

Lehrstuhl für Wirtschaftsinformatik | Business Intelligence Research

Technische Universität Dresden
Fakultät Wirtschaftswissenschaften

Ein Textbasiertes Recommender- System zur Automatisierten Selekt- ion von Analytischen Verfahrens- klassen

Richard Horn (3885589)¹

Daniel Höschele (3879394)²

Masterarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

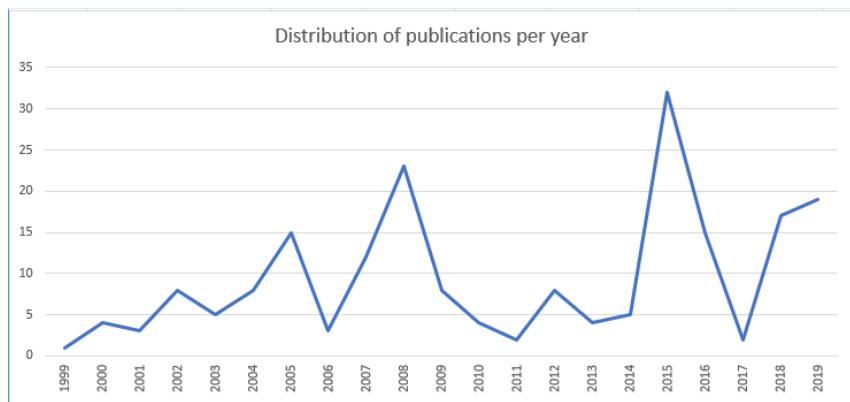
Prüfer: **Prof. Dr. Andreas Hilbert**

Betreuer: **Patrick Zschech**

Eingereicht am: **31.10.2018**

Bearbeitungszeitraum: **19.10.2018-31.10.2018**

Quantitative Zusammenfassung



Kommentiert [m1]:

Eigenschaft	Wert
Quellenanzahl gesamt	XX
Internetquellen	0
Zeitlicher Rahmen	
Älteste Quelle	XX
Neueste Quelle	XX
Anteile	
Bis 1999:	XX
2000 – 2010	XX
2011 bis heute	XX
Anzahl der Abbildungen	
Anzahl der Tabellen	

Kurzfassung

KURZFASSUNG

Schlüsselwörter:

Abstract

ABSTRACT

Keywords:

Inhaltsverzeichnis

ABBILDUNGSVERZEICHNIS	IV
TABELLENVERZEICHNIS	VII
ABKÜRZUNGSVERZEICHNIS	VIII
1 THEMENBESCHREIBUNG UND MOTIVATION¹.....	1
1.1 EINORDNUNG, ABGRENZUNG UND HERAUSFORDERUNGEN	3
1.2 FORSCHUNGSVORHABEN	9
1.2.1 <i>Forschungsdesign¹</i>	9
1.2.2 <i>Forschungsfragen²</i>	10
1.3 METHODISCHES VORGEHEN.....	10
1.3.1 <i>Systematische Literaturrecherche</i>	11
1.3.2 <i>Der Design-Science-Research-Prozess</i>	12
1.4 AUFBAU UND STRUKTUR DER ARBEIT ¹	17
2 ÜBERBLICK DES FORSCHUNGSEMINARS	19
3 FORSCHUNGSSSTAND.....	22
4 ADDITIONALE METHODEN.....	28
4.1 EMBEDDINGS	28
4.1.1 <i>FastText</i>	29
4.1.2 <i>Deep Averaging Networks</i>	31
4.1.3 <i>Universal Sentence Encoder</i>	34
4.2 RNN-BASIERTE TECHNOLOGIEN.....	36
4.2.1 <i>Long Short-Term Memory</i>	38
4.2.2 <i>Gated Recurrent Units</i>	41

4.3	KEYWORD-EXTRACTION-METHODEN	42
4.3.1	<i>YAKE</i>	42
4.3.2	<i>PositionRank</i>	44
4.3.3	<i>TopicRank</i>	45
4.4	ENSEMBLE LEARNING.....	47
4.4.1	<i>Nicht-generative Methoden des Ensemble Learnings</i>	48
4.4.2	<i>Generative Methoden des Ensemble Learnings</i>	49
4.5	NAMED-ENTITY RECOGNITION	52
4.5.1	<i>Regelbasierte und wörterbuchbasierte Vorgehen</i>	52
4.5.2	<i>Lernbasierte Verfahren der NER</i>	53
5	PROZESS DER DATENBESCHAFFUNG	56
5.1	CRAWLING VON DATENBANKEN.....	56
5.2	DATENEVALUIERUNG UND -SELEKTION	59
5.2.1	<i>Regelbasierte Datenbereinigung</i>	59
5.2.2	<i>Entfernen von Ausreißern</i>	60
5.3	DATA AUGMENTATION	64
5.3.1	<i>Methoden zur Substitution und Reihenfolgenanpassung</i>	64
5.3.2	<i>Generation von Textdaten</i>	65
5.3.3	<i>Applizierte Methoden der Data Augmentation</i>	67
5.4	ERSTELLUNG VON DATENSÄTZEN	71
5.4.1	<i>Erstellung der Trainingsdatensätze</i>	71
5.4.2	<i>Erstellung der Validationsdatensätze</i>	73
6	KONZEPTION UND IMPLEMENTIERUNG DES PROTOTYP.....	75
6.1	ANFORDERUNGSDEFINITION.....	75
6.2	FRONTEND KONSEPTIONIERUNG	78
6.3	ENTWICKLUNG DES BACKEND	80

6.3.1	<i>Vorbereitung und Vorverarbeitung der Daten</i>	81
6.3.2	<i>Aufbau der Knowledge Base</i>	83
6.3.3	<i>Nutzung der Knowledge Base</i>	87
6.4	PROGRAMMSTRUKTUR.....	92
6.5	ANFORDERUNGSGEGENÜBERSTELLUNG.....	94
7	EVALUATION DER METHODEN SOWIE DES GESAMTSYSTEMS	98
7.1	EVALUATION VON EMBEDDINGS	98
7.1.1	<i>Extrinsische Evaluation</i>	99
7.1.2	<i>Intrinsische Evaluation</i>	104
7.1.3	<i>Ergebnisse der Evaluation</i>	107
7.2	EVALUATION VON TOPIC-MODELLEN.....	110
7.2.1	<i>Überblick über Evaluationsmethoden</i>	110
7.2.2	<i>Ergebnisse der Evaluation von Topic-Modellen</i>	112
7.3	EVALUIERUNG DER KLASIFIKATIONSMODELLE	118
7.3.1	<i>Überblick multinomialer Evaluationsmetriken</i>	118
7.3.2	<i>Ergebnisse der Evaluation von Klassifikationsmethoden (D)</i>	121
8	ERGEBNISSE.....	125
9	KRITISCHE WÜRDIGUNG (D)	130
10	FAZIT UND AUSBLICK (D, R).....	133
LITERATURVERZEICHNIS		VIII
ANHANG		XXV
EIDESSTATTLICHE ERKLÄRUNG		XXVII

Abbildungsverzeichnis

Abbildung 1: Illustration der allgemeinen Zielstellung	2
Abbildung 2: Illustration der aufgestellten Term-Sets zur Literaturrecherche	12
Abbildung 3: Der Design-Science-Prozess	13
Abbildung 4: Vorgehensmodell des Prototyping	15
Abbildung 5: Struktureller Aufbau der Arbeit.....	17
Abbildung 6: Übersicht der methodischen Kategorien.....	19
Abbildung 7: Architektur des Word2Vec-Modells	30
Abbildung 8: Funktionsweise eines DANs	32
Abbildung 9: Gegenüberstellung DAN und Mittelwertsverfahren.....	34
Abbildung 10: Berechnungsgraph eines Recurrent Neural Networks.....	37
Abbildung 11: Illustration der Funktionsweise einer LSTM-Zelle	39
Abbildung 12: Illustration der Funktionsweise einer GRU-Zelle	41
Abbildung 13: Verarbeitungsschritte des TopicRank-Algorithmus	46
Abbildung 14: Illustration der Funktionsweise des Bagging-Ansatzes	50
Abbildung 15: Vorgehen bei Boosting	51
Abbildung 16: Prozess der Datenbeschaffung und -selektion.....	56
Abbildung 17: Crawling-Ergebnisse in Elsevier	58
Abbildung 18: Density Graph des Datenpools.....	60
Abbildung 19: Streudiagramme des Datenpools.....	61
Abbildung 20: Durch DBSCAN identifizierte Ausreißer	62
Abbildung 21: Struktureller Aufbau eines Variational-Autoencoder-Modells	66

Abbildung 22: Dichtediagramm der augmentierten Daten des gefilterten Datensatzes	70
Abbildung 23: Prozess zur Generierung des gefilterten Datensatzes	71
Abbildung 24: Illustration der Kosinusähnlichkeit zur Datenauswahl	72
Abbildung 25: Klassenverteilung der finalen Datensätze	73
Abbildung 26: Übersicht der Datenmenge in verschiedenen Verarbeitungsschritten... <td>74</td>	74
Abbildung 27: Wireframe der initialen Nutzeroberfläche.....	79
Abbildung 28: Wireframe der Ergebnisseite des Prototyps	80
Abbildung 29: Illustration der identifizierten Entwicklungsebenen.....	81
Abbildung 30: Klassenübersicht zur Datenvorverarbeitung.....	83
Abbildung 31: Konzeptioneller Entwurf der Knowledge Base.....	84
Abbildung 32: Illustration des Vorgehens in der Klassifikation	88
Abbildung 33: Weighted Averaging im Rahmen des Ensemble Learning.....	89
Abbildung 34: Illustration des Vorgehens zur Entitätextraktion	90
Abbildung 35: Klassenstruktur der benötigten Klassen zur Verfahrenszuweisung	92
Abbildung 36: Struktureller Aufbau des Prototyps	93
Abbildung 37: Struktur des implementierten Python-Pakets	93
Abbildung 38: Ausgewählte Spearman-Korrelationen zwischen Modellgenauigkeit und Modelparametern (FastText)	101
Abbildung 39: Scatter Plots des besten FastText Modells.....	102
Abbildung 40: Ausgewählte Spearman-Korrelationen zwischen Modellgenauigkeit und Modelparametern (DAN)	103
Abbildung 41: Deep-Averaging-Netzwerke mit mutipler Layer-Anzahl.....	104
Abbildung 42: Ähnlichkeitsmatrix zur Evaluierung von Embedding-Modellen.....	106
Abbildung 43: Illustration der Evaluationsergebnisse der Keyword Extraction	114
Abbildung 44: Parameterkorrelationen der LDA-Modelle mit n>3	115

Abbildung 45: Parameterkorrelationen der LDA-Modelle mit n=3	116
Abbildung 46: Visualisierung eines LDA-Modells	117
Abbildung 47: Übersicht über die Schlüsselwörter pro Topic	118
Abbildung 48: Darstellung der Startseite des entwickelten Prototyps.....	126

