handelt es sich bei Dokumenten, wie zum Beispiel Patente oder wissenschaftliche Arbeiten, um die am meisten genutzten Analyseeinheiten zur Visualisierung von Wissensdomänen. Dabei wird anhand ihrer bibliografischen Koppelung ermittelt, welche Dokumente gemeinsam auf ein anderes Dokument referenzieren (Gao und Guan 2009). Des Weiteren lassen sich die Beziehungen zwischen Dokumenten anhand ihrer Zitationen darstellen und analysieren (Cobo, et al. 2011, S. 1384).

7.2.3 Datenvorverarbeitung

Die Datenvorverarbeitung wird gemäß Cobe, et al. (2011, S. 1384 - 1385) wie folgt zusammengefasst.

Um die Beziehungen zwischen den Analyseeinheiten zu ermitteln, müssen vorab die dafür benötigten Informationen aus den bibliographischen Datenquellen mittels eines Datenvorverarbeitungsprozesses extrahiert werden. Um aus den Daten aussagekräftige Ergebnisse zu erzielen können dafür verschiedenen Schritte, wie beispielsweise die Korrektur von falsch oder unterschiedlich geschriebenen Wörtern, durchgeführt werden. Des Weiteren können bei der Datenvorverarbeitung bereits unwichtige Dokumente, aufgrund von beispielsweise fehlenden Beziehungen oder einer zu geringen Anzahl an Beziehungen zu anderen, aussortiert werden.

7.2.4 Beziehungsmaße

Um die Beziehungen zwischen Analyseeinheiten zu messen, benennen White und McCain (1997) für die in Tabelle 19 beschriebenen bibliomantischen Techniken folgende Herangehensweisen:

- **Bibliografische Koppelung**

Die Beziehungen zwischen Analyseeinheiten von bibliografischen Kopplungen lassen sich anhand der Anzahl an Indikatoren, welche in beiden Einheiten auftreten, messen.

- **Co-Zitation**

Die Co-Zitation zwischen Analyseeinheiten wird durch die Anzahl mit der zwei Dokumente von anderen Dokumenten zusammen zitiert werden, gemessen. Es müssen also mindestens zwei Dokumente gemeinsam von einem anderen Dokument zitiert werden, damit diese als co-zitiert angesehen werden.

- **Co-Wörter**

Dabei wird gemessen, wie oft ein Wort jeweils beispielsweise im Titel oder der Zusammenfassung von zwei Dokumenten vorkommt.

- **Co-Autor**

Diese Messung erfasst die Anzahl, des gemeinsamen Auftretens zweier Einheiten in einem Dokument.

wobei die Stärke der Beziehung anhand der Größe des gemessenen Wertes repräsentiert wird.

7.2.5 Mapping-Prozess

Gemäß Cobe, et al. (2011, S. 1385) wird beim Mapping-Prozess ein Algorithmus angewandt, welcher anhand der Analyseeinheiten sowie der ermittelten Beziehungsmaße eine Kartenvisualisierung erstellt. Laut Börner, Chen und Boyack (2003, S. 13-20) muss dafür vorab, ähnlich wie in Unterabschnitt 4.3.3 beschrieben, ein Vektorraummodell für jedes Dokument erstellt werden, wobei die aus den Inhalten extrahierten Wörter die Dimensionen darstellen. Die Gewichtung für jede Dimension pro Dokument wird häufig durch TF-IDF berechnet. Im Anschluss werden mithilfe einer Dimensionsreduzierungstechnik die Dimensionen des Vektorraummodells reduziert. Hierfür erläutert Börner, Chen und Boyack Techniken wie beispielsweise das multidimensionale Scaling, eine Eigenvalue/Eigenvektor decomposition oder eine Factor Analyse. Daraufhin kann gemäß Cobe, et al. (2011, S. 1385) ein Cluster Algorithmus angewandt werden, um einzelne Themengemeinschaften zu erkennen und das globale Netzwerk in kleine Subnetzwerke aufzuteilen.

7.2.6 Visualisierungstechniken

Infolge der Anwendung der richtigen Visualisierungsart soll der Nutzer in der Lage sein, einen Überblick über allgemeine Muster und Trends zu erlangen und die dargestellten Datenobjekte leicht interpretieren zu können (Börner, Chen und Boyack 2003, S. 23).

Chaomei Chen (2017, S. 3) nennt in seiner Arbeit eine Vielzahl von angewandten Visualisierungsarten wie beispielsweise Graphen- und Netzwerkvisualisierungen (Herman, Melançon und Marshall 2000), Hierarchie- und Baumvisualisierungen (Johnson und Schneiderman 1991), Visualisierungen von zeitlichen Strukturen (Morris, et al. 2003) und raumbezogene Visualisierungen.

- **Netzwerkvisualisierungen**

Die Netzwerkvisualisierung stellt die Beziehung zwischen Analyseeinheiten mit Knoten und Kanten dar. Dabei repräsentieren die Knoten die Analyseeinheiten und die Kanten die Beziehungen (Cobo, et al. 2011, S. 1384). Die Länge der Kanten kann die Stärke der Beziehung zwischen den verknüpften Knoten widerstrengeln. Im Regelfall wird davon ausgegangen, dass eine kurze Distanz eine stärkere Beziehung verdeutlicht (Van Eck und Waltman 2010, S. 525). Des Weiteren nennen Cobo, et al. (2011, S. 1386) Visualisierungen wie heliozentrische Karten (De Moya-Anegón, et al. 2005), geometrische Modelle (Skupin 2009) und Thematische Netzwerke (Bailón-Moreno und Jurado-Alameda 2006), welche sich für die Darstellung von Netzwerken und Subnetzwerken eignen.

- **Visualisierung von Zeitlichen Strukturen**

Anhand der Visualisierung zeitlicher Strukturen kann die Evolution von wissenschaftlichen Feldern dargestellt werden. Dafür werden beispielsweise anhand von in den Quellen verfügbaren Veröffentlichungsdaten, Cluster von aufeinanderfolgenden Zeiträumen gebildet (Zhang, Liu und Zhao 2008, S. 10). Hierfür können gemäß Cobo, et al. (2011, S. 1386) verschiedene Techniken, wie beispielsweise Cluster-String (Small 2006), Rolling-Clustering (Kandylas, Upham und Ungar 2010), Alluvial Diagramme (Rosvall und Bergstrom 2010) und

ThemeRiver Visualisation (Havre, et al. 2002) angewandt werden. Durch andere Herangehensweisen wird einfach nur der zu betrachtende Zeitraum, in Abhängigkeit der Vorherigen und Folgenden, (Leydesdorff und Schank 2008) oder zeitliche Veränderungen auf einem Graphen (C. Chen 2004) dargestellt.

- **Raumbezogene Visualisierung**

Gemäß Cobo, et al. (2011, S. 1386) werden raumbezogene Daten für gewöhnlich mittels einer Weltkarte oder einer thematischen Karte visualisiert. Beispielsweise lassen sich dadurch Beziehungen von Autoren auf einer Weltkarte anhand ihres Landes oder ihrer Stadt darstellen (Nagel, Duval und Heidmann 2011) oder die Beziehungen von Instituten, aus welchen gemeinsame Veröffentlichungen stammen (Yao, et al. 2017).

- **Baumvisualisierungen**

Baumvisualisierungen werden üblicherweise für die Darstellung von Hierarchien genutzt. Bei der Baumvisualisierung wird ein Knoten in der Regel als Rechteck dargestellt. Im Falle, dass einem Knoten andere Knoten untergestellt sind, werden diese kleiner dargestellt und mit ihrem überliegenden Knoten mithilfe von Kanten verbunden. So wird eine Hierarchie im ursprünglichen Sinne als gerichteter Baum dargestellt, wobei sich der Baum von oben nach unten ausbreitet und die Elternteile auf ihre Kinder zeigen (Long, et al. 2017, S. 108-109).

7.3 Projekte

Es existiert eine Vielzahl von Projekten im Bereich des Science Mapping. Diese reichen von diversen Softwarelösungen (Cobo, et al. 2011) über Onlineprojekte (Open Knowledge Maps - Verein zur Förderung der Sichtbarkeit wissenschaftlichen Wissens, Open Knowledge Maps) (Moritz Stefaner and Studio NAND, Mapping Scientific Excellence) bis hin zu Ausstellungen (Cyberinfrastructure for Network Science Center, Places & Spaces: Mapping Science). In diesem Abschnitt werden die Projekte Paperscape, VOSviewer und CiteSpace kurz vorgestellt.

7.3.1 Paperscape

Gemäß den Angaben der Webseite von George und Knegjens (About Paperscape) handelt es sich bei Paperscape um ein Onlineprojekt zur Visualisierung aller wissenschaftlichen Forschungsarbeiten aus der arXiv Datenbank. Dabei beinhaltet arXiv rund 1,5 Millionen Arbeiten aus den Bereichen Physik, Mathematik, Informatik, nichtlineare Wissenschaften, quantitative Biologie, quantitative Finanzen, Statistik, Elektronik und Systemwissenschaft sowie Wirtschaft (Cornell University, General Information About arXiv).

Jede wissenschaftliche Forschungsarbeit wird, wie in Abbildung 28 sichtbar, als Kreis auf einer Karte dargestellt, wobei ihre Position basierend auf den Referenzen zwischen den Arbeiten bestimmt wird. Die Größe des Kreisumfangs ist proportional zur Anzahl der auf die Arbeit gesetzten Zitationen. Dabei werden die Referenzen und Zitatanzahlen täglich automatisch aus den TeX/LaTeX und PDF Dateien extrahiert. Die Kreise sind, in Abhängigkeit des Forschungsbereiches, farbig dargestellt, wobei der Farbton mit steigendem Alter der Arbeit dunkler wird. Des Weiteren sind alle Kreise mit einem Label markiert, welches, durch eine Worthäufigkeitsanalyse aus dem Abstrakt und der Überschrift, automatisiert extrahiert wird (George und Knegjens, About Paperscape).

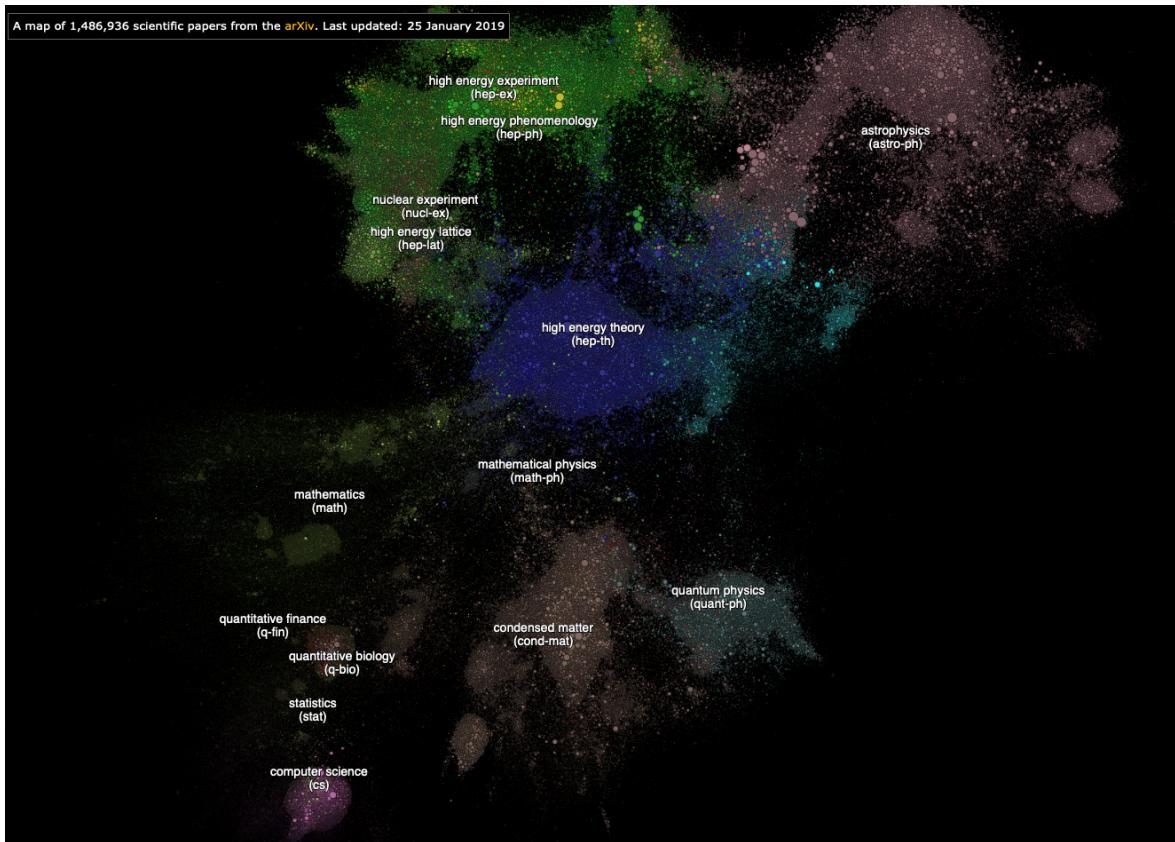


Abbildung 28 Paperscape Kartenausschnitt. (Quelle: Ansicht der Webseite von George und Knegjens (Paperscape 2019))

7.3.2 VOSviewer

Gemäß Van Eck und Waltman (2010, S. 523) handelt es sich bei VOSviewer um ein Computerprogramm, mit dessen Hilfe bibliometrische Karten erstellt und visualisiert werden.

Diese Karten können Netzwerke von wissenschaftlichen Veröffentlichungen, wissenschaftlichen Zeitschriften, Forschern, Forschungsorganisationen, Ländern, Schlüsselwörtern oder Begriffen abbilden, wobei die einzelnen Netzwerkelemente durch gemeinsame Zitationen, Co-Autorschaften oder Vorkommen verbunden werden können. Die Stärke der Verbindung wächst hierbei beispielsweise mit der Anzahl an zitierten Verweisen die zwei Publikationen gemeinsam haben, der Anzahl an gemeinsam auftretenden Forschern oder der Anzahl an Veröffentlichungen in denen zwei Schlüsselwortphrasen gemeinsam vorkommen. Die Elemente können als Wörter oder Kreise dargestellt werden, wobei deren Größe von ihrer Wichtigkeit bzw. ihrer Anzahl an Verbindungen abhängig ist. Als Datenquellen können dafür bibliografische Datenbankdateien, beispielsweise von Web of Science (Clarivate Analytics, Web of Science) oder Scopus (Elsevier B.V., Scopus), sowie Referenzmanager Daten, beispielsweise von EndNote (Clarivate Analytics, EndNote) oder RefWorks (ProQuest LLC, RefWorks) genutzt werden (Van Eck und Waltman 2019, S. 3-5).

VOSviewer nutzt Text-Mining Methoden, um aus Daten der genannten Quellen Zitierbeziehungen zwischen Publikationen oder Zeitschriften, Kooperationsbeziehungen zwischen Forschern und Zusammenhänge zwischen wissenschaftlichen Begriffen zu erstellen. Beispielsweise werden die Zusammenhänge zwischen wissenschaftlichen Begriffen durch die Extraktion von Wörtern aus wissenschaftlichen Publikationen, Patenten oder Zeitungsartikeln gewonnen. Zu diesem Zweck werden Wortphrasen, die aus Substantiven und

Adjektiven bestehen und mit einem Substantiv enden, mittels einer Wortartenbestimmung aus den Texten herausgefiltert. Diese Phrasen werden dabei in die Singularform umgewandelt und im Anschluss die Wortphrasen mit der höchsten Relevanz ermittelt. Es wird für jede Phrase ermittelt, wie oft diese in dem Text vorkommt, wobei der daraus resultierende Wert mit dem gesamten Vorkommen aller Phrasen verglichen wird. Umso größer der Unterschied dieser beiden Werte ausfällt, desto höher wird die Relevanz der Phrase eingeschüttet. Phrasen mit einer hohen Relevanz werden daraufhin durch ein Clusterverfahren (Van Eck, Waltman und Dekker, et al. 2010, Waltman, Van Eck und Noyons 2010) weiterverarbeitet und zum Schluss visualisiert (Van Eck und Waltman 2011, S. 1-2).

Beispielhaft wird in Abbildung 29 eine Phrasenkarte dargestellt, welche anhand von Doktorarbeiten erzeugt wurde (Centre for Science and Technology Studies, Leiden University).

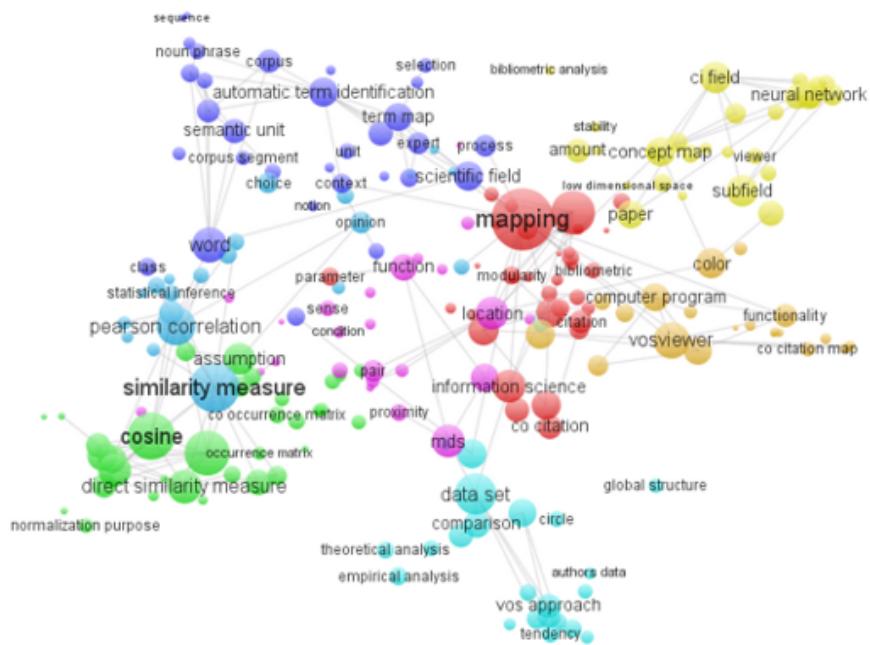


Abbildung 29 VOSviewer Darstellung. (Quelle: Van Eck und Waltman (2011, S. 5))

Gemäß Van Eck und Waltman (2019, S. 3) handelt es sich bei VOSviewer um eine mit der Programmiersprache Java erstellte Software, die plattformunabhängig ist und zur freien Nutzung zur Verfügung steht.

7.3.3 CiteSpace

Gemäß den Angaben der Webseite von Chaomei Chen (CiteSpace) handelt es sich bei CiteSpace um eine frei erhältliche Java-Anwendung, welche zur Visualisierung und Analyse von Trends und Mustern in wissenschaftlicher Literatur entwickelt wurde.

Als Hauptdatenquellen dienen die bibliografische Datenbankdateien der Onlineplattform Web of Science (Clarivate Analytics). Des Weiteren können Daten von Anbietern wie PubMed (National Center for Biotechnology Information), arXiv (Cornell University, arXiv.org) und NSF Award Abstracts (National Science Foundation) über eine einfache Schnittstelle abgerufen werden (C. Chen, CiteSpace). Gemäß Chaomei Chen (2006, S.365) nutzt CiteSpace Text-Mining Methoden, um beispielsweise die Überschriften, Zusammenfassungen und Zitatreferenzen aus den wissenschaftlichen Publikationen zu extrahieren. Des

Weiteren werden Forschungsbegriffe aus den Überschriften und Zusammenfassungen ermittelt. Dafür filtert CiteSpace N-Gramme oder Begriffe, die in Abhängigkeit der Anzahl des Auftretens oder der Wachstumsstärke extrahiert werden. Die Visualisierung erfolgt anhand einer Graphen-Netzwerk- und Zeitzonansicht, wobei Beziehungen zwischen wissenschaftlichen Publikationen durch Autorenkooperationen sowie Autoren- und Dokumentenzitationen dargestellt werden können (C. Chen, CiteSpace). Die Publikationen werden dabei wie in Abbildung 30 sichtbar als Kreis dargestellt. Die Ringe sind mit einem Label versehen, welches beispielsweise den Autor, das Erscheinungsjahr und ein Schlagwort der Publikation beinhaltet. Des Weiteren befindet sich eine Nummer im Inneren des Kreises, welche die gesamte Anzahl an Zitationen repräsentiert. Die Anzahl der Zitationen pro Zeitintervall wird mit farbigen Ringen im Kreis dargestellt, wobei die Dicke der Ringe abhängig von der Anzahl an Zitationen in diesem Zeitraum ist. Dabei repräsentiert der innere Ring den ältesten und der äußere Ring den aktuellsten Zeitraum. Verbindungen zwischen Publikationen sind, in Abhängigkeit des Jahres der Zitation, genau wie die Ringe gemäß des Farb-Zeitintervalls eingefärbt (C. Chen 2006, S. 365).

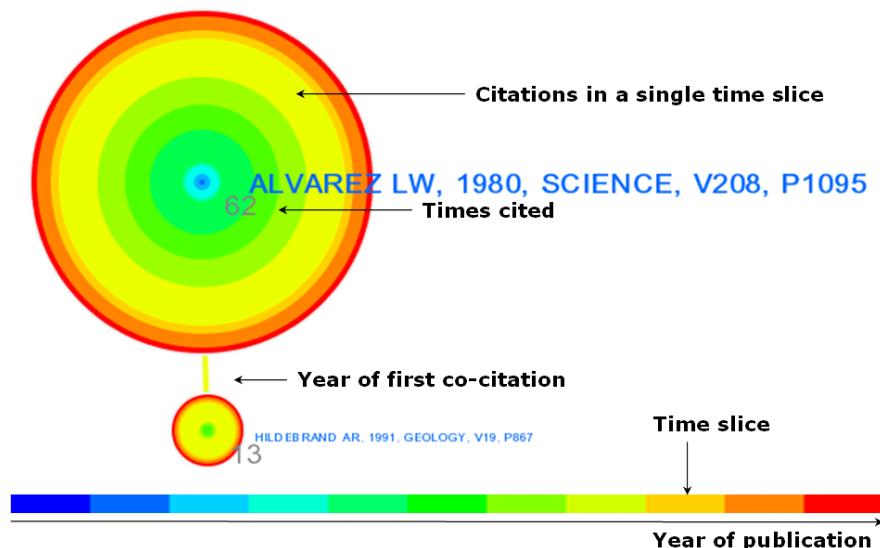


Abbildung 30 CiteSpace Darstellung einer Publikation. (Quelle: C. Chen (2006, S. 365))

CiteSpace ist speziell als Werkzeug für die Visualisierung von Wissensbereichen konzipiert und konzentriert sich dabei darauf, kritische Punkte in der Entwicklung und intellektuelle Wendepunkte eines wissenschaftlichen Feldes zu finden (C. Chen 2004). Dafür bietet CiteSpace eine Vielzahl an Funktionen, die das Verständnis und die Interpretation von Netzwerkmustern erleichtern sollen. Dazu gehören das Erkennen der Themenbereiche mit schnellem Wachstum, das Auffinden von Hotspots für Zitate im Land der Veröffentlichungen, das Zerlegen eines Netzwerks in Cluster sowie das automatische Kennzeichnen von Clustern mit Begriffen aus zitierenden Artikeln (C. Chen, CiteSpace).

8 Schlussbemerkungen und Ausblick

Diese Arbeit verfolgte das Ziel ein Verfahren zu entwickeln, welches anhand von zukünftigen Forschungsprojekten etwaig auftretende Schnittpunkte zwischen Forschungsbereichen mithilfe von Text-Mining Methoden sichtbar macht.

Zu diesem Zweck wurde einleitend der Begriff des Text-Mining definiert, wobei es sich dabei um ein Forschungsgebiet handelt, welches Techniken aus verschiedenen anderen Richtungen der Informatik nutzt, um Lösungen für das Problem der textuellen Informationsüberflutung zu entwickeln. Insgesamt sechs Anwendungsbereiche konnten aus den umliegenden und sich teils überschneidenden Forschungsbereichen des Text-Mining identifiziert und erläutert werden. Dabei haben sich die folgenden zwei Bereiche als besonders relevant für die Entwicklung des Zielverfahrens herauskristallisiert:

1. Verarbeitung von natürlicher Sprache
2. Textklassifikation

Aus Erkenntnissen des Anwendungsbereiches der Verarbeitung von natürlicher Sprache erfolgte eine Vorstellung von einzelnen Vorverarbeitungsschritten, die nahezu immer erforderlich sind, bevor Texte analysiert werden können. Ausgehend vom Vorgang der Trennung des Dokumenteninhaltes in einzelne Fragmente wurden dabei speziell die Stoppwortfilterungs-, Wortbestimmungs- und Stammformreduktionsprozesse erörtert.

Daraufhin wurde, auf der Grundlage des zweiten relevanten Anwendungsbereichs des Text-Mining, ein Überblick über die methodischen Ansätze der automatischen Textklassifikation vermittelt. Es erfolgte, neben statistischen Ansätzen, eine Vorstellung der Kategorisierungsmöglichkeiten von Texten durch das maschinelle Lernen. Für das Verständnis des für die automatische Textklassifizierung notwendigen Vorgehens wurden die einzelnen theoretischen Prozessschritte herausgearbeitet und zum Schluss mit einem Beispiel zusammengefasst.

Als ein weiterer relevanter Themenbereich wurde die automatische Schlüsselwort-Extraktion identifiziert, welche auf Methoden der Verarbeitung von natürlicher Sprache zurückgreift. Die dafür notwendigen theoretischen Prozessschritte ließen sich grob in die Bestimmung der Kandidaten und Extraktion der tatsächlichen Schlüsselwörter unterteilen. Letzterer beruht dabei auf überwachten oder unbeaufsichtigten Ansätzen, wobei beide mittels statistischer, sprachlicher und graphbasierender Techniken versuchen, relevante Schlüsselwörter aus den vorab durch Textvorverarbeitungsschritte bestimmten Kandidaten zu extrahieren. Eigens für den überwachten Ansatz werden zusätzlich Techniken des maschinellen Lernens angewandt, die aufbauend auf einem aus einer Trainingsphase mit Beispieldaten entstandenen Modell die Extraktion der Schlüsselwörter durchführen. Weiterhin wurde eine Evaluationsstrategie definiert, durch die als Erweiterung des Prozesses die Effektivität von angewandten Verfahren gemessen wird. Um die unterschiedlichen Herangehensweisen von auf den vorab beschriebenen Techniken basierenden Verfahren zu verdeutlichen, erfolgte eine theoretische Ausarbeitung der drei Strategien YAKE!, KEA und PositionRank anhand der dafür zu Grunde liegenden wissenschaftlichen Publikationen.