Tabellenverzeichnis

Tabelle 1: Gegenüberstellung von technischen Artikeln und Problemstellungen	7
Tabelle 2: Festlegen der Clustering-Parameter	63
Tabelle 3: Kriterien zur Augmentierung von Daten mittels Substitution	67
Tabelle 4: Exemplarische Substituierung von Synonymen in Sätzen	69
Tabelle 5: Beispiel für generierte Sätze mittels eines Markov-Modells	70
Tabelle 6: Übersicht über die erstellten Datensätze.....	72
Tabelle 7: Prototypbezogene Anforderungsdefinition	76
Tabelle 8: Zuordnung von Vorverarbeitungsmethoden zu Methoden des KB-Aufbaus.	82
Tabelle 9: Gegenüberstellung der Anforderungen	94
Tabelle 10: Übersicht der trainierten Parameterkombinationen für Embedding-Modelle	107
Tabelle 11: Ergebnisübersicht der Ähnlichkeitsevaluation von Embedding-Modellen	108
Tabelle 12: Übersicht der Evaluationsergebnisse von DAN-Modellen	109
Tabelle 13: Evaluationsergebnisse von Topic-Modellen.....	113
Tabelle 14: Tabellarische Übersicht der identifizierten Design-Prinzipien	126
Tabelle 15: Minima und Maxima der Bewertung von Embedding-Modellen	127

Abkürzungsverzeichnis

CNN	Convolutional Neural Net
DAN	Deep-Averaging-Network
ELMO	Embedding from Language Models
GLoVe	Global Vectors
GRU	Gated-Recurrent-Unit
IEEE	Institute of Electrical and Electronics Engineers
KDD	Knowledge Discovery in Databases
LDA	Latent Dirichlet Allocation
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TF-IDF	Term frequency-inverse document frequency

1 Themenbeschreibung und Motivation¹

Das Areal der Data Science (DS) ist ein aufstrebendes Gebiet, das zunehmende Relevanz in Unternehmen erhält. Dabei ist die DS als ein stark interdisziplinäres Feld zu charakterisieren, welches die Kombination von mathematischem, informatischem sowie domänenspezifischem Wissen in der Person des Data Scientists voraussetzt (vgl. Schutt & O'Neil, 2013, S. 5f.). Letzteres zeigt hierbei eine hohe Notwendigkeit in einer der Hauptherausforderungen in Projekten der DS – der Zuordnung von adäquaten Methoden zu einer Problemstellung. Festzustellen ist allerdings, dass heutzutage im Allgemeinen ein Mangel an Data Scientists mit umfangreichem Wissen in allen Unternehmensdomänen herrscht, weshalb Projekte der DS zumeist in Kollaboration mit unterschiedlichen Stakeholdern und Domänenexperten durchgeführt werden (vgl. McAfee & Brynjolfsson, 2012, S. 66; Mikalef, Giannakos, Pappas, & Krogstie, 2018, S. 503f.). Als Unterstützung der Kommunikation zwischen den Personengruppen ordnet das übergeordnete Ziel der vorliegenden Arbeit in der fünften Phase des Knowledge-Discovery-in-Databases-Prozesses nach FAYYAD ET AL. (1996) ein, welche nach vorhergehendem Erlangen von Domänenwissen sowie der Vorverarbeitung der Daten, die Selektion des geeigneten Analysemodells bzw. der geeigneten Analysemethode fokussiert. So wird das Ziel verfolgt diese Selektion auf Basis von natürlichsprachlichen Problemstellungen mittels adäquater Methoden automatisiert zu unterstützen um eine Erleichterung hinsichtlich der initialen Kommunikation zwischen Domänenexperten und Data Scientists zu erreichen.

Folglich unterliegt das Untersuchungsvorhaben einem zweigeteilten Forschungsauftrag. So hat sich der erste Teil, im Rahmen einer vorangegangenen Forschungsarbeit, mit der Erörterung eines Methodensets zur Erreichung dieses Ziels beschäftigt, wohingegen der zweite Teil die Entwicklung eines Prototyps auf Basis der zuvor identifizierten Methoden sowie weiterer Methoden fokussiert, welcher anhand gegebener Problemstellungen potentielle Analysemethoden errechnet. Anschließend wird das Resultat dieser Einordnung durch den Prototyp, inklusive einer exemplarischen Analyse, als Empfehlung an den Nutzer zurückgegeben, um somit Unterstützung für Personen ohne Fachkennt-

nisse der Datenanalyse zu bieten. Anzumerken ist, dass, im Anschluss an die vorangegangene Forschungsarbeit, in der vorliegenden Arbeit der zweite Teil des Forschungsauftrags fokussiert wird.

Konkret existieren unter dem aktuellen Forschungsziel zwei konträre Areale bzw. Personengruppen. Auf der einen Seite finden sich Datenanalysten mit dem notwendigen Wissen über Algorithmen und Analysemethoden sowie deren Einsatzzweck. Weiterhin sind hier natürlichsprachliche, semi-formale bzw. mathematische sowie formale bzw. programmatische Deskriptionen von Algorithmen zu verorten. Auf der anderen Seite sind Personen ohne Kenntnisse im Bereich der Datenanalyse, welche jedoch über umfangreiche Kenntnisse in spezifischen Unternehmensdomänen verfügen und entstehende Problemstellungen der Domäne dokumentieren.

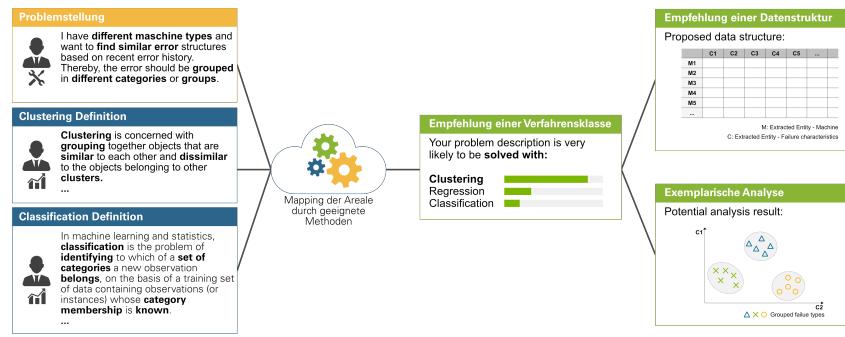


Abbildung 1: Illustration der allgemeinen Zielstellung (Eigene Darstellung)

Als Folge der differenten Vokabulare und Verständnisse der beiden Areale bzw. Personengruppen entsteht die Notwendigkeit diese miteinander in Beziehung zu setzen. Wie in **Abbildung 1** dargestellt, existieren exemplarisch eine Problemstellung eines Experten im Umfeld eines produzierenden Unternehmens sowie Definitionen differenter datenanalytischer Verfahren. Anhand dieser Inputdaten besteht das Ziel darin, mithilfe adäquater Methoden, eine Annäherung der Areale zu erreichen, um als Resultat eine Aussage über die zu verwendende Verfahrensklasse zur Lösung der Problemstellung zu erwirken. Im Kontext der obenstehenden Abbildung wird Clustering als adäquate Methode eruiert. Im Zuge dessen erfolgt weiterhin eine Beschreibung der Struktur der benötigten Daten sowie eine exemplarische Analyse anhand identifizierter Entitäten der Problem-

stellung. Diese Entitäten gilt es hierfür im eingegebenen Text mittels adäquater Methoden zu identifizieren. Durch den resultierenden Informationsgewinn kann einerseits der Prozess der Modellselektion nach FAYYAD ET AL. (1996) für Datenanalysten automatisiert unterstützt werden, um ggf. deren Effizienz zu steigern. Andererseits entsteht hiermit eine Hilfestellung für Fachpersonen, ohne Kenntnisse der Datenanalyse, da eine Beauftragung von Analysen mit adäquaten Methoden und Daten ohne lange Eruierung und Absprache des Problems ermöglicht wird. Generell wird diesen Personen somit eine Empfehlung über die zu verwendende Methodik ausgesprochen und eine notwendige Datenstruktur unterbreitet, welche als Diskussionsgrundlage zwischen den Arealen sowie als Kategorisierungshilfe für Datenanalysten fungieren kann.

1.1 Einordnung, Abgrenzung und Herausforderungen

Bezüglich der o.g. automatisierten Selektion von Analysemethoden existiert in der Literatur eine Vielzahl an Ansätzen, welche unterschiedliche Vorgehen applizieren. SERBAN, VANSCHOREN, KIETZ & BERNSTEIN (2013) geben in ihrer Publikation eine allgemeine Übersicht zu bereits hierfür entwickelten Systemen bzw. Herangehensweisen. So spezifizieren die Autoren verschiedene Kategorien derartiger Systeme.

Zunächst sind hierbei **Experten Systeme** zu nennen, welche auf einer manuell erstellten Wissensbasis, in Form von Regelsets, operieren. Zur Einordnung eines neuen Problems, und somit zur Auswahl der anwendbaren bzw. geltenden Regeln, wird ein Nutzerdialog forciert, in welchem gezielte Informationen anhand des prädefinierten Regelsets erfragt werden. Final wird dem Nutzer in solchen Systemen eine Rangliste an verschiedenen Methoden ausgegeben (vgl. Serban et al., 2013, S. 31:8).

Weiterhin beschreiben SERBAN ET AL. (2013) sog. **Meta-Learning-Systeme**, welche den Zusammenhang zwischen messbaren Informationen des Problems bzw. des Datensets und der Güte von differenten Algorithmen untersuchen. Dabei unterteilen sich derartige Systeme in eine Trainings- und Vorhersagephase, wobei erstere die Extraktion von Metainformationen über Datensets sowie die anschließende Anwendung von Algorithmen auf diese umfasst. Anhand der jeweiligen Ergebnisse wird ein Modell, wie beispielsweise ein Entscheidungsbaum, trainiert, welches zur Evaluierung einer Empfehlung für

weitere Datensets verwendet wird (vgl. Serban et al., 2013, S. 31:9). Metainformationen werden hierbei wie folgt kategorisiert: 1) statistische Daten, 2) modellbasierte Daten und 3) Messdaten (vgl. Brazdil, Carrier, Soares, & Vilalta, 2009, S. 6).

Ferner werden von SERBAN ET AL. (2013) sog. **Case-based-Reasoning-Systeme, Planning-based-Data-Analysis-Systems** sowie **Workflow-Composition-Environments** beschrieben. Aufgrund deren zur aktuellen Zielstellung konträren Fokussierung auf gesamte Workflows und damit auf multiple Phasen des KDD-Prozesses findet keine nähere Erläuterung dieser statt.

Trotz der Unterschiede der genannten Systeme ist eine Parallele zu erkennen, welche auf der Verwendung von Datensets und Informationen über diese beruht. So erstellen diese Systeme einen Variablenraum, welcher aus bestimmten statistischen, modellbasierten und/oder messbasierten Informationen besteht. Weiterhin zielen derartige Systeme auf die Strukturierung der Menge an möglichen Algorithmen der Data Science und deren Einordnung anhand bestimmter Charakteristiken ab. Zudem sind sie zumeist an eine **Zielgruppe aus Data Scientists oder Personen mit ausreichendem Fachwissen im Bereich der Datenanalyse gerichtet**, um dieser Unterstützung zu geben. **(Eventuell QUELLE nötig?)**

Kommentiert [m2]:

Die vorliegende Forschungsarbeit setzt bereits vorher an und verfolgt, wie oben bereits dargelegt, das Ziel fachfremde Personen zu unterstützen, während noch keine expliziten Daten bzw. Datensets vorliegen. Somit greift das zu entwickelnde System bereits in der Ideenphase von Nutzern ein und gibt diesen eine Empfehlung bzgl. der zu verwendenden Verfahrensklasse sowie der hierfür benötigten Datenstruktur. Ebenfalls wird dem Nutzer eine exemplarische Analyse der eruierten Verfahrensklasse, unter Einbezug der zu untersuchenden Entitäten der Problemstellung, ausgegeben. Diese ermöglicht es dem Nutzer festzustellen, ob das errechnete Verfahren konform zu dessen initialer Vorstellung ist. Hierbei wird die Empfehlung alleinig auf Basis, der vom Nutzer übergebenen, natürlichsprachlichen Problemstellung errechnet. Da in der Arbeit eine domänenunabhängige Unterstützung von Personen intendiert wird, ist ebenfalls keine spezifische Terminologie der Problemstellungen vorgegeben bzw. anzunehmen.

In der Folge entsteht die Unterstützung auf einer höheren Granularität, da kein spezifischer Algorithmus, sondern die Verfahrensklasse als Gesamtes empfohlen wird. Diese Information, in Verbindung mit der exemplarischen, personalisierten Analyse sowie der

Datenstruktur, ist als Handlungsempfehlung für den Nutzer anzusehen und dient der Unterstützung in der Kommunikation zwischen Data Scientists und der jeweiligen Fachabteilung. Aufgrund dieses Sachverhaltes sind ebenfalls sog. **Recommender-Systeme** anzusprechen, welche generell das Ziel verfolgen, Entscheidungsunterstützungen für Nutzer in informationskomplexen Arealen zu geben (vgl. Rashid et al., 2002, S. 127).

In Hinblick auf Recommender-Systeme definieren Balabanović & Shoham (1997) drei differente Kategorien. Zunächst deskribieren die Autoren rein **inhaltlich basierte Recommender-Systeme**, welche alleinig auf Basis eines Nutzerprofils und inhaltlichen Informationen der zu empfehlenden Daten arbeiten und Empfehlungen aussprechen. Exemplarisch bedeutet diese Fokussierung auf inhaltliche Faktoren in der Empfehlung von Webseiten, dass lediglich der textuelle Inhalt, nicht aber das Aussehen sowie die Struktur der Website, in die Evaluation einfließen. Die zweite Kategorie bilden **kollaborative Recommender-Systeme**. Diese prozessieren Empfehlungen alleinig mithilfe von zueinander ähnlichen Nutzerprofilen und missachten dadurch inhaltliche Faktoren. Im Kontext des gleichen Beispiels werden somit Webseiten vorgeschlagen, die von anderen Nutzern mit ähnlichen Profilen präferiert werden. Zuletzt erläutern die Autoren **hybride Recommender-Systeme**, die die jeweiligen Vorteile der beiden Systeme kombinieren und somit den jeweiligen Limitationen entgegenwirken (vgl. Balabanović & Shoham, 1997, S. 66ff.).

Das hier zu entwickelnde System ist demnach den inhaltlichen Systemen zuordenbar, da lediglich Aspekte der übergebenen, natürlichsprachlichen Problemstellung beachtet werden und somit keine weiteren Nutzer in den Empfehlungsprozess einbezogen werden. Dies resultiert aus dem Sachverhalt, dass im Rahmen der Untersuchung keine umfangreiche Menge an Nutzern zur Verfügung steht, um kollaborative Aspekte zu integrieren. Anzumerken ist allerdings, dass eine Beachtung von kollaborativen Aspekten durch die Verwendung historischer Nutzereingaben im System erreicht werden kann, indem bei der Evaluierung neuer Problembeschreibungen gewisse Zielkategorien bzw. Data-Mining-Verfahrensklassen durch Ähnlichkeiten zu vergangenen Anfragen ausgeschlossen werden könnten. Gleichzeitig ist an dieser Stelle eine Restriktion des aktuellen Vorhabens zu identifizieren, da nur wissenschaftliche Publikationstexte und eine sehr geringe Anzahl an Problembeschreibungen als initiale Datenbasis zur Verfügung stehen. Dazu sei auf **Kapitel XX** verwiesen, in welchem dieser Sachverhalt adressiert und der Aufbau der Datenbasis aufzeigt wird.

Neben der Einordnung des zu implementierenden Prototyps ist ebenfalls eine Einschränkung hinsichtlich der Zielklassen und somit der potentiellen Verfahrensklassen, denen Problemstellungen zugewiesen werden können, zu erwähnen. Im Rahmen der aktuellen Untersuchung und der einhergehenden Entwicklung wird sich auf insgesamt drei Verfahrensklassen beschränkt. Primär wird sich zur Auswahl dieser an der Kategorisierung nach TSAI, LAI, CHIANG & YANG (2014) orientiert, die die Verfahrensklassen *Clustering*, *Classification*, *Sequential Pattern* sowie *Association Rules* aufzeigen. Weiterführend erfolgt eine Adaption dieser Einteilung, bei welcher das Areal der *Klassifikation* mit dem Verfahren der *Regression* zur Klasse *Prediction*, und die Kategorien *Sequential Pattern* und *Association Rules* zum Gebiet des *Frequent Pattern Mining* vereint werden. Diese Vereinigung wird vorgenommen, da die beiden Areale als Subareale des Frequent Pattern Mining anzusehen sind (vgl. Aggarwal, 2014, S. 1f.). Final werden demnach in der vorliegenden Forschungsarbeit die Zielklassen *Clustering*, *Prediction* sowie *Frequent Pattern Mining* fokussiert, welchen neue Problemstellungen zugeordnet werden.

Linguistische Herausforderung der Daten

Wie aus dem vorherigen Kapitel hervorgeht, sind die Eingabeparameter des zu entwickelten Prototyps klar definiert. Dem System werden durch einen Domänenexperten natürlichsprachliche Problemstellungen zugeführt, und dieses bestimmt unabhängig von der domänenspezifischen Terminologie eine geeignete Verfahrensklasse.

Eine der großen Herausforderungen besteht in der Verarbeitung zweier unterschiedlicher Domänen. So ist das Areal der Data Science durch eine eigene fachspezifische Terminologie gezeichnet, die umfassendes Wissen von mathematischen, statistischen sowie informatischen Zusammenhängen voraussetzt. Für das Verständnis der zugrundeliegenden Daten benötigt der Data Scientist jedoch ebenfalls umfassendes Domänenwissen (vgl. Schutt & O'Neil, 2013, S. 5f.). Konträr beherrscht die fachfremde Person i.d.R. die benötigte Domänenexpertise, besitzt allerdings kein umfassendes Wissen über Datenanalyseverfahren. Bedingt durch die jeweiligen fach- bzw. domänenspezifischen Terminologien, ist davon auszugehen, dass Problemstellungen different von beiden Personengruppen definiert werden.

Wie aus den obenstehenden Restriktionen hervorgeht, bilden wissenschaftlich, technische Artikel die einzige zur Verfügung stehende Datengrundlage, um den Prototypen

Kommentiert [m3]:

des Recommender-Systems zu implementieren. Im Vergleich zum Eingabetext der fachfremden Person besitzen derartige Artikel andere linguistische Eigenschaften. Technische Artikel setzen häufig domänenspezifisches Wissen und die Kenntnis der jeweiligen Terminologie voraus und bedienen sich komplexen wissenschaftlichen Konstrukten (**Quelle**). Dabei bestehen die kommunikative Funktion und die Intention von technischen Artikeln im Übermitteln von Wissen in Form von abstrakten Konstrukten, Konzepten, Prinzipien, um Lösungen für konkrete Probleme zu erreichen (**Quelle**). Hinzukommt, dass technische Artikel wenig Spielraum für semantische Redundanzen lassen und Informationen strukturiert im Text wiederzufinden sind (vgl. Rinaldi, Hess, Dowdall, Mollá, & Schwitter, 2004, S. 73). Konträr dazu ist die wesentliche Intention von Problembeschreibungen eine umfangreiche Darlegung von Problemen, wobei sie keiner klaren Form unterliegen. Ferner sind die von der Domäne geprägten, terminologischen Bausteine unterschiedlich zu denen aus wissenschaftlichen Artikeln. Dabei beziehen sich terminologische Bausteine nicht nur auf Wörter, sondern bestimmen auch die Syntax und Satzstruktur von natürlicher Sprache (vgl. Faber, 2012, S. 10). **Tabelle XX** gibt einen Überblick über die Unterschiede der zugrundeliegenden Daten des zu entwickelten Prototyps.

Kommentiert [m4]: Alleinig die Intention der Inputdaten unterscheidet sich stark von den Trainingsdaten.

Kommentiert [m5]:

Tabelle 1: Gegenüberstellung von technischen Artikeln und Problemstellungen

Kriterium	Technische Artikel	Problemstellung
<i>Verwendung</i>	Trainingsdaten	Einzuordnende Daten
<i>Struktur</i>	Strukturiert	Unstrukturiert
<i>Intention</i>	Lösen von Problem	Beschreiben von Problemen
<i>Terminologie</i>	Areal der Data Science	Domänenspezifisch

In Anlehnung an IYYER ET AL. (2015) wird die Diskrepanz zwischen Inputdaten und Trainingsdaten im Folgenden als syntaktische Divergenz verstanden. Das Konzept der syntaktischen Divergenz inkludiert somit domänenspezifische sowie syntaktische Diskrepanzen der beiden Datenpools. Unterscheiden sich natürlichsprachliche Texte grundlegend in ihrer Bedeutung, wird im Folgenden von semantischer Divergenz gesprochen. Die Herausforderung der vorliegenden Arbeit besteht darin auf Basis von syntaktisch

divergenten Daten ein System zu entwickeln, das syntaktische Divergenz überwindet und semantische Ausreißer extrahiert.

1.2 Forschungsvorhaben

Im folgenden Kapitel wird das zugrundeliegende Forschungsvorhaben deskribiert. Auf Basis der vorangegangenen Forschungsarbeit stehen hierbei die Untersuchung weiterer Methoden, der Prozess zur Datenbeschaffung, die Evaluierung der Methoden und des Systems sowie die tatsächliche Implementierung des Recommender-Systems im Vordergrund.

1.2.1 Forschungsdesign¹

Vor dem Hintergrund der aufgezeigten Zielstellung ist ein Erkenntnisziel sowie ein Gestaltungsziel zu differenzieren. BECKER ET AL. (2004) definieren das Erkenntnisziel als den Wunsch, Verständnis gegenüber einem bestimmten Sachverhalt zu erlangen. Konträr dazu stehen Gestaltungsziele, welche die Konzeption und Veränderung bestehender und neuer Sachverhalte beschreiben. Dabei bedienen sich Gestaltungsziele häufig an den Ergebnissen von Erkenntniszielen (vgl. Becker, Niehaves, & Knackstedt, 2004, S. 11f.).

Das Erkenntnisziel, der nachfolgenden Arbeit definiert sich folgendermaßen:

Es soll Erkenntnis darüber erlangt werden, welche Anforderungen an die Implementation eines textbasierten Recommender-Systems existieren und wie ein solches System in seiner Gesamtheit sowie dessen einzelne Methoden evaluiert werden können. Weiterhin soll Erkenntnis darüber erlangt werden, wie natürlichsprachliche Daten mithilfe adäquater Methoden expandiert werden können.

Das beschriebene Erkenntnisziel bewegt sich somit im Rahmen eines methodischen Auftrags. Das Gestaltungsziel der Arbeit ist hingegen wie folgt zu formulieren:

In Rückschluss auf die Ergebnisse des zuvor definierten Erkenntnisziels wird zunächst ein Prozess zur Datenbeschaffung definiert, um anschließend einen Prototyp eines solchen Systems mithilfe adäquater Methoden zu entwickeln und diesen hinsichtlich seiner verwendeten Methoden sowie in seiner Gesamtheit zu evaluieren.