Mittels des im praktischen Teil dieser Arbeit vorgestellte Verfahren wird ein Ansatz erläutert, der es ermöglicht, unter Einsatz von Text-Mining Methoden und Visualisierungstechniken aus durch CORDIS veröffentlichten Projekt-Zielbeschreibungen Verbindungen anhand von extrahierten Schlüsselwörtern zwischen Forschungsbereichen darzustellen. Hierfür wurde eine Dokumentenkollektion bestehend aus 979 CORDIS Projektdatensätzen mit einem

Projektstart ab dem 01.01.2019 erstellt und die zur weiteren Bearbeitung relevanten Informationen extrahiert. Daraufhin erfolgte eine Klassifizierung der Projekte, wofür auf eine Klassifizierungslösung von IBM Watson zurückgegriffen wurde. Mit dieser war es möglich, anhand der jeweiligen Projektzielbeschreibungen und Titel alle Dokumente in insgesamt 15 der 23 zur Verfügung stehenden Hauptkategorien einzuteilen, die sich in eine granulare Tiefe von bis zu fünf Ebenen untergliedern. Aufgrund von fehlenden Testdokumenten, sollte bei diesem Schritt jedoch die nicht vorhandene Möglichkeit der Evaluation des Ergebnisses kritisch betrachtet werden. Infolge dessen kann die Korrektheit der durch die Klassifikationslösung bestimmten Kategorien nicht validiert werden.

Weiterhin erfolgte die Implementierung des Schlüsselwort-Extraktionsprozesses. Dabei wurde auf bestehende Python Bibliotheken zurückgegriffen, welche die drei Schlüsselwort-Extraktionsverfahren YAKE!, KEA und PositionRank, auf der Grundlage der im theoretischen Teil erörterten Funktionsweise, realisieren. Als besonders herausfordernd bei der Ermittlung von Schlüsselwörtern stellten sich in diesem Prozessschritt die Unterschiede in der Art der Projektzielbeschreibung dar. Manche Texte leiten die Zielsetzung mit längerer Vorstellung des Themas ein, andere benennen direkt in der Einleitung das Ziel des Projektes. Um den Schlüsselwort-Extraktionsprozess einfach und übersichtlich zu gestalten, wurde eine Schlüsselwort-Extraktorklasse entwickelt, welche die drei genannten Verfahren sowie eine Methode für die Stammformreduktion beinhaltet. Anhand von selbstdefinierten Trainingsdokumenten wurde im Selbstversuch eine Evaluation von YAKE!, KEA und PositionRank durchgeführt. Dafür erfolgte vorab eine manuelle Bestimmung von Schlüsselwörtern für insgesamt 90 CORDIS Projekte, wovon 50 für die Erstellung des KEA Trainingsmodells und 40 für die Evaluation genutzt wurden. Anhand der daraus resultierenden Evaluationsergebnisse wurde PositionRank, mit einer Anzahl von 10 extrahierten Schlüsselwörtern pro Dokument, als das beste Verfahren mit einem F-Score Ergebnis von 28,67 % identifiziert. Als kritisch müssen hierbei die im Selbstversuch erarbeiteten Testdokumente betrachtet werden, da dieser Vorgang eine Validierung von Experten erfordert um eine Korrektheit der Ergebnisse zu garantieren.

Als nächster Schritt erfolgte der Aufbau des Graphen, der für die Visualisierung des Resultates der vorab beschriebenen Schritte notwendig ist. Auch hierfür wurde eine eigene Klasse entwickelt, um den Graph-Aufbauprozess einfacher und übersichtlicher zu gestalten. Mithilfe dieser Klasse wurden die Schlüsselwörter und Kategorien dem Graphen als Knotenpunkte zugewiesen. Deren Beziehungen wurden mit Kanten bestimmt, wobei diese die Attribute der Verbindungsstärke sowie der Bezeichnung der ausgehenden Hauptkategorie der verbundenen Elemente besitzen. Das Ergebnis dieses Prozessschrittes ist eine Datei im GML Format, mit welcher eine Visualisierung des Graphen in einer Netzansicht möglich ist. Hierfür wurde die Open-Source Software Gephi genutzt. Die Netzdarstellung zeigt ein Geflecht an Verbindungen zwischen den 15 Hauptkategorien, deren Unterebenen sowie den extrahierten Schlüsselwörtern. Dabei werden die Kategorien als Hauptstränge mit der höchsten Kantengewichtung dargestellt. Die Dicke der Verbindungen zwischen den Kategorie- und Schlüsselwortknoten signalisiert die Häufigkeit des gemeinsamen Auftretens in den Projektdokumenten. Des Weiteren ermöglicht Gephi, die drei am Häufigsten verkommenden Kategoriebäume herauszufiltern, anhand welcher grafischen Gegenüberstellungen erstellt wurden. Diese Ergebnisse können im Anhang B, C, D, E, F, G und H betrachtet werden.

Als Schlussfolgerung des entwickelten Verfahrens lässt sich in Hinblick auf die eingangs formulierten Forschungsfragen zusammenfassen, dass es mithilfe von Text-Mining Methoden möglich ist Schnittstellen zwischen wissenschaftlichen Bereichen anhand von Zielbeschreibungen zukünftiger Forschungsprojekte zu identifizieren. Das dafür entwickelte

Verfahren beschreibt einen Ansatz, durch den Informationen aus den Texten selektiert, extrahiert, klassifiziert und visualisiert werden können.

Ausblick

Trotzdem bereits ein Ergebnis mit Verbindungen zwischen den verschiedenen wissenschaftlichen Bereichen dargestellt werden kann, ist es schwierig für den Betrachter deren zukünftige Konvergenzen zu erkennen. Daher erfordert es Expertenwissen, um tatsächlich feststellen zu können, welche der dargestellten Schlüsselwort-Verbindungen relevant für die zukünftige Entwicklung der wissenschaftlichen Domäne sind.

Grundlegend könnte außerdem das Potenzial des Verfahrens mittels Weiterentwicklungen gesteigert werden. Beispielsweise wäre es interessant, Trainingsdokumente anhand von Expertenwissen zu erstellen, um eine verbesserte Validierung der Klassifikation und Schlüsselwort-Extraktion durchzuführen. Darauf aufbauend könnten weitere Techniken für diese beiden Prozesse untersucht werden, die gegebenenfalls genauere Resultate erzielen. Wie in Abschnitt 5.4 aufgelistet, existiert dafür bereits eine große Auswahl an verschiedenen Schlüsselwort-Extraktionsverfahren, welche in Betracht gezogen werden können. Zusätzlich ist natürlich auch die Entwicklung einer neuen Technik möglich, die speziell auf den Aufbau der Projektbeschreibungen angepasst wird. Weiterhin ist eine detailliertere Definition der Kategorien für die Klassifizierung denkbar. Als Folge dessen lassen sich die einzelnen Forschungsbereiche gegebenenfalls noch spezifischer untersuchen. Außerdem bietet das Clustering im Gegensatz zur Klassifizierung eine Möglichkeit der Identifikation von Wissensdomänen. Auch dieser Ansatz sollte zukünftig betrachtet werden, da hierdurch eine Abkapselung von bereits bestehenden festen Strukturen entstehen kann, ohne auf vorab bestimmte Kategoriennamen angewiesen zu sein.

Neben den Verbesserungen des Verfahrens wäre es auch interessant, weitere Datenquellen zu betrachten. Beispielsweise könnten vor dem 01.01.2019 beginnende Projekte betrachtet werden, die dennoch ein Enddatum in der Zukunft aufweisen. Diese beinhalten gegebenenfalls bereits Zwischenberichte, welche viel detailliertere Informationen über die einzelnen Projekte beinhalten. Da das aktuelle Förderprogramm der EU-Kommission Horizon 2020 bis zu dessen Ablauf im kommenden Jahr nur noch verhältnismäßig wenige neue Projekte beinhaltet, sollte außerdem eine Untersuchung des darauffolgenden Programms durchgeführt werden. Auch eine Recherche und anschließende Betrachtung anderer Projektquellen ist denkbar.

Weiterhin müsste eine Auseinandersetzung mit weiteren Visualisierungsarten erfolgen, um etwaige Konvergenzen zwischen den Forschungsbereichen aufzudecken. Wie im Unterabschnitt 7.2.6 herausgearbeitet, existiert hierfür bereits eine Vielzahl an Visualisierungstechniken. Zusätzlich könnten die Informationen zu den Projekten, wie beispielsweise die Projekt-ID, in die Visualisierung eingearbeitet werden, wodurch der Betrachter die Möglichkeit hat, genauere Informationen über die Herkunft der einzelnen Verbindungen zu erhalten. Ein denkbare Darstellungsbeispiel bietet die genutzte Visualisierungsart von dem im Unterabschnitt 7.3.3 vorgestellten Projekt CiteSpace.

Zusammenfassend lässt sich feststellen, dass aus dem in dieser Arbeit vorgestellten Verfahren auch zukünftig weitere Forschungsfragen hergeleitet und untersucht werden können.

Literaturverzeichnis

- Aggarwal, Charu C., und Cheng Xiang Zhai. 2012. „A Survey of Text Classification Algorithms.“ In *MINING TEXT DATA*, 163 - 223. Boston/Dordrecht/London: Kluwer Academic Publishers.
- Allahyari, Mehdi, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, und Krys Kochut. 2017. „A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques.“ *arXiv:1707.02919*. 1-13.
- Altinel, Berna, und Murat Can Ganiz. 2016. „A new hybrid semi-supervised algorithm for text classification with class-based semantics.“ *Knowledge-Based Systems Volume 108*. 50-64.
- Azam, Nouman, und JingTao Yao. 2012. „Comparison of term frequency and document frequency based feature selection metrics in text categorization.“ *Expert Systems with Applications Volume 39, Issue 5*. 4760-4768 .
- Börner, Katy, Chaomei Chen, und Kevin W. Boyack. 2003. „Visualizing Knowledge Domains.“ *Annual Review of Information Science & Technology, Volume 37*. Medford, NJ. 179-255.
- Bailón-Moreno, Rafael, und Encarnación Jurado-Alameda. 2006. „The scientific network of surfactants: Structural analysis.“ *Journal of the American Society for Information Science and Technology*, 57 (7). 949–960.
- Barker, Ken, und Nadia Cornacchia. 2000. „Using noun phrase heads to extract document keyphrases.“ In *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*. 40–52.
- Barzilay, R., K. R. McKeown, und M. Elhadad. 1999. „Information fusion in the context of multi-document summarization.“ *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. ACL. 550-557.
- Barzilay, Regina, und Michael Elhadad. 1997. „Using lexical chains for text summarization.“ In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*. 10-17.
- Bastian, Mathieu, Sébastien Heymann, und Mathieu Jacomy. 2009. „Gephi: An Open Source Software for Exploring and Manipulating Networks.“ *INTERNATIONAL AAAI CONFERENCE ON WEBLOGS AND SOCIAL MEDIA; THIRD INTERNATIONAL AAAI CONFERENCE ON WEBLOGS AND SOCIAL MEDIA*.
- Beliga, Slobodan. 2014. „Keyword extraction : A review of methods and approaches.“
- Beliga, Slobodan, Ana Meštrović, und Sanda Martincic-Ipsic. 2015. „An Overview of Graph-Based Keyword Extraction Methods and Approaches.“ *Journal of Information and Organizational Sciences*. 1-20.
- Bennani-Smires, Kamil , Claudiu Musat, Andreaa Hossmann, Michael Baeriswyl, und Martin Jaggi. 2018. „Simple Unsupervised Keyphrase Extraction using Sentence Embeddings.“
- Berend, Gábor. 2011. „Opinion expression mining by exploiting keyphrase extraction.“ *Proceedings of the 5th International Joint Conference on Natural Language Processing*. 1162–1170.
- Bharti, Santosh Kumar, Korra Sathya Babu, und Sanjay Kumar Jena. 2017. „Automatic Keyword Extraction for Text Summarization: A Survey.“
- Boudin, Florian. kein Datum. *Document frequency counts* . Zugriff am 28. 01 2019. <https://boudinfl.github.io/pke/build/html/tutorials/df.html>.
- . 2018b. *GitHub PKE*. 06. 12. Zugriff am 28. 01 2019. <https://github.com/boudinfl/pke>.
- . kein Datum. *pke 1.8 documentation*. Zugriff am 28. 01 2019. <https://boudinfl.github.io/pke/build/html/index.html>.

- . 2016. „pke: an open source python-based keyphrase extraction toolkit.“ *COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*. Osaka, Japan. 69–73.
- . kein Datum. *PositionRank*. Zugriff am 28. 01 2019. <https://boudinfl.github.io/pke/build/html/unsupervised.html#positionrank>.
- . kein Datum. *Training supervised models*. Zugriff am 28. 01 2019. <https://boudinfl.github.io/pke/build/html/tutorials/training.html>.
- Boudin, Florian. 2018. „Unsupervised Keyphrase Extraction with Multipartite Graphs.“
- Bougouin, Adrien, Florian Boudin, und Béatrice Daille. 2013. „TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction.“ *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Nagoya, Japan. 543–551 .
- Brin, Sergey, und Lawrence Page. 1998. „The anatomy of a large-scale hypertextual Web search engine.“ *Computer Networks and ISDN Systems* 30. Stanford CA USA. 107–117.
- Campos, Ricardo, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, und Adam Jatowt. 2018a. „A Text Feature Based Automatic Keyword Extraction Method for Single Documents.“ *European Conference on Information Retrieval*. Grenoble, France. 684–691.
- . 2018c. *pipy.org yake 0.3.7*. 16. 11. Zugriff am 28. 01 2019. <https://pypi.org/project/yake/>.
- . 2018b. „YAKE! Collection-independent Automatic Keyword Extractor.“ *Proceedings of the 40th European Conference on Information Retrieval (ECIR'18)*. Grenoble, France. 806–810.
- . 2018d. *YAKE! StopwordsList*. 09. 11. Zugriff am 28. 01 2019. https://github.com/LIAAD/yake/blob/master/yake/StopwordsList/stopwords_en.txt.
- Centre for Science and Technology Studies, Leiden University. kein Datum. *Text mining and visualization using VOSviewer*. Zugriff am 28. 01 2019. <http://www.vosviewer.com/text-mining-and-visualization-using-vosviewer#JASIST>.
- Chen, Chaomei. kein Datum. *CiteSpace*. Zugriff am 28. 01 2019. <http://cluster.cis.drexel.edu/~cchen/citespace/>.
- . 2006. „CiteSpace II: Detecting and Visualizing Emerging Trends and Transient Patterns in Scientific Literature.“ *Journal of the American Society for Information Science and Technology* 57(3). 359–377.
- Chen, Chaomei. 2017. „Science Mapping: A Systematic Review of the Literature. Vol. 2 No. 2.“ *DE Gruyter* 1–40.
- . 2004. „Searching for intellectual turning points: Progressive Knowledge Domain Visualization.“ *Proc. Natl. Acad. Sci. USA*, 10. 5303–5310.
- Chen, Chaomei, Rachael Dubin, und Timothy Schultz. 2014. „Science Mapping.“ *Encyclopedia of Information Science and Technology, Third Edition*. . IGI Global.
- Chen, Chaomei, Ray J. Paul, und Bob O’Keefe. 2001. „Fitting the Jigsaw of Citation: Information Visualization in Domain Analysis.“ *Journal of the American Society for Information Science and Technology*, v52 n4. 315–330.
- Chen, Ping-I, und Shi-Jen Lin. 2010. „Automatic keyword prediction using Google similarity distance.“ *Expert Systems with Applications Volume 37, Issue 3*. 1928–1938 .
- Chien, L.-F. . 1997. „Pat-tree-based keyword extraction for Chinese information retrieval.“ *ACM SIGIR Forum*, Vol. 31, ACM. 50–58.
- Clarivate Analytics. kein Datum. *EndNote*. Zugriff am 28. 01 2019. <https://endnote.com>.
- . kein Datum. *Web of Science*. Zugriff am 28. 1 2019. www.webofknowledge.com.
- Cobo, Manuel J., Enrique Herrera-Viedma, A. G. López-Herrera, und Francisco Herrera. 2011. „Science Mapping Software Tools: Review, Analysis, and Cooperative Study Among Tools.“ *Journal of the American Society for Information Science and Technology* . 1382–1402.
- Cohen, J. D. 1995. „Highlights: Language- and domain-independent automatic indexing terms for abstracting.“ *JASIS*, vol. 46 (3). 162–174.

- Conroy, J. M., und D. P. O'leary. 2001. „Text summarization via hidden Markov models.“ *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM. 406-407.
- Cornell University. kein Datum. arXiv.org. Zugriff am 28. 1 2019. <https://arxiv.org/>.
- . kein Datum. *General Information About arXiv*. Zugriff am 28. 01 2019. <https://arxiv.org/help/general>.
- Cyberinfrastructure for Network Science Center. kein Datum. *Places & Spaces: Mapping Science*. Zugriff am 28. 01 2019. <http://scimaps.org>.
- De Fauw, Jeffrey, Joseph R. Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, et al. 2018. „Clinically applicable deep learning for diagnosis and referral in retinal disease.“ *Nature Medicine* volume 24. 1342–1350.
- De Moya-Anegón, Félix, Benjamín Vargas-Quesada, Zaida Chinchilla-Rodriguez, Elena Corera-Álvarez, Víctor Herrero-Solana, und Francisco J. Muñoz-Fernández. 2005. „Domain analysis and information retrieval through the construction of heliocentric maps based on ISI-JCR category cocitation.“ *Information Processing & Management*, 41 (6). 1520–1533.
- Dennis, S. F. 1967. „The design and testing of a fully automatic indexing searching system for documents consisting of expository text.“ *Information Retrieval: a Critical Review*. Washington DC: Thompson Book Company. 67-94.
- Dorogovtsev, S. N., A. V. Goltsev, und J. F. F. Mendes. 2008. „Critical phenomena in complex networks.“ *Reviews of Modern Physics* 80. 1275.
- Dredze, Mark, Hanna M. Wallach, Danny Puller, und Fernando Pereira. 2008. „Generating summary keywords for emails using topics.“ *Proceedings of the 13th International Conference on Intelligent User Interfaces*. 199-206.
- Dwivedi, Sanjay K., und Chandrakala Arya. 2016. „Automatic text classification in information retrieval: A survey.“ *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*. Udaipur, India: ACM.
- El-Beltagy, Samhaa R., und Rafea Ahmed. 2010. „KP- Miner: Participation in SemEval-2.“ *Proceedings of the 5th International Workshop on Semantic Evaluation*. 190-193.
- Elsevier B.V. kein Datum. Scopus. Zugriff am 28. 1 2019. <https://www.scopus.com>.
- Ercan, G., und I. Cicekli. 2007. „Using lexical chains for keyword extraction.“ *Information Processing and Management* vol. 43 (6). 1705- 1714.
- Erkan, G., und D. R. Radev. 2004. „Lexrank: graph-based lexical centrality as salience in text summarization.“ *Journal of Artificial Intelligence Research*. 457-479.
- European Union. kein Datum. *About CORDIS*. Zugriff am 28. 01 2019. <https://cordis.europa.eu/about/en>.
- . 2018. *CORDIS - EU research projects under Horizon 2020 (2014-2020)*. 12. 10. Zugriff am 28. 01. 2019. <https://data.europa.eu/euodp/en/data/dataset/cordish2020projects>.
- . kein Datum. *What is Horizon 2020?* Zugriff am 28. 01 2019. <https://ec.europa.eu/programmes/horizon2020/en/what-horizon-2020>.
- ExplosionAI GmbH. kein Datum. spaCy. Zugriff am 28. 01 2019. <https://spacy.io>.
- . 2017. *spaCy Github Stopwordlist*. Zugriff am 28. 01 2019. https://github.com/explosion/spaCy/blob/master/spacy/lang/de/stop_words.py.
- Fayyad, Usama M., und Keki B. Irani. 1993. „Multi-interval discretization of continuous-valued attributes for classification learning.“ *Proc IJCAI'93*. Ann Arbor Michigan. 1022–1027.
- Feldman, Ronen, und James Sanger. 2006. *THE TEXT MINING HANDBOOK*. Cambridge, UK: Cambridge University Press.
- Florescu, Corina, und Cornelia Caragea. 2017. „PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents.“ *Proceedings of the 55th*

- Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 1105–1115.
- Forman, George. 2003. „An Extensive Empirical Study of Feature Selection Metrics for Text Classification.“ *Journal of Machine Learning Research* 3. 1289-1305.
- Frank, Eibe, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, und Craig G. Nevill-Manning. 1999. „Domain-specific keyphrase extraction.“ In *Proceedings of 16th International Joint Conference on Artificial Intelligence*. 668–673.
- Fruchterman, Thomas M., und Edward M. Reingold. 1991. „Graph drawing by force-directed placement.“ *Software—Practice & Experience archive Volume 21 Issue 11*. New York, USA: John Wiley & Sons, Inc. 1129 - 1164.
- Fukumoto, F., Y. Sekiguchi, und Y. Suzuki. 1998. „Keyword extraction of radio news using term weighting with an encyclopedia and newspaper articles.“ *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 373-374.
- Gao, Xia, und Jiancheng Guan. 2009. „Networks of scientific journals: An exploration of Chinese patent data.“ *Scientometrics* 80(1). 283-302.
- George, Damien P., und Rob J. Knegjens. kein Datum. *About Paperscape*. Zugriff am 28. 01 2019. https://blog.paperscape.org/?page_id=2.
- . 2019. *Paperscape*. 15. 02. Zugriff am 15. 02 2019. <https://paperscape.org>.
- Gephi.org. kein Datum. *Gephi Download*. Zugriff am 28. 01 2019. <https://gephi.org/users/download/>.
- . kein Datum. *Gephi Features*. Zugriff am 28. 01 2019. <https://gephi.org/features/>.
- . kein Datum. *Gephi. The Open Graph Viz Platform*. Zugriff am 28. 01 2019. <https://gephi.org>.
- Glänzel, Wolfgang. 2001. „National characteristics in international scientific co- authorship relations.“ *Scientometrics*, 51(1). 69–115.
- Gollapalli, Sujatha Das, und Cornelia Caragea. 2014. „Extracting keyphrases from research papers using citation networks.“ *Proceedings of the 28th American Association for Artificial Intelligence*. 1629–1635.
- Grineva, Maria, Maxim Grinev, und Dmitry Lizorkin. 2009. „Extracting key terms from noisy and multitheme documents.“ In *Proceedings of the 18th International Conference on World Wide Web*. 661–670.
- Gupta, Er. Tanya. 2017. „KEYWORD EXTRACTION: A REVIEW.“ *International Journal of Engineering Applied Sciences and Technology Vol. 2, Issue 4*. 215-220.
- Gupta, Shashank. 2018. *Towards Data Science*. 11. 01. Zugriff am 20. 01 2019. <https://towardsdatascience.com/automated-text-classification-using-machine-learning-3df4f4f9570b>.
- Gutwin, Carl, Gordon Paynter, Ian Witten, Craig Nevill-Manning, und Eibe Frank. 1999. „Improving browsing in digital libraries with keyphrase indexes.“ *Decision Support Systems*. 81–104.
- HaCohen-Keren, Y, Z Gross, und A Masa. 2005. „Automatic Extraction and Learning of Keyphrases from Scientific Articles Computational Linguistics and Intelligent Text Processing.“ *Proceedings of 6th Int. Conference CICLing*. Mexico City, Mexico: LNCS 3406. 657-669.
- HaCohen-Kerner, Yaakov, Rakefet Dilmon, Shimon Friedlich, und Daniel Nissim Cohen. 2015. „Distinguishing between True and False Stories using various Linguistic Features.“ *PACLIC-29: The 29th Pacific Asia Conference on Language, Information and Computing*. Shanghai, China.
- Hammouda, Khaled M., Diego N. Matute, und Mohamed S. Kamel. 2005. „Keyphrase extraction for document clustering.“ *Proceedings of the 4th International Conference on Machine Learning and Data Mining in Pattern Recognition*. 265-274.