Die vorliegende Arbeit basiert auf der Grundposition des wissenschaftlichen Konstruktivismus, der sich auf ein subjektgebundenes Empfinden der Realität stützt und damit gleichzeitig impliziert, dass die daraus resultierende Erkenntnis ebenfalls nur subjektiver Natur sein kann (Becker, Holten, Knackstedt, & Niehaves, 2003). Dabei wird bei der Forschungsmethode hauptsächlich ein deduktives Vorgehen gewählt.

Kommentiert [m6]: Ist jetzt mies aber: Seite? :D
Kommentiert [DH7R6]: 18!

1.2.2 Forschungsfragen²

Basierend auf den definierten Forschungszielen lassen sich die folgenden Forschungsfragen identifizieren, die das Gesamtziel der Arbeit charakterisieren und somit einen Leitfaden darstellen:

Forschungsfrage I:

Wie lässt sich eine natürlichsprachliche Datenbasis erstellen und expandieren, um die Implementierung des fokussierten Prototyps zu ermöglichen?

Forschungsfrage II:

Welche Anforderungen existieren an ein textbasiertes Recommender-System und wie lässt sich ein Prototyp eines solchen Systems entwickeln?

Forschungsfrage III:

Wie lassen sich die im Rahmen der Arbeit identifizierten Methoden sowie das zu implementierende Gesamtsystem vor dem Hintergrund der aktuellen Zielstellung evaluieren?

1.3 Methodisches Vorgehen

Das folgende Kapitel beschäftigt sich mit dem methodischen Vorgehen, das der Durchführung des Forschungsvorhabens zugrunde liegt. Hierbei wird zunächst auf die Methodik der systematischen Literaturrecherche eingegangen. Im Anschluss hieran wird der Design-Science-Research-Prozess zur Strukturierung des Vorhabens mit dem Vorgehen zur Anforderungspriorisierung sowie zum Prototyping kombiniert.

1.3.1 Systematische Literaturrecherche

Zunächst wird zur Evaluation relevanter Literatur die Forschungsmethode der systematischen Literaturrecherche nach FETTKE (2006) angewandt, welche „[...] aus der Perspektive einer bestimmten Fragestellung die zu einem Themengebiet relevanten Arbeiten und vorliegenden Erkenntnisse“ (Fettke, 2006, S. 258) untersucht. Diese Methode erlaubt es, anschließend an die im Rahmen der vorhergegangenen Forschung, additive Methoden und Herangehensweisen zu identifizieren und diese potentiell in der Programmierung des deskribierten Systems zu berücksichtigen. Grundsätzlich werden hierfür mehrere fachspezifische sowie -übergreifende Datenbanken nach Stichwörtern durchsucht, wobei allgemeingefasste Suchbegriffe sukzessiv konkretisiert werden, um bestmögliche Resultate zu erzielen. Dies kann anhand von logischen Operatoren wie „AND“ oder „OR“ geschehen, wie auch mittels Trunkierungen und Festlegung von starken Wortfolgen durch Einschließen dieser in Anführungszeichen. Wie in **Abbildung 2** exemplarisch illustriert, werden hierfür zunächst, aufbauend auf Ergebnissen der vorhergegangenen Forschungsarbeit, differente Term-Sets definiert, welche Recommender-Systeme mit den bereits identifizierten Forschungsarealen in Verbindung setzen. Mit Hilfe der hierdurch gefundenen weiteren Methoden werden additionale Sets definiert, welche über „AND“ und „OR“ miteinander verknüpft werden und somit die Gesamtheit der Suchbegriffe darstellen. Die differenten Sets werden, für bestmögliche Resultate, mit unterschiedlichem Detailgrad aufgestellt. Anzumerken ist hier, dass der Fokus aufgrund der Menge von zu untersuchenden Gebieten sowie deren jeweiligen Umfang lediglich auf die Top-Ergebnissen pro Suchanfrage restriktiert wird.

Kommentiert [m8]: Ist das wirklich sinnvoll oder sollte man die lieber raus lassen oder wie zeigen wir da jetzt irgendwie auf warum wir die machen, nach was wir da suchen und so weiter?

Man könnte auch sagen man macht damit nochmal Grundlagensuche und sowas und sucht nach State-of-the-Art Methoden und so aber da muss dann halt auch irgendwas stehen...

Kommentiert [m9]: Idee:
(1) Quellenauswahl baut auf Literaturrecherche nach Fettke der vorangegangenen Forschungsarbeit auf.
→ Hierbei werden differente Datenbanken (ArXiv, Science Direct, IEEE, Scholar) nach speziellen Suchbegriffen durchsucht um weiterführend Termssets aufzubauen
→ Im Anschluss werden dann diese Termssets untersucht um weitere potentielle Methoden zu finden
→ Zur Übersicht sei das nochmal dargestellt aber nicht weiter deskribiert oder erläutert. Dafür sei auf die vorangegangene Arbeit verwiesen.
(2) Vorangegangene Arbeit identifiziert Methodengruppen, welche hier noch weiter untersucht werden
(3) Dafür wird vor allem das Verfahren Backward reference search benutzt
→ Erlaubt einerseits zu erkennen wie sich Wissen zusammensetzt und wie die Abhängigkeiten zwischen Konstrukturen aussehen
→ Weiterhin können Originalquellen identifiziert werden
→ Auch wird relevante Literatur durch vermehrte Verweise auf diese gefunden
→ Experten können gefunden werden
(4) ?!

Keine Ahnung ob das doof kommt oder methodisch komplett kacke
→ Nochmal drüber quatschen vielleicht fällt uns zusammen noch was ein

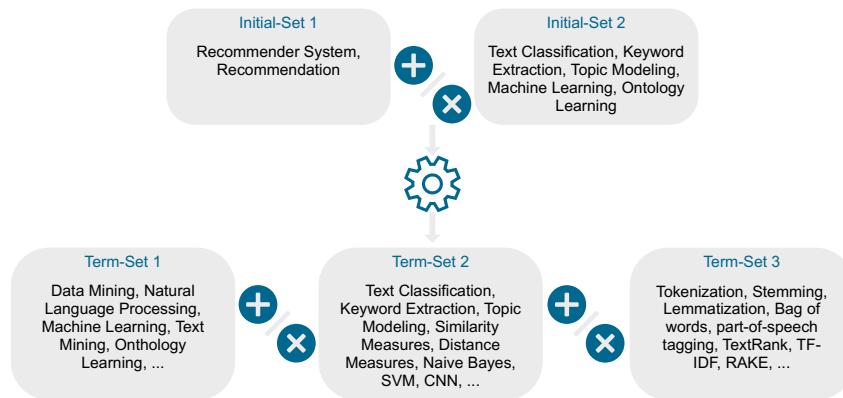


Abbildung 2: Illustration der aufgestellten Term-Sets zur Literaturrecherche (Eigene Darstellung)

1.3.2 Der Design-Science-Research-Prozess

Im Rahmen des aktuellen Forschungsvorhabens sowie in Hinsicht auf die Entwicklung eines Prototyps wird sich an dem Design-Science-Research-Prozess (DSP) nach PEFERS, TUUNANEN, GENGLER, ROSSI, & HUI (2006) orientiert. Die *Design Science* beschreibt ein wichtiges Paradigma im Areal der *Information Systems Research* und fokussiert im Allgemeinen die Entwicklung von sozio-technischen Artefakten, die indes weit gefasst sind und nach MARCH & SMITH (1995) Konstrukte, Methoden, Modelle sowie Instanzierungen umfassen. Hierbei zentrieren diese Artefakte die Leistung eines Beitrags zur Lösung einer relevanten Problemstellung aus der Wissenschaft oder Praxis und manifestieren sich somit als prototypische Entwicklungen und nicht als komplettierte Systeme. Als finales Resultat einer Untersuchung in der Design Science spezifizieren HEVNER, MARCH, PARK & RAM (2004) einen allgemeinen Forschungsbeitrag, welcher unterschiedlich charakterisiert sein kann. Zumeist handelt es sich hierbei um das entwickelte Artefakt per se, welches zur Lösung der definierten Problemstellung verwendet wird, wohingegen ebenfalls neu entwickelte und evaluierte Konstrukte und Methoden zur Erweiterung der Wissensbasis oder die Entwicklung neuer Evaluationsmetriken einen möglichen Beitrag darstellen. Additional ist das Ziel Design Prinzipien hinsichtlich verschiedener Ebenen der Entwicklung zu identifizieren (vgl. Gregor & Hevner, 2013, S. 341ff.; Hevner et al., 2004, S. 78ff.)

PEFFERS ET AL. (2006) definieren in ihrer Forschungsarbeit sechs differente Phasen, die in **Abbildung XX** schematisch illustriert werden. Dieser Prozess bildet dabei einen generellen Rahmen für die Forschung im Bereich von Informationssystemen.

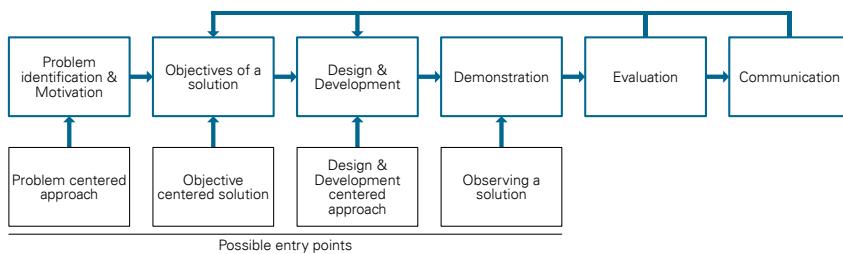


Abbildung 3: Der Design-Science-Prozess nach PEFFERS ET AL. (2006)

Zunächst definieren die Autoren die Phase **Problemidentifikation und Motivation**, in der die zugrundeliegende Problem- und Zielstellung dargelegt wird. Zeitgleich erfolgt hier die Darlegung der Motivation sowie des Nutzens einer Problemlösung (vgl. **Kapitel XX**) (vgl. Peffers et al., 2006, S. 89).

In der darauffolgenden Phase werden die **Ziele der zu erstellenden Lösung definiert**. Hierbei werden Aspekte festgehalten, welche das zukünftige Artefakt erfüllen muss, um die Lösung der zuvor aufgestellten Problemstellung herbeizuführen. Generell können die aufgestellten Ziele qualitativer sowie quantitativer Natur sein (vgl. Peffers et al., 2006, S. 90). Im Rahmen der aktuellen Untersuchung ist dieser Prozessschritt in zwei Kategorien zu dividieren, wobei auf der einen Seite globalere Ziele des Gesamtsystems fokussiert werden. Auf der anderen Seite werden Anforderungen an das System bzw. dessen Komponenten gestellt. Demzufolge ist hierbei eine Kombination aus Zielen und Anforderungen, welche das Gesamtsystem sowie das zu entwickelnde Artefakt betreffen auszuarbeiten.

Anschließend erfolgt in der Phase **Design und Entwicklung** die eigentliche Entwicklung des Artefakts. Wie bereits eingangs dargelegt ist der Begriff eines Artefakts ist hier sehr breit aufgestellt und umfasst Konstrukte, Modelle, Methoden und Instanziierungen (vgl. Hevner et al., 2004, S. 78). Im Rahmen der vorliegenden Forschungsarbeit handelt es sich hierbei um einen Prototypen eines Recommender-Systems, welcher mithilfe einer

adaptierten Variante des Prototyping-Vorgehensmodells nach PRESSMAN (2015) entwickelt wird. Somit findet eine Integration dieses Vorgehensmodell in aktuelle Phase des DSP statt.

PRESSMAN (2015) definiert in seinem Modell fünf differente Phasen eines Zyklus, welche in ihrer Gesamtheit den Prozess der Softwareentwicklung wiederspiegeln (siehe **Abbildung XX**). Zunächst spezifiziert der Autor das Eruieren und Dokumentieren von Anforderungen. Anzumerken ist, dass hier eine Überschneidung mit der zweiten Phase des DSP vorliegt. Resultierend wird hier die Festlegung von systemseitigen Anforderungen fokussiert, wohingegen die globalen Ziele der Untersuchung im Rahmen der allgemeinen Zielstellung des DSP definiert werden. Im Sinne einer Differenzierung der prototypbezogenen Anforderungen erfolgt eine Unterteilung in funktionale sowie nicht-funktionale Anforderungen. Erstere beschreiben im Allgemeinen alle Aktionen, die das System bzw. das Artefakt ausführen können muss, um den Nutzen und somit den Beitrag zur Lösung der Problemstellung zu gewährleisten. Demzufolge ist ein großer Teil dieser Anforderungen aus der intendierten Nutzung ableitbar. Nicht-Funktionale Anforderungen hingegen repräsentieren Eigenschaften und Qualitätsattribute des Systems, wie beispielsweise dessen Performanz. Generell sind diese Anforderungen direkt an der Akzeptanz des Systems gekoppelt, woraus sich eine essentielle Relevanz dieser ergibt (vgl. Suzanne Robertson & Robertson, 2012, S. 9 f.). Anschließend wird eine Priorisierung aller Anforderungen nach IEEE STD 830-1998 (1998) angewendet, um die jeweiligen Anforderungen in die Kategorien „*essential*“, „*conditional*“ sowie „*optional*“ zu unterteilen. Erstere Gruppe charakterisiert Anforderungen, welche essentiell notwendig für den Erfolg der Untersuchung sind. Konditionale Anforderungen hingegen beschreiben jene, welche zusätzlichen Mehrwert für das Vorhaben erwirken, allerdings keine negativen Auswirkungen bei Fehlen aufweisen. Abschließend beinhaltet die Kategorie „*optional*“ Anforderungen, die eventuelle Vorteile versprechen, allerdings diese nicht definitiv hervorbringen und somit ebenfalls keine essentielle Notwendigkeit darstellen (vgl. IEEE Std 830-1998, 1998, S. 7).

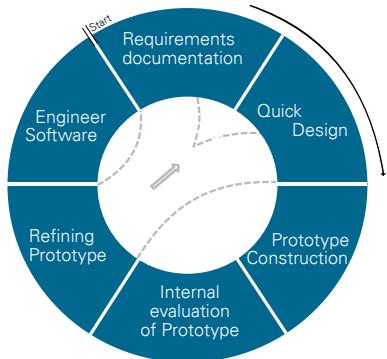


Abbildung 4: Vorgehensmodell des Prototyping in Anlehnung an PRESSMAN (2015)

Die somit aufgestellten Anforderungen bilden weiterhin das Fundament zur Erstellung des ersten groben Entwurfs des Prototyps, welcher zunächst auf sichtbare Bereiche abzielt und somit Bezug auf das User Interface sowie dessen grobe Struktur nimmt. In Phase drei des Zyklus folgt darauf aufbauend die Entwicklung des Prototyps, um diesbezüglich Feedback des Kunden zu erlangen. Hierbei ist anzumerken, dass aufgrund der Inexistenz eines realen Kunden im Rahmen der vorliegenden Arbeit eine interne Evaluation des programmierten Prototyps stattfindet. Das Feedback eröffnet wiederum die Möglichkeit Anpassungen und Präzisierungen des Prototyps im Sinne des Interfaces sowie des programmierten Systems vorzunehmen und bietet somit eine Rückkopplung im sequentiellen Prozessablauf. Im Sinne einer höheren Agilität des Vorgehens, wird diese Rückkopplung auf die Phase der Anforderungsdefinition erweitert, um potentiellen Anforderungsänderungen während der Entwicklung gerecht zu werden. Generell treten jene Anpassungen vor allem in frühen Stadien der Softwareentwicklung ein. Abschließend erfolgt bei kritikloser Review die Phase der eigentlichen Systementwicklung, welche das fertige Softwareprodukt in seiner ersten Version als Resultat hervorbringt (vgl. Pressman, 2015, S. 51f.).

Weiterführend wird das Prinzip des Refactoring in der Implementierung verfolgt, das die kontinuierliche Anpassung der internen Programmstruktur bei gleichbleibender Funktionalität beschreibt und somit das Ziel verfolgt die Lesbarkeit, Wartbarkeit sowie Erweiterungsfähigkeit des Programmcodes sicherzustellen (vgl. Fowler, 2018, S. 9). Die Anwen-

dung dieses Prinzips resultiert vor allem aus den nicht allumfänglich im Vorfeld definierbaren Anforderungen und der dadurch entstehenden Erweiterung bzw. Anpassung von Anforderungen über den Prozess der Softwareentwicklung hinweg, durch welche strukturelle Änderungen des Programms auftreten können.

Nach erfolgreicher Implementierung des Artefakts bzw. des Prototyps wird in der nächsten Phase des Design-Science-Prozesses eine **Demonstration** dieses angestrebt, welche aufzeigt, wie jener zur Lösung des initial definierten Problems Verwendung finden kann. Die Autoren PEFFERS ET AL. (2006) verweisen hierbei auf die Vielfältigkeit an Möglichkeiten zur Durchführung dieser Phase. So kann diese durch eine Fallstudie, ein Experiment, eine Simulation oder eine für das Problem adäquate Demonstrationsart realisiert werden, um den Nutzen sowie die Lösung des Problems zu illustrieren (vgl. Peffers et al., 2006, S. 90).

In der Phase der **Evaluation** wird das Artefakt hinsichtlich der in Phase zwei definierten Ziele untersucht. Hierbei ist eine starke Abhängigkeit zur Art des zugrundeliegenden Problems zu erkennen, da diese Phase Kenntnisse über potentielle Kennzahlen und Methoden zur adäquaten Systemevaluation erfordert (vgl. Peffers et al., 2006, S. 90). In Referenz zum angestrebten Forschungsziel ist zu konstatieren, dass diese Evaluation in messbare und nicht-messbare Aspekte zu unterteilen ist. So ist wie unter **Kapitel XX** bereits dargelegt, exemplarisch die Vereinfachung der Kommunikation zwischen den beiden existenten Personengruppen in Form einer gemeinsamen Kommunikationsbasis als Ziel zu erkennen. Dieses ist allerdings aufgrund des theoretischen Rahmens der Arbeit nicht quantitativ messbar. Gleichwohl findet sich, abgeleitet aus der Kommunikationsbasis, das Ziel der bestmöglichen Zuweisung von etwaigen Datenanalyseproblemstellungen, die mit differenten Metriken der Data Science analysiert und gemessen werden kann.

Letztlich erfolgt die **Kommunikation** des fertiggestellten Artefakts inklusive seines Nutzens, seines Designs und seiner Anwendung. Hierbei ist die Zielgruppe der Kommunikation geprägt von Forschern sowie weiteren potentiell interessierten Personen oder -gruppen. Generell ist dieser Prozess nicht als streng sequentiell anzusehen. Wie in **Abbildung XX** zu erkennen, gibt es mehrere Rückkopplungen zwischen den Phasen, welche eine Anpassung der Zielstellung oder Verfeinerung des erstellten Artefakts ermöglichen. So besteht die Option im Anschluss an die Phasen der Evaluation und Kommunikation eine erneute Iteration beginnend mit der Zieldefinition oder der Entwicklung zu

erwirken, um potentielle Mängel zu beheben oder Feedback aus ersten Phasen zu implementieren. Zur Vollständigkeit sei darauf verwiesen, dass der Prozess, in Abhängigkeit der Problemstellung, differente Startpunkte aufweisen kann. Beispielsweise können Untersuchungen, welche bereits fertiggestellte Artefakte bewerten, direkt in der Demonstrationsphase beginnen (vgl. Peffers et al., 2006, S. 90).

1.4 Aufbau und Struktur der Arbeit¹

Übergeordnet folgt die vorliegende Arbeit der in **Abbildung 5** illustrierten Struktur. Während die obigen Kapitel die ersten Phasen des DSP fokussieren und somit die allgemeine Problem- und Zieldefinition umfassen, gliedert sich die weitere Arbeit wie folgt.

Zunächst erfolgt eine Reflexion der vorangegangenen Forschungsarbeit, die die allgemeine Identifikation von potentiellen Methodensets erörtert und somit eine Basis für die aktuelle Arbeit bildet. Aufbauend hierauf wird der aktuelle Forschungsstand hinsichtlich der Art des abgezielten Systems sowie definierter Methodengruppen aufgezeigt. Anschließend werden in **Kapitel XX**, zur Beantwortung der ersten forschungsleitenden Frage, der Prozess zur Datenbeschaffung deskribiert sowie die hierfür verwendeten Methoden eruiert.



Abbildung 5: Struktureller Aufbau der Arbeit (Eigene Darstellung)

Anhand der in diesen Abschnitten erlangten Erkenntnisse werden unter **Kapitel XX** weiterführende Methoden aufgezeigt, welche eine Relevanz für die Untersuchung aufweisen. Hinsichtlich der bereits angesprochenen Entwicklung des Artefakts, und somit der Phase *Design und Development* im DSP, werden weiterhin systemseitige Anforderungen dokumentiert, welche es im Rahmen der Umsetzung zu beachten gilt. Zusätzlich legt **Kapitel XX** den Fokus auf die Entwicklung des Prototyps, um diesen sowie dessen Methoden darauffolgend in **Kapitel XX** zu evaluieren und zu präsentieren. Abschließend wird in den **Kapiteln XX und XX** das ausgewählte Vorgehen kritisch beleuchtet und ein allgemeines Fazit sowie ein Ausblick gegeben.

2 Überblick des Forschungsseminars

Im Einklang mit der in **Kapitel XYZ** beschriebenen Zielstellung beschäftigt sich die vorangegangene Forschungsarbeit damit, adäquate Methoden zur Erreichung dieser zu identifizieren. Dabei ist sich auf die grundlegenden Methodengruppen der tangierenden Forschungsareale beschränkt worden. Des Weiteren sind alle Methoden mit identifizierten Anforderungen an ein derartiges System abgeglichen worden, um darauf aufbauend verschiedene Lösungswege zu beschreiben. Im Folgenden wird ein kurzer Überblick zu den Erkenntnissen der vorangegangenen Arbeit aufgezeigt.