- Hasan, Kazi Saidul, und Vincent Ng. 2014. „Automatic Keyphrase Extraction: A Survey of the State of the Art.“ *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1262–1273.
- Havre, Susan, Elizabeth Hetzler, Paul Whitney, und Lucy Nowell. 2002. „ThemeRiver: Visualizing Thematic Changes in Large Document Collections.“ *IEEE Transactions on Visualization and Computer Graphics*, 8(1),. 9-20.
- Herman, Ivan, Guy Melançon, und M. Scott Marshall. 2000. „Graph visualization and navigation in information visualization: A survey.“ *IEEE Transactions on Visualization and Computer Graphics Volume: 6 , Issue: 1*. 24 - 43.
- Heyer, Gerhard, Uwe Quasthoff, und Thomas Wittig. 2008. *Text Mining: Wissensrohstoff Text*. Bochum: W3L-Verlag.
- Hovy, E., und C.-Y. Lin. 1998. „Automated text summarization and the summarist system.“ *Proceedings of a workshop on held at Baltimore*. 197-214.
- Hu, Yifan. 2005. „Efficient and high quality force-directed graph drawing.“
- Huang, Chong, Yonghong Tian, Zhi Zhou, Charles X. Ling, und Tiejun Huang. 2006. „Keyphrase extraction using semantic networks structure analysis.“ In *Proceedings of the 6th International Conference on Data Mining*. 275–284.
- Hulth, Anette. 2003. „Improved automatic keyword extraction given more linguistic knowledge.“ In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. 216– 223.
- Hulth, Anette, und Bea ta B. Megyesi. 2006. „A study on automatically extracted keywords in text categorization.“ *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. 537-544.
- Humphreys, J. K. 2002. *Phraserate: An html key-phrase extractor*. Tech. Rep, California, USA: Dept. of Computer Science.
- IBM. 2017. *Natural Language Understanding Kategoriehierarchie*. 23. 10. Zugriff am 28. 01 2019. <https://cloud.ibm.com/docs/services/natural-language-understanding/categories.html>.
- . kein Datum. *IBM Cloud Registrierung*. Zugriff am 28. 01 2019. <https://cloud.ibm.com/registration>.
 - . kein Datum. *Natural Language Understanding Categories*. Zugriff am 28. 01 2019. <https://cloud.ibm.com/apidocs/natural-language-understanding?language=python#concepts>.
 - . kein Datum. *Watson Natural Language Classifier*. Zugriff am 28. 01 2019. <https://www.ibm.com/cloud/watson-natural-language-classifier>.
- Ignatov, Gabe, und Rada Mihalcea. 2017. *Text Mining A Guidebook for the Social Sciences*. SAGE Publications Inc.
- Ikonomakis, Emmanouil K., Sotiris Kotsiantis, und V. Tampakas. 2005. „Text Classification Using Machine Learning Techniques.“ *WSEAS Transactions on Computers* 4(8). 966-974.
- Ingersoll, Grant S., Thomas S. Morton, und Andrew L. Farris. 2013. *Taming Text. How to find, organize and manipulate it*. Manning Publications Co.
- Jackson, Peter, und Isabelle Moulinier. 2002. *Natural Language Processing for Online Applications*. Amsterdam / Philadelphia: John Benjamins Publishing Company.
- Jacomy, Mathieu, Tommaso Venturini, Sébastien Heym, und Mathieu Bastian. 2014. „ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software.“ *PLoS ONE* 9(6): e98679.
- Jiang, Jing. 2012. „Information Extraction from Text.“ In *Mining Text Data*, von Charu C. Aggarwal und Zhai Cheng-Xiang, 11–41. New York: Springer US.
- Jindal, Rajni, Ruchika Malhotra, und Abha Jain. 2015. „Techniques for text classification: Literature review and current trends.“ [www.webology.org](http://www.webology.org/2015/v12n2/a139.pdf). 26. 12. Zugriff am 20. 01 2019. <http://www.webology.org/2015/v12n2/a139.pdf>.

- Johnson, Brian, und Ben Shneiderman. 1991. „Tree-maps: a space-filling approach to the visualization of hierarchical information structures.“ *Proceeding Visualization '91*. San Diego, CA, USA. 284-291.
- Kandylas, Vasileios, S. Phineas Upham, und Lyle H. Ungar. 2010. „Analyzing knowledge communities using foregroundand background clusters.“ *ACM Transactions on Knowledge Discovery from Data (TKDD)*. 1-34.
- Kao, Anne, und Stephen R. Poteet. 2010. *Natural language processing and text mining*. . London: Springer.
- Kaur, Manjot, und Navjot Kaur. 2013. „Web Content Mining Techniques: A Survey.“ *IJCST Vol. 4, ISsue 2*. 149-152.
- Kim, Su Nam, Olena Medelyan, Min-Yen Kan, und Timothy Baldwin. 2010. „SemEval-2010 Task 5: Automatic keyphrase extraction from scientific articles.“ *Proceedings of the 5th International Workshop on Semantic Evaluation*. 21-26.
- Kim, Su Nam, und Timothy Baldwin. 2012. „Extracting keywords from multi-party live chats.“ *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*. 199-208.
- Klein, Bernd. kein Datum. *Einführung in die Text-Klassifikation*. Zugriff am 20. 01 2019. https://www.python-kurs.eu/text_klassifikation_einfuehrung.php.
- Ko, Youngjoong, und Jungyun Seo. 2000. „Automatic Text Categorization by Unsupervised Learning.“ *COLING '00 Proceedings of the 18th conference on Computational linguistics - Volume 1*. Saarbrücken, Germany: ACM. 453-459 .
- Koppel, Moshe, Shlomo Argamon, und Anat Rachel Shimoni. 2002. „Automatically Categorizing Written Texts by Author Gender.“ *Literary and Linguistic Computing, Volume 17, Issue 4*. 401–412.
- Korde, Vandana, und N. C. Mahender. 2012. „Text classification and classifiers: a survey.“ *International Journal of Artificial Intelligence & Applications* 3(2). 85-99.
- Levenshtein, V. 1966. „Binary Codes Capable of Correcting Deletions, Insertions, and Reversals.“ *Soviet Physics Dokl, Vol 10(8)*. 707-710.
- Lewis, David D. 1998. „Naive (Bayes) at Forty: The independence assumption in informal retrieval.“ *Lecture Notes in Computer Science* 1398. Springer. 4-15.
- Leydesdorff, Loet, und Thomas Schank. 2008. „Dynamic animations of journal maps: Indicators of structural changes and interdisciplinary developments.“ *Journal of the American Society for Information Science and Technology*, 59(11). 1810–1818.
- Litvak, M., und M. Last. 2008. „Graph-based keyword extraction for singledocument summarization.“ *Proceedings of the workshop on Multisource Multilingual Information Extraction and Summarization*. 17-24.
- Liu, Feifan, Deana Pennell, Fei Liu, und Yang Liu. 2009a. „Unsupervised approaches for automatic keyword extraction using meeting transcripts.“ *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 620-628.
- Liu, Hugo, und Rada Mihalcea. 2007. „Of Men, Women, and Computers: Data-Driven Gender Modeling for Improved User Interfaces.“ Paper.
- Liu, Lu. 2008. *An Evaluation of Kea: An Automatic Keyphrase Extraction Algorithm*. New Brunswick: School of Communication, Information and Library Studies, Rutgers University.
- Liu, Zhiyuan, Peng Li, Yabin Zheng, und Maosong Sun. 2009b. „Clustering to Find Exemplar Terms for Keyphrase Extraction.“ *In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. 257–266.
- Liu, Zhiyuan, Wenyi Huang, Yabin Zheng, und Maosong Sun. 2010. „Automatic keyphrase extraction via topic decomposition.“ *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 366–376.
- Long, Lim Kian, Lim Chien Hui, Gim Yeong Fook, und Wan Mohd Nazmee Wan Zainon. 2017. „A Study on the Effectiveness of Tree-Maps as Tree Visualization

- Techniques.“ *4th Information Systems International Conference 2017*. Bali, Indonesia: Elsevier B.V. 108-115.
- Lopez, Patrice, und Laurent Romary. 2010. „HUMB: Automatic key term extraction from scientific articles in GROBID.“ *Proceedings of the 5th International Workshop on Semantic Evaluation*. 248-251.
- Lovins, Julie Beth. 1968. „Development of a Stemming Algorithm.“ *Mechanical Translation and Computational Linguistics*, vol. 11, nos. 1 and 2. Cambridge, Massachusetts. 22-31.
- Luhn, H. P. 1957 . „A statistical approach to mechanized encoding and searching of literary information.“ *IBM Journal of Research and Development Volume 1 Issue 4*. Riverton, NJ, USA . 309-317 .
- López, Roberto, Luis Carlos González Gurrola, Leonardo Trujillo, Olanda Prieto, Graciela Ramírez, Antonio Posada, Perla Juárez-Smith, und Leticia Méndez. 2018. *How Am I Driving? Using Genetic Programming to Generate Scoring Functions for Urban Driving Behavior*. Article, MDPI.
- Mahinovs, Aigars, und Ashutosh Tiwari. 2007. *TEXT CLASSIFICATION METHOD REVIEW*. Report, Cranfield, United Kingdom: Cranfield University.
- Mani, I., und M. T. Maybury. 1999. „Advances in automatic text summarization.“ Vol. 293. MIT Press.
- Martin, Shawn, W. Michael Brown, Richard Klavans, und Kevin W. Boyack. 2011. „OpenOrd: an open-source toolbox for large graph layout.“ *SPIE Proceedings Vol. 7868: Visualization and Data Analysis 2011*. 11.
- Matsuo, Yutaka, und Mitsuru Ishizuka. 2004. „Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information.“ *International Journal on Artificial Intelligence Tools*. 157-169.
- McCallum, Andrew, und Kamal Nigam. 1998. „A Comparison of Event Models for Naive Bayes Text Classification.“ *Learning for Text Categorization AAAI Workshop*. 41-48.
- Mebrahtu, Andemariam, und Balu Srinivasulu. 2017. „Web Content Mining Techniques and Tools.“ *International Journal of Computer Science and Mobile Computing*, Vol.6 Issue.4. 49-55.
- Medelyan, Olena, Eibe Frank, und Ian H. Witten. 2009. „Human-competitive tagging using automatic keyphrase extraction.“ In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. 1318–1327.
- Michie, D., D. J. Spiegelhalter, und C. C. Taylor. 1994. *Machine Learning, Neural and Statistical Classification*.
- Mihalcea, Rada. 2004. „Graph-based ranking algorithms for sentence extraction, applied to text summarization.“ *Proceedings of the Association for Computational Linguistics on Interactive poster and demonstration sessions*.
- Mihalcea, Rada, und Paul Tarau. 2004. „TextRank: Bringing Order into Texts.“ *'Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*. University of North Texas.
- Miller, Thomas W. 2005. *Data and Text Mining: A Business Applications Approach*. Pearson Prentice Hall.
- Miner, Gary, John Elder, Andrew Fast, Thomas Hill, Robert Nisbet, und Dursun Delen. 2012. *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*. Academic Press.
- Moritz Stefaner and Studio NAND. kein Datum. *Mapping Scientific Excellence*. Zugriff am 28. 01 2019. <http://www.excellencemapping.net>.
- Morris, Steven A., G. Yen, Zheng Wu, und Benyam Asnake. 2003. „Timeline visualization of research fronts.“ *Journal of the American Society for Information Science and Technology*, 55(5). 413–422.

- Nagel, Till, Erik Duval, und Frank Heidmann. 2011. „Visualizing Geospatial Co-Authorship Data on a Multitouch Tabletop.“ *Conference: Smart Graphics - 11th International Symposium, SG 2011*,. Bremen, Germany.
- National Center for Biotechnology Information. kein Datum. *PubMed*. Zugriff am 28. 01 2019. <https://www.ncbi.nlm.nih.gov/pubmed/>.
- National Science Foundation. kein Datum. *National Science Foundation*. Zugriff am 28. 01 2019. <https://www.nsf.gov/awardsearch/>.
- NetworkX developers. 2018. *NetworkX*. Zugriff am 28. 01 2019. <https://networkx.github.io>.
- Newman, David, Nagendra Koilada, Jey Han Lau, und Timothy Baldwin. 2012. „Bayesian text segmentation for index term identification and keyphrase extraction.“ *In Proceedings of the 24th International Conference on Computational Linguistics*. 2077–2092.
- Nguyen, Thuy Dung, und Minh-Thang Luong. 2010. „WINGNUS: Keyphrase Extraction Utilizing Document Logical Structure.“
- Nguyen, Thuy Dung, und Min-Yen Kan. 2007. „Keyphrase extraction in scientific publications.“ *Proceedings of the International Conference on Asian Digital Libraries*. 317-326.
- Nigam, Kamal, Andrew Kachites Mccallum, Sebastian Thrun, und Tom Mitchell. 2000. „Text Classification from Labeled and Unlabeled Documents using EM.“ *Machine Learning Volume 39, Issue 2–3*. SpringerLink. 103–134.
- Ohsawa, Y., N. E. Benson, und M. Yachida. 1998. „Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor.“ *Research and Technology Advances in Digital Libraries, IEEE*. 12-18.
- Open Knowledge Maps - Verein zur Förderung der Sichtbarkeit wissenschaftlichen Wissens. kein Datum. *Open Knowledge Maps*. Zugriff am 28. 01 2019. <https://openknowledgemaps.org>.
- Pang, Bo, und Lillian Lee. 2008. „Opinion mining and sentiment analysis.“ *Foundations and Trends in Information Retrieval Vol. 2, No 1-2*. 1-135.
- Patidar, Vinod, Divakar Singh, und Anju Singh. 2013. „A Novel Technique of Email Classification for Spam Detection.“ *International Journal of Applied Information Systems 5(10)*. 15-19.
- Porter, M. F. 1980. „An algorithm for suffix stripping.“ *Program, Vol. 14 Issue: 3*. 130-137.
- Prasad, K. N. S. S. V., S. K. Saritha, und Dixa Saxena. 2017. „A Survey Paper on Concept Mining in Text Documents.“ *International Journal of Computer Applications (0975 – 8887) Volume 166 – No.11*. 7-10.
- Pritchard, Alan. 1969. „Statistical Bibliography or Bibliometrics?“ *Journal of Documentation 25(4)*. 348-349.
- ProQuest LLC. kein Datum. *RefWorks*. Zugriff am 28. 01 2019. <https://www.refworks.com>.
- Ramos, J. 2003. „Using tf-idf to determine word relevance in document queries.“ *Proceedings of the first instructional conference on machine learning*. 1-4.
- Roco, M. C., W. S. Bainbridge, B Tonn, und G Whitesides. 2013. „Convergence of Knowledge, Technology and Society.“ *Science Policy Reports*.
- Rose, Stuart, Dave Engel, Nick Cramer, und Wendy Cowley. 2010. „Automatic Keyword Extraction from Individual Documents.“ *In Text Mining: Applications and Theory*.
- Rosvall, M., und C. T. Bergstrom. 2010. „Mapping change in large networks.“ *PLoS ONE, 5(1)*. 1-10.
- Salton, Gerhard. 1989. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley.
- Salton, Gerhard, A. Wong, und C. S. Yang. 1975. „A vector space model for automatic indexing.“ *Communications of the*. New York, NY, USA: ACM. 613–620.
- Salton, Gerhard, Amit Singhal, Mandar Mitra, und Chris Buckley. 1997. „Automatic text structuring and summarization.“ *Information Processing and Management: an*

- International Journal - Special issue: methods and tools for the automatic construction of hypertext.* Tarrytown, NY, USA: Pergamon Press, Inc. 193 - 207.
- Salton, Gerhard, C. S. Yang, und C. T. Yu. 1975. „A Theory of Term Importance in Automatic Text Analysis.“ *Journal of the American society for Information Science*. 33-44.
- Salton, Gerhard, und Christopher Buckley. 1988. „Term-weighting approaches in automatic text retrieval.“ *Information Processing and Management: an International Journal Volume 24 Issue 5*. Tarrytown, NY, USA. 513-523.
- Salton, Gerhard, und Christopher Buckley. 1991. „Automatic text structuring and retrieval experiments in automatic encyclopedia searching.“ *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM. 21-30.
- Sebastiani, Fabrizio. 1999. „A tutorial on automated text categorisation.“ *Proceedings of the 1st Argentinian Symposium on Artificial Intelligence (ASA'I'99)*,. Buenos Aires, AR. 7-35.
- . 2002. „Machine learning in automated text categorization.“ *ACM Computing Surveys* 34. 1-47.
 - . 2005. „Text categorization.“ *Text Mining and its Applications to Intelligence, CRM and Knowledge Management*. WIT Press. 109-129.
- Shafabady, Niusha, L. H. Lee, R. Rajkumar, V. P. Kallimani, Nik Ahmad Akramc, und Dino Isac. 2016. „Using unsupervised clustering approach to train the support vector machine for text classification.“ *Neurocomputing Volume 211*. 4–10.
- Skupin, André. 2009. „Discrete and continuous conceptualizations of science: Implications for knowledge domain visualization.“ *Journal of Informetrics*, 3 (3). 233–245.
- Small, Henry. 2006. „Tracking and predicting growth areas in science.“ *Scientometrics*, 68(3). 595-610.
- . 1999. „Visualizing science by citation mapping.“ *Journal of the American Society for Information Science*, 50(9). 799–813.
- Srivastava, Tavish. 2015. *Difference between machine learning & statistical modeling*. 1. 07. Zugriff am 20. 01. 2019. <https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/>.
- Sterckx, Lucas, Thomas Demeester, Johannes Deleu, und Chris Develder. 2015. „Topical Word Importance for Fast Keyphrase Extraction.“ *24th International Conference on World Wide Web*. 121-122 .
- Taeho, Jo. 2018. *Text Mining Concepts, Implementation, and Big Data Challenge*. Springer International Publishing AG.
- Thangaraj, M., und M. Sivakami. 2018. „Text Classification Techniques: A Literature Review.“ *Interdisciplinary Journal of Information, Knowledge, and Management • Volume 13* . 117-135.
- Thomas, J. R., S. K. Bharti, und K. S. Babu. 2016. „Automatic keyword extraction for text summarization in e-newspapers.“ *Proceedings of the International Conference on Informatics and Analytics*. 86-93.
- Tomokiyo, Takashi, und Matthew Hurst. 2003. „A language model approach to keyphrase extraction.“ *In Proceedings of the ACL Workshop on Multiword Expressions*. 33–40.
- Turney, P. D. 1997. *Extraction of Keyphrases from Text: Evaluation of Four Algorithms*. National Research Council of Canada.
- . 1999. „Learning to Extract Keyphrases from Text.“ *Technical Report ERB-1057*. National Research Council, Institute for Information Technology.
- van der Plas, L., V. Pallotta, M. Rajman, und H. Ghorbel. 2004. „Automatic keyword extraction from spoken text: a comparison of two lexical resources: the edr and WordNet.“ *Procedings of the LREC 2004 international conference*. Lisbon, Portugal. 2205-2208.

- Van Eck, Nees Jan, Ludo Waltman, Rommert Dekker, und Jan van den Berg. 2010. „A Comparison of Two Techniques for Bibliometric Mapping: Multidimensional Scaling and VOS.“ *Journal of the American Society for Information Science and Technology*, 61(12). 2405–2416.
- Van Eck, Nees Jan, und Ludo Waltman. 2010. „Software survey: VOSviewer, a computer program for bibliometric mapping.“ *Scientometrics*, 84(2). Springer. 523–538.
- Van Eck, Nees Jan, und Ludo Waltman. 2011. *Text mining and visualization using VOSviewer*. Artikel, arXiv.
- . 2019. „VOSviewer.“ *VOSviewer Manual*. 10. 01. Zugriff am 28. 01 2019. http://www.vosviewer.com/documentation/Manual_VOSviewer_1.6.10.pdf.
- Van Wijk, Jarke J., und Huub Van de Wetering. 1999. „Cushion Tree-maps: Visualization of Hierarchical Information.“ *EEE Symposium on Information Visualization (INFOVIS '99)*. San Francisco, USA. 1-6.
- Vieira, A. S., L. Borrajo, und E. L. Iglesias. 2016. „Improving the text classification using clustering and a novel HMM to reduce the dimensionality.“ *Computer Methods and Programs in Biomedicine* 136. 119-130.
- Waltman, Ludo, Nees Jan Van Eck, und Ed C.M. Noyons. 2010. „A unified approach to mapping and clustering of bibliometric networks.“ *Journal of Informetrics*, 4(4). 629–635.
- Wan, Xiaojun, und Jianguo Xiao. 2008. „Single document keyphrase extraction using neighborhood knowledge.“ *Proceedings of the 2008 American Association for Artificial Intelligence*. 855–860.
- Wetzel, Kai. kein Datum. *pebbles - using Circular Treemaps to visualize disk usage*. Zugriff am 28. 01 2019. <http://lip.sourceforge.net/ctreemap.html>.
- White, Howard D., und Katherine W. McCain. 1997. „Visualization of literatures.“ *Annual Review of Information Science and Technology (ARIST)*, v32. 99-168.
- Wiebe, Janyce, Theresa Wilson, und Claire Cardie. 2005. „Annotating Expressions of Opinions and Emotions in Language.“ *Language Resources and Evaluation Volume 39, Issue 2–3*. Scpringer Link. 165-210.
- Witten, Ian H., Gordon W. Paynter, Eibe Frank, Carl Gutwin, und Craig G. Nevill-Manning. 1999. „KEA: Practical automatic keyphrase extraction.“ In *Proceedings of the 4th ACM Conference on Digital Libraries*. 254–255.
- Witten, Ian, und Olena Alyona Medelyan. 2006. „Thesaurus based automatic keyphrase indexing.“ *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '06)*. Chapel Hill, NC, USA: IEEE.
- Yang, Y., und J. O. Pedersen. 1997. „A comparative study on feature selection in text categorization.“ *Proc. the Fourteenth International Conference on Machine Learning (ICML-97)*. 412–420.
- Yao, Xiaohui, Jingwen Yan, Michael Ginda, Katy Börner, Andrew J. Saykin, und Li Shen. 2017. „Plos.org.“ *Mapping longitudinal scientific progress, collaboration and impact of the Alzheimer's disease neuroimaging initiative*. 2. 11. Zugriff am 28. 01 2019. <https://doi.org/10.1371/journal.pone.0186095>.
- Yih, Wen-Tau, Joshua Goodman, und Vitor R. Carvalho. 2006. „Finding advertising keywords on web pages.“ In *Proceedings of the 15th International Conference on World Wide Web*. 213–222.
- Zhang, Chengzhi, Huilin Wang, Yao Liu, Dan Wu, Yi Liao, und Bo Wang. 2008. „Automatic keyword extraction from documents using conditional random fields.“ *Journal of Computational Information Systems*, vol. 4 (3). 1169-1180.
- Zhang, Juan, Jun Xie, Wanli Hou, Xiaochen Tu, Jing Xu, Fujian Song, Zhihong Wang, und Zuxun Lu. 2012. „Mapping the Knowledge Structure of Research on Patient Adherence: Knowledge Domain Visualization Based Co-Word Analysis and Social Network Analysis.“ Artikel.

- Zhang, Kuo, Hui Xu, Jie Tang, und Juan-Zi Li. 2006. „Keyword Extraction Using Support Vector Machine.“ *Advances in Web-Age Information Management, 7th International Conference*. 85–96.
- Zhang, Ting, Ze Yuan Liu, und Tai Yang Zhao. 2008. „Timeline and Landscape: A Case Study of Visualizing the Evolution of Science Communication Research Front.“ *Proceedings of WIS 2008*. Berlin, Germany. 1–10.
- Zhang, Yongzheng, Nur Zincir-Heywood, und Evangelos Miliots. 2004. „World Wide Web site summarization.“ *Web Intelligence and Agent Systems*. 39–53.
- Zhao, Dangzhi, und Andreas Strotmann. 2008. „Evolution of research activities and intellectual influences in information science 1996–2005: Introducing author bibliographic-coupling analysis.“ *Journal of the American Society for Information Science and Technology*, 59(13). 2070–2086.
- Zurek, W. H. 1985. „Cosmological experiments in superfluid helium?“ *Naturevolume* 317. 505–508.

Anhang

A CORDIS Beispielprojekt

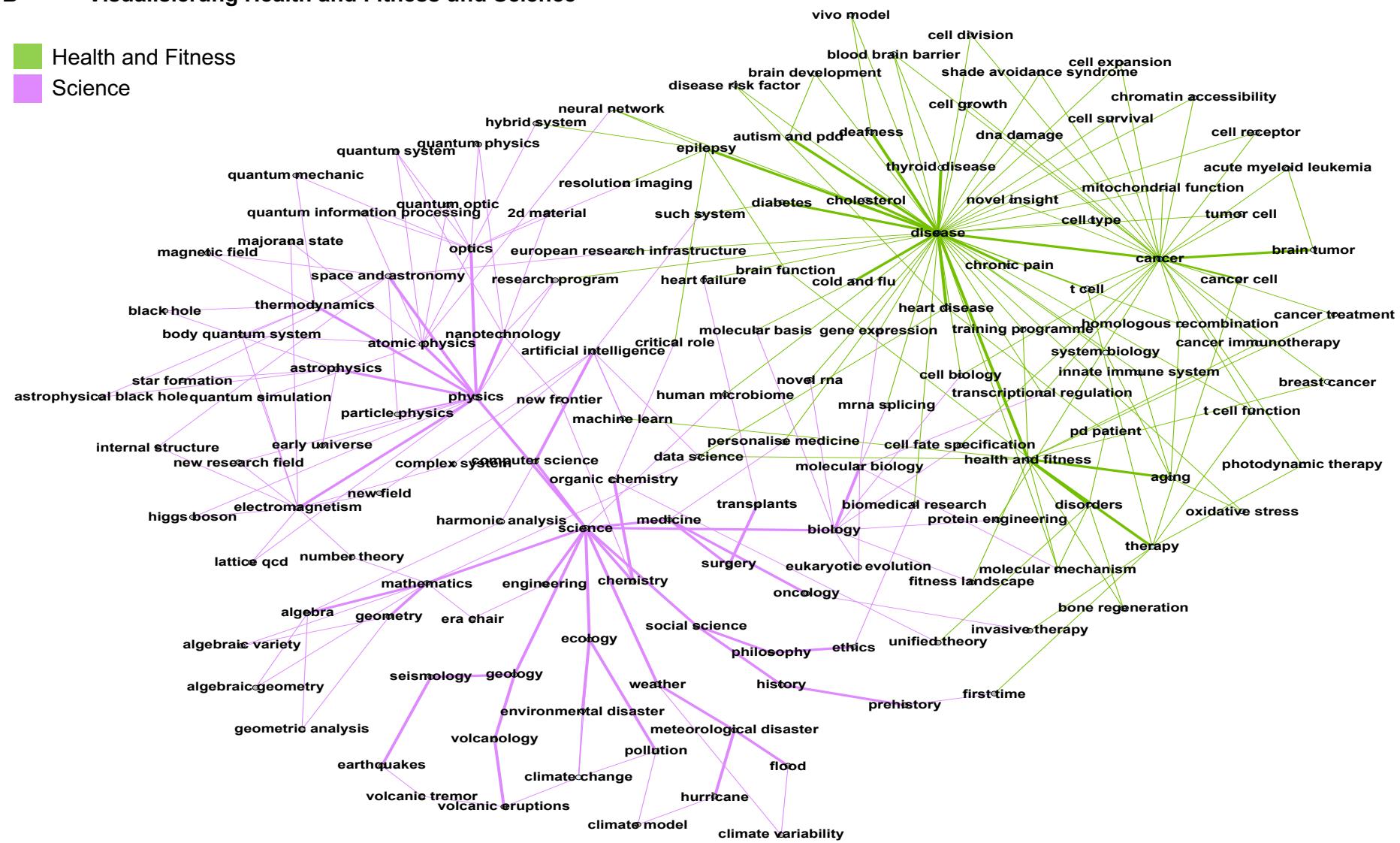
Anhand dieses Beispielprojektes werden die Ergebnisse der Datenextrahierung, Klassifizierung und Schlüsselwort-Extrahierung dargestellt. Dabei wurden durch die Klassifizierung drei Kategorieebenen für diesen Text (Titel und Zielbeschreibung) bestimmt. Die im Titel und der Zielbeschreibung gelb markierten Wörter sind beispielhaft vorbestimmte Schlüsselwörter. Mit den drei Schlüsselwort-Extraktionsverfahren YAKE!, KEA und PositionRank wurden die 10 relevantesten Schlüsselwörter extrahiert und anhand der vorbestimmten Schlüsselwörter verglichen. Die Übereinstimmungen wurden ebenfalls jeweils gelb markiert. Dabei ist zu beachten, dass die gefundenen Schlüsselwörter bereits auf ihre Stammform reduziert wurden.