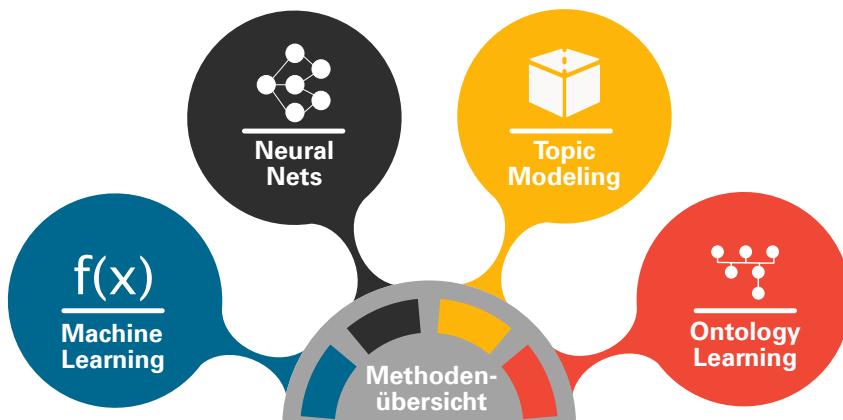


Abbildung 6: Übersicht der methodischen Kategorien (Eigene Darstellung)

Machine Learning

Aufgrund dessen, dass sich die Hauptaufgabe der Zielstellung auf ein Klassifizierungsproblem auf Basis von Textdaten reduzieren lässt, bieten sich eine Vielzahl von Klassifizierungsverfahren zur Erreichung jener an. In der vorangegangenen Forschungsarbeit sind das Naive-Bayes-Verfahren, die logistische Regression sowie Support-Vector-Machines als potentielle Verfahren identifiziert und im Zusammenhang mit Textdaten beschrieben worden. Dabei hat sich herausgestellt, dass diese Verfahren essentiell sind, um

im Zusammenhang mit nichtklassifizierenden Methoden die gegebene Zielstellung zu erreichen. Zugehörig sind ausgewählte Deep-Learning-Verfahren zur Textklassifikation in einem additionalen Kapitel erläutert.

Neuronale Netze

Als Teilgebiet des Machine-Learning etabliert sich das Forschungsareal der neuronalen Netze (vgl. Goodfellow, Bengio, & Courville, 2016, S. 9). Besonders in Bezug auf Textdaten ist dieses Gebiet vielseitig einsetzbar. Zum einen sind in der Arbeit die fundamentalen Deep-Learning-Verfahren zur Textklassifikation charakterisiert und zum anderen sind das grundlegende Prinzip sowie Verfahren von distribuierten Repräsentationen (engl. *distributed representations/Embeddings*) dargelegt. Generell ist festzuhalten, dass Deep-Learning-Verfahren State-of-the-Art-Ergebnisse erzielen, jedoch komplexer als klassische Machine-Learning-Modelle sind und i.d.R. eine größere Datengrundlage benötigen. Daneben haben sich Modelle zur Berechnung von distribuierten Textrepräsentation etabliert, welche deutliche Verbesserungen in Bezug auf verschiedene Anwendungen des *Natural Language Processing* (NLP) aufzeigen (vgl. Perone, Silveira, & Paula, 2018a, S. 1). Dabei überführen diese Methoden Textdaten in dicht besetzte Vektoren und verinnerlichen, beruhend auf der distributionellen Hypothese, die Bedeutung eines Wortes auf Basis des zugrundeliegenden Kontexts (vgl. Mikolov, Chen, Corrado, & Dean, 2013, S. 2). Als Pionierarbeit können hier die Embedding-Modelle von MIKOLOV ET AL. (2013) angeführt werden, die einen fundamentalen Baustein für aktuelle Modelle der distributionellen Semantik legen. Nichtsdestotrotz haben sich in den letzten Jahren ausgereiferte Modelle etabliert, welche eine Verbesserung zu den von MIKOLOV ET AL. (2013) vorgestellten Ausgangsmodellen darstellen (vgl. Perone, Silveira, & Paula, 2018b, S. 2ff.). Neben dem reinen Überführen von Wörtern in distribuierte Repräsentationen etablieren sich auch Modelle, um Paragraphrepräsentationen zu berechnen. Ein Beispiel bietet das Doc2Vec Modell, das auf Basis der Word2Vec-Modelle dicht besetzte Vektorrepräsentationen einzelner Paragraphen bzw. Dokumente errechnet (vgl. Le & Mikolov, 2014, S. 1188ff.). Jedoch müssen ungesehene Dokumente bzw. Paragraphen neu angeleert werden, weshalb sich eine Verwendung des Modells als diffizil erweist. Allerdings haben sich durch den Erfolg distribuierter Repräsentation neue Paragraph-Embedding-Methoden entwickelt, welche die Semantik einzelner Paragraphen besser inkludieren (vgl. Perone et al., 2018b, S. 9). Diese Modelle zeigen besonders hinsichtlich geschlossener Klassifikationsaufgaben, starke Optimierungen.

Topic Modeling

Die Modelle des Topic-Modeling versuchen aus großen Sammlungen von Textdaten bzw. Dokumenten semantisch relevante Informationen in Form von Themen (engl.: Topics) zu extrahieren (vgl. Todor, Lukasiewicz, Athan, & Paschke, 2016, S. 735). Dabei ist in der vorangegangenen Forschungsarbeit, als populärstes Vorgehen das Latent-Dirichlet-Allocation-Modell (LDA) beschrieben, das jedes Topic als eine Wahrscheinlichkeitsverteilung über Wörter der Dokumente charakterisiert. Im Vergleich zu Keyword-Extraction-Verfahren wird hierbei die Anzahl der zu findenden Topics initial festgelegt. Ähnlich zu Topic-Modeling-Modellen filtern auch Keyword-Extraction-Verfahren die semantisch relevanten Wörter aus einem Dokument heraus und sortieren diese anhand ihrer Relevanz (vgl. Biswas, Bordoloi, & Shreya, 2018, S. 51). Die vorangegangene Untersuchung zeigt auf, dass Keyword-Extraction-Methoden nicht alleinig ausreichen, um die Zielstellung zu erreichen, da sie i.d.R. nur schwache Ontologien extrahieren und im Zusammenspiel mit anderen Klassifikationsverfahren erfolgen müssen. LDA hingegen kann, unter der Voraussetzung, dass die extrahierten Topics die Zielklassen wiederspiegeln, indirekt als Klassifikationsalgorithmus angewandt werden, da die Topic-Verteilung eines ungewöhnlichen Dokuments extrahiert und somit auf dessen Klasse geschlossen werden kann.

Kommentiert [DH10]: Irgendwie mit rein bringen dass hier nur LDA weiter gemacht wird und kein weitere Methode beachtet wird

Ontology Learning

Neben den eben genannten Forschungsarealen gibt es des Weiteren das Gebiet des Ontology Learning. Dabei ist Ontology Learning ein wachsendes Forschungsgebiet, welches auf Basis von NLP-Methoden, Machine-Learning-Verfahren sowie Data Mining das automatisierte Auffinden von Ontologien ermöglicht (vgl. Wong, Liu, & Bennamoun, 2012, S. 1f.). Allerdings zeigt die vorangegangene Arbeit auf, dass das reine Entdecken von Ontologien der angestrebten Zielstellung nicht genügt. Dies resultiert zum einen aus der Kongruenz der Methoden zur Extraktion von leichtgewichtigen Ontologien mit den bereits deskribierten, zum anderen aus der eingangs festgelegten Domänenunabhängigkeit des zu entwickelnden Systems und der folgenden Inexistenz einer speziellen Terminologie. So besteht kein begrenztes Vokabular, dessen Abhängigkeiten und semantischen Beziehungen in einer Ontologie abgebildet werden können. Aus diesen Gründen wird das Gebiet des Ontology Learning im Kontext der vorliegenden Forschungsarbeit nicht berücksichtigt.

3 Forschungsstand

Wie unter **Kapitel XX** bereits dargelegt, verfolgt der zu entwickelnde Prototyp das Ziel der Entscheidungsunterstützung in der Modellselektion im Rahmen des KDD. Das referenzierte Kapitel deskribiert diesbezüglich bereits existente Architekturen nach SERBAN ET AL. (2013), zu welchen nachfolgend konkret Anwendungsfälle aus der Literatur dargelegt werden.

Hinsichtlich der Architektur von **Experten-Systemen** ist als erster Vertreter der *Machine Learning Toolbox Consultant* nach GRANER ET AL. (1993) bzw. dessen Weiterentwicklung nach SLEEMAN, RISSAKIS, CRAW, GRANER & SHARMA (1995) zu nennen, welcher ca. 250 Regeln zur Auswahl passender Algorithmen umfasst und sowohl vor, wie auch nach der Anwendung der identifizierten Methoden einen Nutzerdialog verwendet. Ersterer fokussiert inhaltliche Informationen über die vorhandenen Daten und dient zur initialen Identifikation des Algorithmus, wohingegen letzterer Zufriedenheitsfaktoren abfragt und zur Anpassung der Modellparameter herangezogen wird (vgl. Serban et al., 2013, S. 31:9; Sleeman et al., 1995, S. 55ff.). Weiterhin zeigen DANUBIANU & SOCACIU (2011) die Implementierung eines Experten-Systems zur Einordnung von Data Mining Methoden und definieren hierfür ebenfalls einen, auf datenbezogenen Informationen basierenden, Fragenkatalog, der in einem Wenn-Dann-Regelsystem mündet (vgl. Danubianu & Socaciu, 2011).

Im Areal der **Meta-Learning-Systeme** ist zunächst, als eines der ersten und umfangreichsten seiner Art, auf das System *StatLog* nach MICHIE, SPIEGELHALTER, TAYLOR & CAMPBELL (1994) zu verweisen. Die Autoren untersuchen hierbei, nach dem unter **Kapitel XX** beschriebenen Vorgehen, 19 Datensets mit zehn differenten Klassifikationsalgorithmen und markieren in der Trainingsphase die jeweiligen Algorithmen mit *applicable* bzw. *not-applicable*. Anschließend definieren die Autoren für jeden Algorithmus Entscheidungsbäume zur Bestimmung dessen Anwendbarkeit auf den verfügbaren Datensatz. Resultierend entsteht ein Set an Regeln der Form „*Algorithmus <- Kondition*“, welche den Anwender bei der Auswahl der Methode unterstützen (vgl. Michie et al., 1994, S. 200ff.). Weiterhin beschreibt GIRAUD-CARRIER (2005) das System *Data Mining Advisor*, welches auf *StatLog* aufbaut, allerdings keine Unterteilung in *applicable* und

not-applicable vornimmt, sondern eine Rangfolge aller Algorithmen anhand der extrahierten Metadaten errechnet (vgl. Vanschoren, 2010, S. 60ff.). Generell existiert in der Literatur eine Vielzahl an weiteren Systemen des Meta-Learnings, wie beispielsweise NOEMON nach KALOUSIS & HILARIO (2001), VBMS nach RENDELL, SHESHU & TCHENG (1987) oder die Verwendung von prädiktiven Clustering Bäumen nach TODOROVSKI, BLOCKEEL & DZEROSKI (2002). Diese werden allerdings aufgrund des Rahmens dieser Arbeit nicht weiter behandelt.