Projekt-ID	101432		
Titel	Innovative gene therapies for epilepsy treatment.		
Zielbeschreibung	<p>A significant part of the costs of neurological diseases on society is associated with epilepsy. About 30-40% of the patients are refractory to pharmacological treatments, which are mostly symptomatic and often have side effects. In few cases surgical intervention is considered and no treatments interfering with or preventing the development of epilepsy are currently available. In this context, EPIXCHANGE aims at exploring, providing the basis for clinical application and implementing in the industrial arena new and unconventional strategies for the therapy of partial epilepsy advancing the state-of-the-art in the field. To achieve this, a strategic partnership will be created, including two internationally recognized academic institutions (UniFE, ULund) and one SME (NsGene) developing encapsulated cell biodelivery (ECB) based therapeutic products. ECBs will be transferred to the academic partners, while knowledge and technology on animal models, viral vectors and BDNF-producing cells will be transferred to the SME. This partnership will implement a joint research programme, which will enable to exploit the complementary competencies and technologies available at each participant site and will increase the knowledge-sharing and technology transfer, as well as the mutual understanding and penetration of the different cultural settings and skills required for both academic and industrial sectors, thus improving partners' RTD capability and competitiveness. This will be combined with transfer of knowledge on complementary skills. Experienced researchers will be also recruited to bring top level experience in techniques that are not present in the consortium. Annual workshops will be organized to benefit the transfer of knowledge programme and the dissemination of results both within the consortium and towards the scientific community.</p>		
Startdatum	2011-12-01		
Enddatum	2015-11-30		
Kategorie	Ebene1	Ebene2	Ebene3
	Health and Fitness	Disease	Epilepsy
Schlüsselwörter	YAKE!	KEA	PositionRank
	1. Innovative gene therapy 2. innovative gene 3. disease on society 4. gene therapy 5. significant part 6. cost of neurological 7. neurological disease 8. epilepsy 9. epilepsy treatment 10. pharmacological treatment	1. this 2. epilepsy 3. academic 4. knowledge 5. available 6. treatment 7. will be transfer 8. industrial 9. transfer 10. be transfer	1. innovative gene therapy 2. epilepsy treatment 3. partial epilepsy 4. significant part 5. epilepsy 6. neurological disease 7. unconventional strategy 8. pharmacological treatment 9. society 10. cost

B Visualisierung Health and Fitness und Science

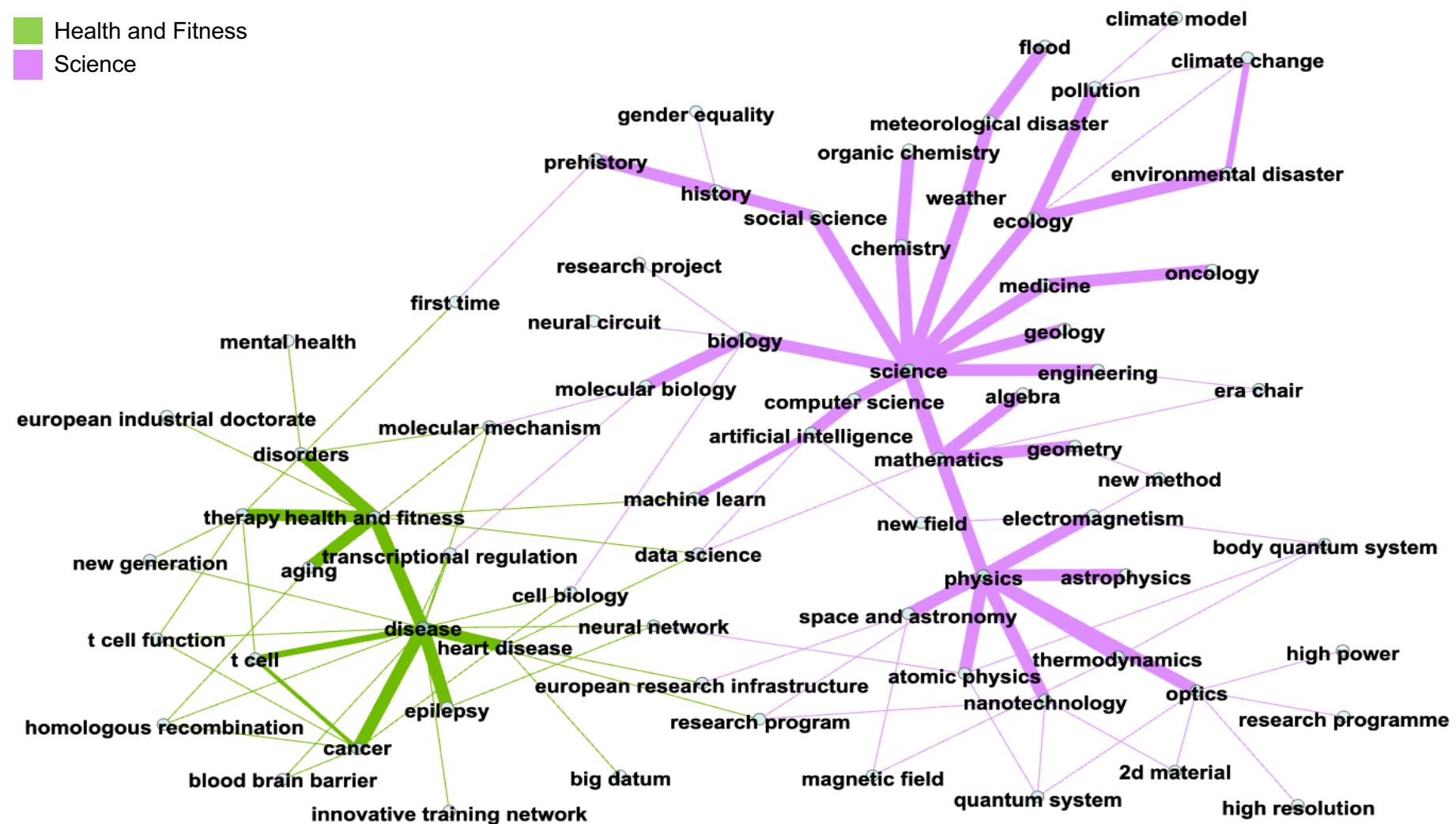
Health and Fitness

Science

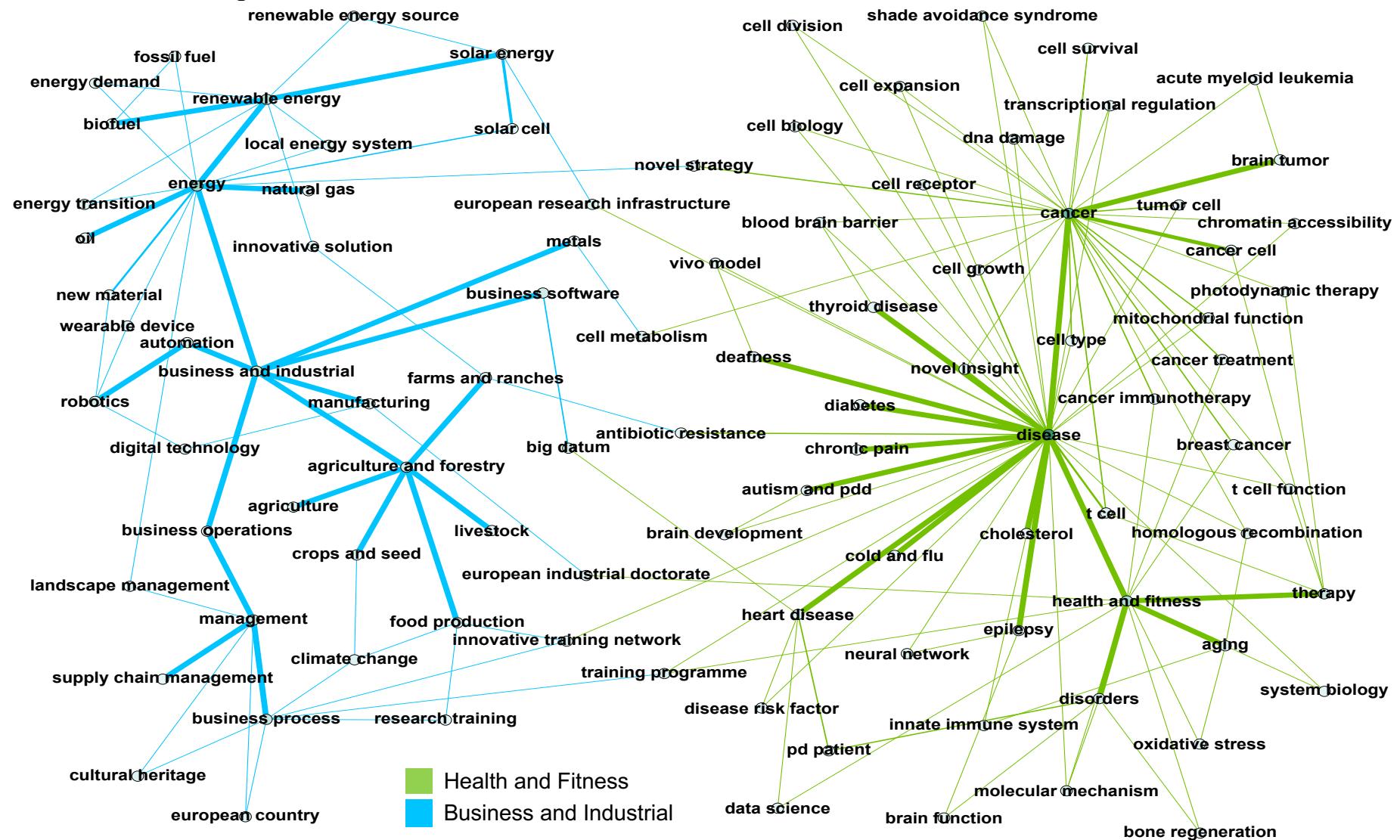


C Visualisierung Health and Fitness und Science Version 2

█ Health and Fitness
█ Science



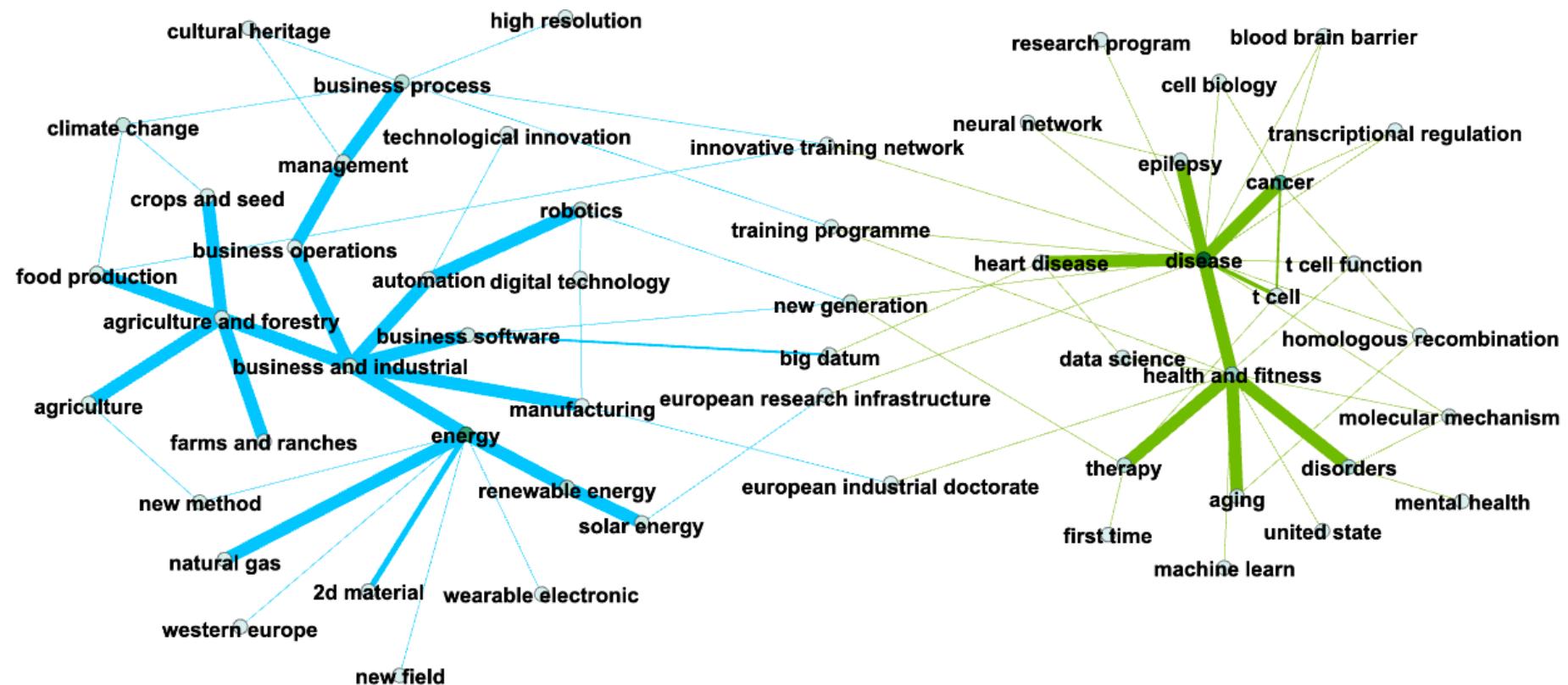
D Visualisierung Health and Fitness und Business and Industrial



E Visualisierung Health and Fitness und Business and Industrial Version 2

Health and Fitness

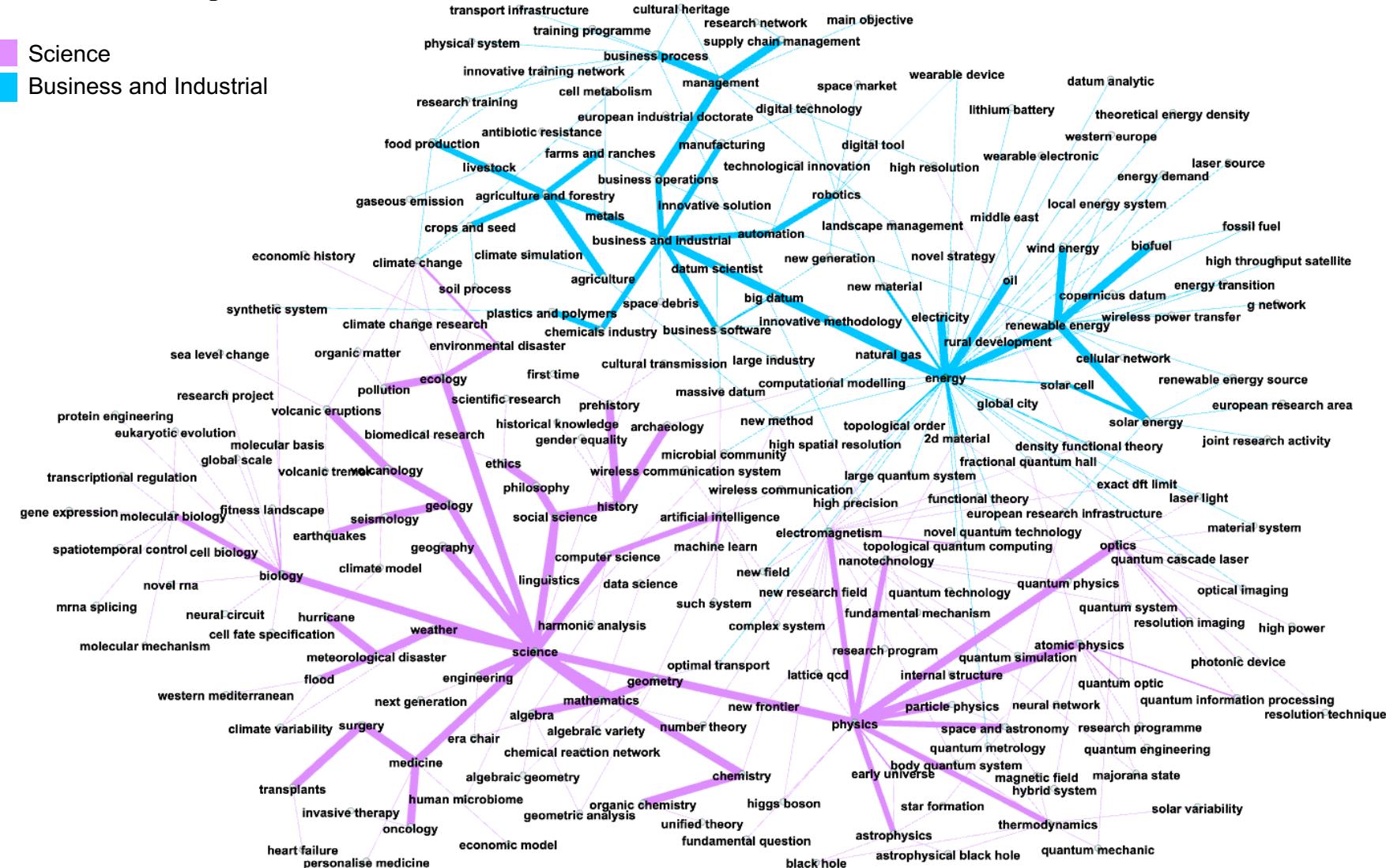
Business and Industrial



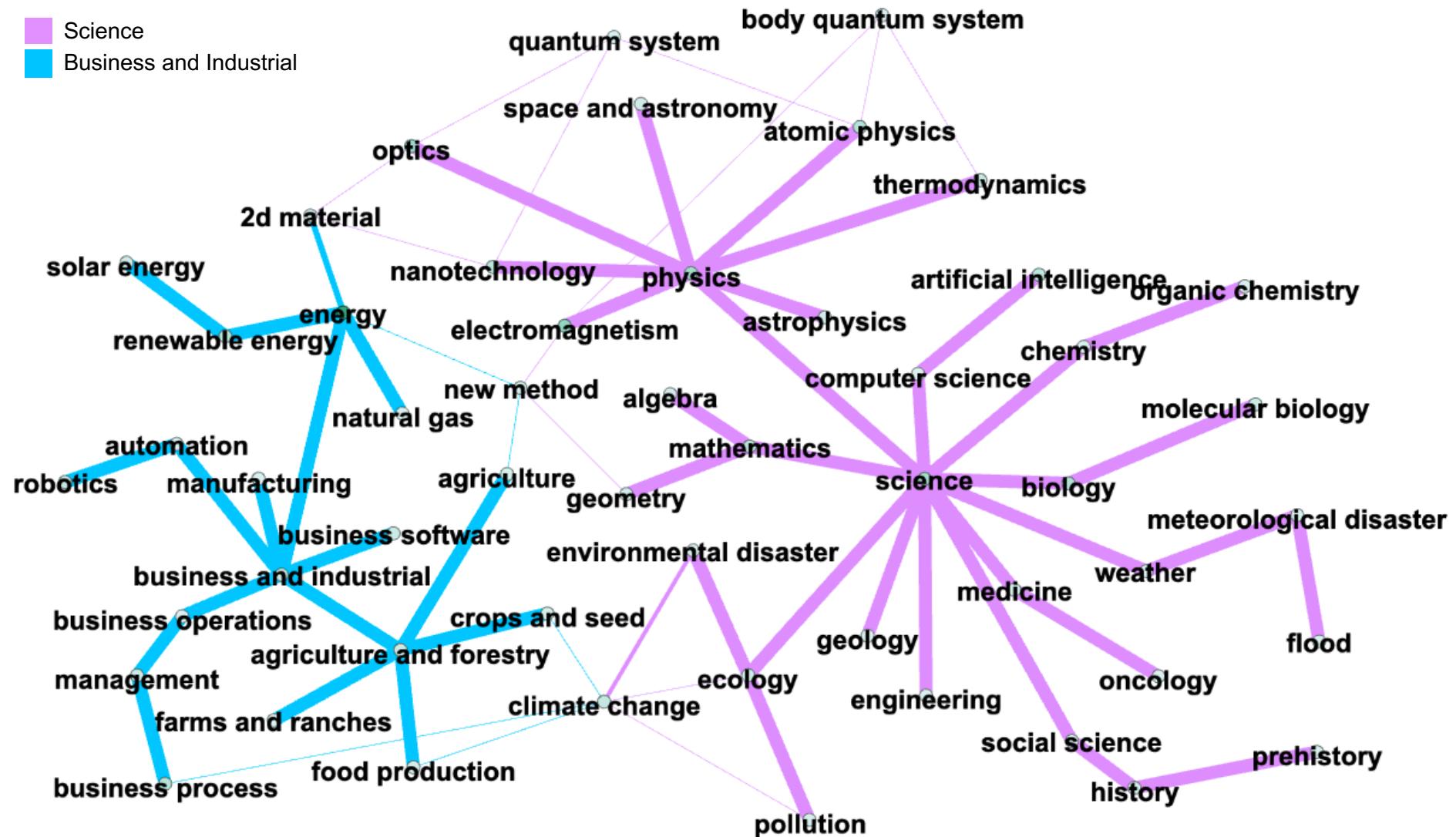
F Visualisierung Science und Business and Industrial

Science

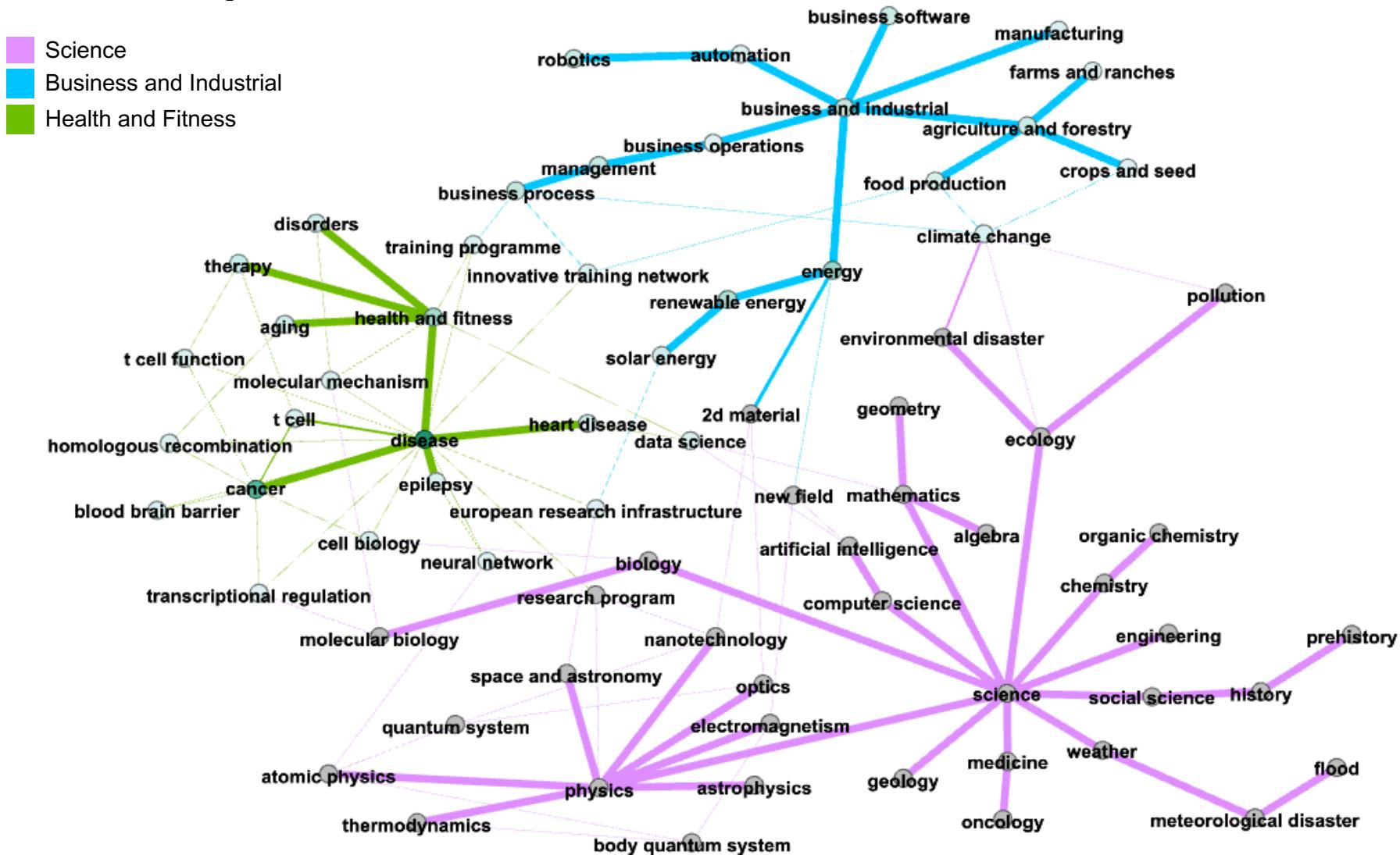
Business and Industrial



G Visualisierung Science und Business and Industrial Version 2



H Visualisierung Science, Business and Industrial und Health and Fitness



Eidesstattliche Versicherung

Ich versichere hiermit, dass ich die vorliegende Bachelorthesis selbstständig und ohne fremde Hilfe angefertigt und keine andere als die angegebene Literatur benutzt habe. Alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnnenden Ausführungen meiner Arbeit sind besonders gekennzeichnet. Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, den 25. Februar 2019

Clemens Brauer