Final wird der Prototyp in **Kapitel XX** aufgrund der Entscheidungsunterstützung als ein inhaltsbasiertes Recommender-System kategorisiert. Die ersten Ansätze für die inhaltsbasierte, personalisierte Empfehlung von Objekten basieren hierbei auf Methoden wie schlüsselwortbasierten Filtertechniken. Besonders hervorzuheben ist hierbei das Areal der **Keyword Extraction**, welches die automatisierte Suche von Schlüsselwörtern in Dokumenten thematisiert (vgl. Hasan & Ng, 2014, S. 1262). In der Umsetzung solcher Systeme werden häufig die extrahierten Schlüsselwörter und andere Eigenschaften eines Textes als Vektor repräsentiert und dieser mit einem anderen verglichen, der den zu klassifizierenden Text darstellt. Derartige Methoden eignen sich dabei besonders für Textanalysen und andere Problemstellungen, die in Verbindung mit natürlicher Sprache stehen (vgl. Resnick, Iacovou, Suchak, Bergstrom & Riedl, 1994, S. 5). Als ein erster Prototyp eines solchen Systems ist das *SIFT*¹ System der Stanford Universität aufzuführen, das basierend auf einem Keyword-Profil Artikel reklamiert. Dabei konfiguriert der Nutzer das Keyword-Profil manuell (vgl. Yan & Garcia-Molina, 1995). Weiterhin konstatieren WANG ET AL. (2018) ein auf Keyword Extraction basiertes Recommender-System für Nachrichten. Sie betonen, dass mittlerweile eine Vielzahl zur automatisierten Suche nach Schlüsselwörtern existiert und die Qualität der gefundenen Keywords stark vom jeweiligen Verfahren abhängt (vgl. Wang et al. (2018), S. 4341ff.). Additional konstruieren HABIBI & POPESCU-BELIS (2015) ein Recommender-System, welches mithilfe der Keyword Extraction und Ähnlichkeitsmaßen verschiedene Wikipedia-Artikel empfiehlt. Auch hier bekräftigen die Autoren die Vielfältigkeit an potentiellen Algorithmen der Keyword Extraction (vgl. Habibi & Popescu-Belis, 2015, S. 746–759). Als Vertreter der Algorithmen zur Termgewichtung in diesem Areal sind exemplarisch die Algorithmen TF-IDF nach SALTON & BUCKLEY (1988), TF-IGM nach CHEN, ZHANG, LONG & ZHANG (2016)

¹ SIFT steht hierbei für *Stanford Information Filtering Tool*.

und YAKE nach CAMPOS ET AL. (2018) anzuführen. In der Subkategorie der graphbasierten Verfahren hingegen ist auf die Algorithmen TextRank nach MIHALCEA & TARAU (2004A), PositionRank nach FLORESCU & CARAGEA (2017) und TopicRank nach Bougouin, Boudin, & Daille (2013) zu verweisen.

Während Recommender-Systeme in multiplen, differenten Bereichen Anwendung finden, kristallisiert sich E-Commerce, bspw. durch das Generieren personalisierter Vorschläge von Verkaufsgegenständen, wie CDs und Büchern bei LINDEN, SMITH & YORK (2003) oder das Vorschlagen von Filmen nach MILLER ET AL. (2003), als Hauptanwendungsfeld in aktueller Literatur heraus. Aufgrund der ökonomischen Relevanz von Recommender-Systemen im E-Commerce ist in den letzten Jahren ein Forschungsanstieg in diesem Areal zu konstatieren. Als Resultat dieser Entwicklung zeigen heutige Implementierungen von Recommender-Systemen die Applikation komplexerer Methoden des Machine Learning, wie beispielsweise bei AMATO, MOSCATO, PICARIELLO & PICCIALLI (2019) oder BAG, GHADGE & TIWARI (2019).

Als weitere Methodengruppe, welche aus diesem Forschungsanstieg sowie der Weiterentwicklung der Keyword Extraction resultiert, manifestiert sich das **Topic Modeling**. Dieses verfolgt das Ziel aus einer variablen Dokumentenmenge die wichtigsten Themen automatisiert zu extrahieren. Diese Themen wiederum können als eine Menge von Schlüsselwörtern verstanden werden, die das jeweilige Thema am besten beschreiben (vgl. Schneider, 2017, S. 1). Auf der Grundlage wissenschaftlicher Artikel propagieren WANG & BLEI (2011) ein Recommender-System, um personalisierte Artikelvorschläge zu berechnen. Dabei wird zum einen das Modell *Latent Dirichlet Allocation* verwendet, um ungesuchte Dokumente bezüglich eines Themas zu klassifizieren. Zum anderen wird auf Basis der „Gefällt-mir-Angaben“ von Nutzern ähnliche Nutzerprofile sowie deren präferierte Artikel empfohlen (vgl. C. Wang & Blei, 2011, S. 450f.). Als ein weiteres Beispiel ist die Forschungsarbeit von PENNACCHIOTTI & GURUMURTHY (2011) zu nennen, die ein auf Topic Modeling basierendes System für Freundschaftsempfehlungen publizieren. Hierbei verwenden die Autoren Kurznachrichten, um neue Freundschaftsempfehlungen zu errechnen.

Zu erkennen ist, dass die eigentliche Textklassifikation in den aufgezeigten Systemen häufig indirekt mittels Ähnlichkeitsmaßen oder auf Basis von Topicverteilungen erfolgt. Gerade in Hinsicht dieser Klassifikation definiert BISHOP (2006) multiple Methoden des

Machine Learning, von denen sich einige ebenfalls in modernen Recommender-Systemen wiederfinden. Beispielsweise entwickeln MOONEY & ROY (2000) ein Empfehlungssystem, welches mithilfe des *Naive-Bayes-Verfahrens* eine Klassifikation auf Basis von Nutzerprofilen durchführt, die ebenfalls auf Textdaten beruhen. Zu Reduktion der Komplexität und Rechenintensivität dieser Algorithmen, wird häufig auf Methoden der Keyword Extraction zurückgegriffen, um das zugrundeliegende Vokabular einzuschränken (vgl. Xu & Araki (2006), S. 402). Unter Verwendung einer *Support Vector Machine* (SVM) haben XU & ARAKI (2006) ein Recommender-System entworfen, das individualisierte Empfehlungen für TV-Programme errechnet. Die Grundlage hierfür sind Programmdateien in Form von natürlichsprachlichem Text, welcher vor dem Training des Modells durch Keyword-Extraction-Methoden auf die wichtigsten Wörter reduziert wird. Neben der Verwendung einer SVM oder der *Naive-Bayes-Klassifizierung* zeigt Bishop (2006) eine Vielzahl an potentiellen Algorithmen auf, die für diesen Zweck angewendet werden können.

Wie bereits angedeutet, bedienen sich moderne Systeme, welche Klassifikationen jeglicher Art vornehmen, zumeist Methoden des Machine oder Deep Learning. Generell ist hierbei festzustellen, dass **Deep-Learning-Verfahren** als State-of-the-Art-Lösung in diesem Bereich zu erachten sind. Im Kontext von Textdaten propagiert ELMANN (1990) als einen der ersten Ansätze im Rahmen des Deep Learning ein *Recurrent Neural Network* (RNN), welches die Speicherung der Semantik vorangegangener Texte ermöglicht. Resultierend aus der Problematik zu geringer oder zu hoher Gradienten im Training von RNNs auf langen Sequenzdaten entwickeln HOCHREITER & SCHMIDHUBER (1997) die rekurrente Architektur des Long Short-Term Memory (LSTM). Als Weiterentwicklung des klassischen RNN verändern die Autoren die interne Struktur des Modells und wirken so der beschriebenen Problematik entgegen. Generell sind LSTMs, durch komplexere Berechnungen in jedem Schritt des Trainings, rechenintensiver als klassische RNNs, weshalb CHO ET AL. (2014) die sog. Gated-Recurrent-Units (GRU) publizieren, die trotz der Lösung der initialen Problematik weniger Komplexität aufweisen. Einen weiteren, nicht den rekurrenten Netzwerken zuordenbaren, Ansatz liefern COLLOBERT & WESTON (2011), die ein *Convolutional Neural Network* (CNN) zur Textklassifikation verwenden. Diese Netzarchitektur verwendet sog. Kernels (z.Dt.: Filter), welche den Text iterativ einlesen und verarbeiten (vgl. Collobert et al., 2011, S. 2267f.).

Prinzipiell greifen die meisten Verfahren aus dem Areal des Machine und Deep Learning auf operationalisierte Textdaten zurück. Hierbei finden in aktueller Literatur und modernen NLP-Systemen häufig **distribuierte Vektorrepräsentationen** von Wörtern Anwendung. Dies resultiert zum einen aus deren Fähigkeit die Semantik bzw. Bedeutung eines Wortes mit in dessen repräsentativen Vektor zu inkludieren, zum anderen aus der Dichte dieser Vektoren (vgl. Huang & Yates, 2009, S. 495; Perone et al., 2018b, S. 2). Basierend auf der Pionierarbeit von MIKOLOV ET AL. (2013) berechnen einfache neuronale Netze distribuierte Repräsentationen, welche die Semantik durch den Wortkontext einfassen. Allerdings gehen diese Modelle mit einigen Limitationen einher. Einerseits wird der Kontext eines Wortes nur durch einen lokalen Filter bestimmt, wodurch in einem Schritt lediglich der momentane Kontext beachtet wird. Andererseits können diese Modelle keine Wörter distribuiert repräsentieren, die sie vorher nicht gelernt haben, sodass sich eine Nutzung als durchaus diffizil gestaltet (vgl. Perone et al., 2018a, S. 2f.). Als eine Weiterentwicklung der Word2Vec Modelle hat sich das Modell *Global Vectors* (GloVe) etabliert, welches mittels einer globalen Co-Occurrence-Matrize den Kontext eines Wortes bestimmt. Untersuchungen zeigen, dass diese Modelle zu besseren Ergebnissen als die klassischen Word2Vec-Modelle führen (vgl. Pennington, Socher, & Manning, 2014, S. 1532). Additional zeigen BOJANOWSKI ET AL. (2016) ein Modell (FastText) auf, welches ebenfalls eine Erweiterung der von MIKOLOV ET AL. (2013) vorgestellten Modelle darstellt. Dieses Modell eignet sich speziell bei der Verwendung kleiner Datenmengen, da es auf Basis von Wort-N-Grammen distribuierte Repräsentationen errechnet (vgl. Bojanowski, Grave, Joulin, & Mikolov, 2016, S. 137f.). Zeitgleich wird auf diese Weise die Wahrscheinlichkeit für ungewöhnliche Wörter vermindert. Hinzukommt, dass die Architektur ein schnelleres Training als die eben genannten Modelle erlaubt (vgl. Joulin, Grave, Bojanowski, & Mikolov, 2017, S. 429). Neben den Erweiterungen der Word2Vec-Modelle hat sich zudem die Architektur *Embedding from Language Models (ELMO)* entwickelt, welche unter Zuhilfenahme eines bidirektionalen Language-Models distribuierte Repräsentationen berechnet (vgl. Peters et al., 2018, S. 2227f.).

Die aufgezeigten Modelle zur Berechnung von distribuierten Wortvektoren verdeutlichen die Größe dieses Forschungsareals. Hinsichtlich der Überführung von ganzen Paragraphen in Vektoren ist allerdings eine Forschungslücke zu verzeichnen (vgl. Perone et al., 2018b, S. 1). Als Grundverfahren zur Errechnung von distribuierten Paragraphrepräsentation ist das Bilden des Mittelwerts über verschiedene Wort-Embeddings zu

nennen. Jedoch weisen derartige Verfahren Unzulänglichkeiten bzgl. der Inklusion der exakten Semantik langerer Paragraphen auf (vgl. Perone et al., 2018a, S. 1ff.). Basierend auf einem Encoder-Decoder-Modell etabliert sich *Skip-Thought* als Embedding-Modell, das es ermöglicht einzelne Paragraphvektoren zu erlernen. Die grundlegende Idee hierbei ist, dass der Encoder einzelne Wortvektoren zu einem Satzvektor vereinigt während der Decoder den Kontext des Paragraphen prognostiziert (vgl. Kiros et al., 2015, S. 3296). Um die Rechenintensität des Skip-Thought-Ansatzes zu verringern, propagieren LAJANUGEN & HONGLAK (2018) weiterhin ein Modell, welches anstelle des Decoders einen Klassifikationsalgorithmus verwendet, der auf einem Kandidatenset, bestehend aus Sätzen, den umstehenden Kontext vorhersagt.

Darüber hinaus sind zwei von Google vorgestellte Modelle im Bereich des *Transfer Learnings* nennenswert: Zum einen ein komplexes, rechenintensives Encoder-Modell, das auf hohe Modellgenauigkeiten abzielt (das sog. *Transformer-Modell*), und zum anderen ein auf Mittelwertbildung basierendes Verfahren (sog. *Deep-Averaging-Networks* (DAN)). Hierbei ist festzuhalten, dass erstgenanntes zwar höhere Genauigkeiten als ein DAN erzielt, dies jedoch auf Kosten der Performanz (vgl. Cer et al., 2018a, S. 2).

Kommentiert [DH11]: 169

4 Additionale Methoden

In Anlehnung an die vorangegangene Forschungsarbeit dient das folgende Kapitel zur Identifikation und Deskription additionaler Methoden, welcher zur Erreichung der aktuellen Zielstellung angewendet werden können. Diese Erweiterung der methodischen Basis erfolgt aufgrund des generischen Fokus der vorherigen Arbeit, welche vor allem das Aufzeigen von potentiellen, methodischen Gruppen zur Umsetzung des Forschungsvorhabens sowie die Deskription deren Basismethoden umfasst.

4.1 Embeddings

Wie in **Abschnitt XYZ** bereits deskribiert, existiert eine Vielzahl an Methoden, um Wörter distributiv zu repräsentieren. Die Verwendung von Embeddings bzw. distribuierten Repräsentationen hat in vielen nachgelagerten NLP-Anwendungen Verbesserungen erzielt, wie beispielsweise in den Forschungsarealen der Textklassifikation, Machine-Translation oder bei Frage-Antwort-Systemen (vgl. José Camacho-Collados & Pilehvar, 2018, S. 743). Zum einen ist dies darauf zurückzuführen, dass mit dichtbesetzten Vektoren gelernt wird, welche Syntax und Semantik eines Wortes inkludieren, zum anderen aber auch auf die Generalisierungsfähigkeit von distribuierten Repräsentationen (vgl. Goldberg, 2016, S. 366). Wenngleich Word-Embedding-Modelle hochqualitative Vektorrepräsentation errechnen, so ist die distribuierte Repräsentation von Paragraphen noch vergleichsweise unerforscht (vgl. José Camacho-Collados & Pilehvar, 2018, S. 1f.)

Das Erzeugen von Paragraphrepräsentationen lässt sich grundsätzlich auf das Verwenden einer Kompositionsfunktion zurückführen, die eine mathematische Funktion beschreibt, welche einzelne Wörter in einen einzigen Vektor transformiert. Dabei wird zwischen zwei Arten von Funktionen unterschieden: ungeordneten und syntaktischen Funktionen. Erstere berechnen die Vektoren ungeachtet ihrer Reihenfolge und zweitgenannte beziehen die Satzstruktur sowie die Reihenfolge der Wörter mit ein (vgl. Iyyer, Manjunatha, Boyd-Graber, & Daumé III, 2015a, S. 1681). In Bezug auf Aufgaben der

Textklassifikation zeigen kürzlich durchgeführte Experimente, dass Modelle mit einer syntaktischen Funktion ungeordnete Kompositionsfunktionen übertreffen (vgl. Perone et al., 2018a, S. 9). Nichtsdestotrotz benötigen diese Methoden mehr Trainingszeit und -daten, um zufriedenstellende Ergebnisse zu erzielen (vgl. Iyyer et al., 2015a, S. 1681).

Im Folgenden werden ausgewählte Methoden für die Erstellung von distribuierten Repräsentationen deskribiert, welche im Rahmen des Forschungsvorhabens Anwendung finden.

4.1.1 FastText

Wie zuvor erwähnt, existieren im Bereich der Wort- bzw. Dokument-Embeddings zahlreiche Ansätze, Methoden und Architekturen. Eine dieser Architekturen wird von BOJANOWSKI, GRAVE, JOULIN UND MIKOLOV (2017) publiziert. Die Autoren beschreiben das Modell FastText, welches als Weiterentwicklung aus dem bereits deskribierten Modell Word2Vec von MIKOLOV, CORRADO, CHEN & DEAN (2013) hervorgeht. Wie bereits in der vorhergegangenen Forschungsarbeit aufgezeigt, trainieren MIKOLOV ET AL. (2013) im Rahmen der Skip-Gram-Architektur Wort-Embeddings unter dem Prinzip, den Kontext eines Wortes vorherzusagen. In **Abbildung XX** wird diese Architektur schematisch illustriert und verdeutlicht, dass auf Basis eines Wortes die Vorhersage der Kontextwörter im Rahmen eines Kontextfensters (Hier mit einer Größe von $n = 2$ dargestellt) fokussiert wird. Der Hidden-Layer p dieser Architektur fungiert dabei als Projizierungsebene, da die initialisierten Gewichte aus der Gewichtsmatrix gw durch die Multiplikation mit dem One-Hot-kodierten Inputwort w auf p projiziert werden (vgl. Mikolov, Corrado, et al., 2013, S. 3ff.).

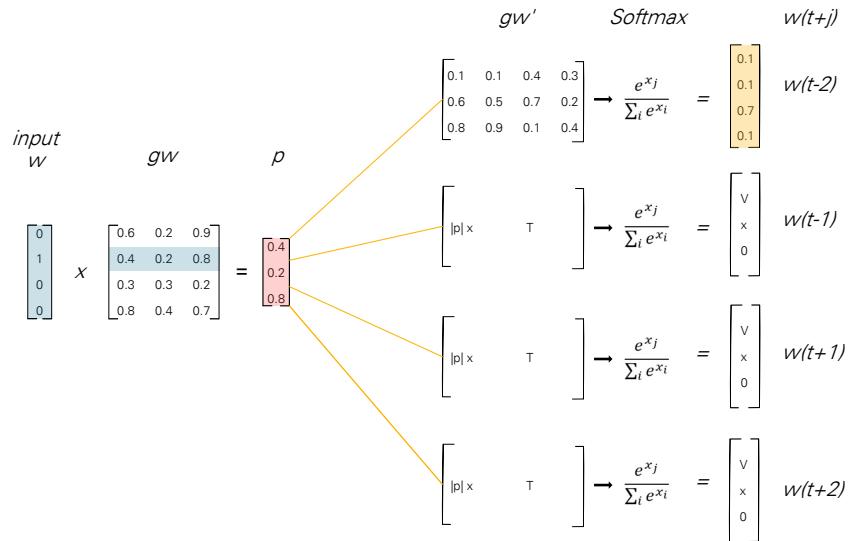


Abbildung 7: Architektur des Word2Vec-Modells nach MIKOLOV ET AL. (2013)

BOJANOWSKI ET AL. (2017) transferieren diese Methodik auf Buchstaben-N-Gramme, trainieren somit Vektoren für sämtliche dieser im Korpus enthaltenen N-Gramme und beachten hierdurch die interne Struktur von Wörtern. Ein Wort-Vektor wird daraufhin aus der Summe der jeweiligen N-Gramm-Vektoren gebildet, was es ermöglicht qualitativ hochwertige Vektoren für seltene Wörter im Korpus zu erreichen, da deren interne Struktur eine Ähnlichkeit zu weiteren im Korpus enthaltenen Wörtern aufweisen kann. Weiterhin wirkt dieses Vorgehen der Problematik entgegen, dass unter Verwendung des Word2Vec-Modells lediglich Vektoren für trainierte Wörter erstellt werden können. Aufgrund der Ausrichtung des Trainings auf Wortbestandteile besteht einerseits die Option Vektoren für bisher unbekannte Wörter auf Basis deren Buchstaben-N-Gramme herzuleiten, andererseits erhöht dieser Sachverhalt die Robustheit der Architektur hinsichtlich der zum Training verfügbaren Datenmenge. Überdies evaluieren die Autoren ihr Modell unter anderem mittels Korrelationen zwischen menschlich eingeschätzter Ähnlichkeit und der vom Modell bzgl. differenter Wortpaare errechneten Werte. Mithilfe dieser Methodik zeigen die Autoren auf neun von zehn differenten Datensets unterschiedlicher

Sprache bessere Ergebnisse als mit dem Word2Vec-Modell mittels der Continuous-Bag-of-Words- bzw. der Skip-Gram-Architektur (vgl. Bojanowski et al., 2017, S. 137ff.).

Vor dem Hintergrund der aktuellen Zielstellung, den hiermit verbundenen, divergenten Vokabularen der beiden bestehenden Personengruppen sowie des geringen Datenumfangs ist demnach anzunehmen, dass sich diese Architektur, im Vergleich zu rein auf Trainingsvokabular beschränkten Methoden, am besten für die aktuelle Untersuchung eignet.

4.1.2 Deep Averaging Networks

Deep Averaging Networks (DAN) verwenden eine ungeordnete Kompositionsfunktion, welche auf Basis bestehender Word-Embeddings gleichdimensionale Paragraphvektoren berechnet. Die Architektur wurde von Google aufgegriffen, um ein universales Embedding-Model zur Verfügung zu stellen, welches die Prinzipien des Transfer-Learning² unterstützt. Im Zusammenspiel mit einem bereits trainierten Modell und wenig annotierten Daten erreicht die Architektur State-of-the-Art-Performance, in nachgelagerten NLP-Anwendungen (vgl. Iyyer, Manjunatha, Boyd-Graber, & Daumé III, 2015b, S. 1681–1686).

Das grundlegende Prinzip von DANs ist es eine arbiträre Anzahl von Word-Embeddings zu Mitteln und daraufhin ein neuronales Netz zu verwenden, welches die Paragraph-Embeddings berechnet (vgl. Iyyer et al., 2015b, S. 1685)

² Transfer-Learning ist ein Gebiet des Machine Learning, bei dem ein bereits gelerntes Modell auf ähnlichen Daten trainiert wird und für die Lösung gleichartiger Problemstellungen verwendet wird (vgl. Goodfellow, Bengio, & Courville, 2016, S. 536).

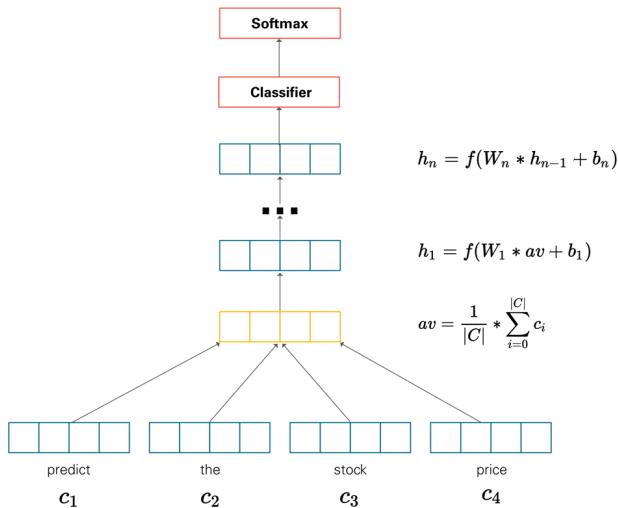


Abbildung 8: Funktionsweise eines DANs (In Anlehnung an IYER ET AL. (2018))

Wie in **Abbildung XYZ** veranschaulicht, funktioniert das Modell in drei grundlegenden Schritten. Zunächst wird der Mittelwert der entsprechenden distributiven Wortrepräsentationen gebildet, um multiple Wort-Embeddings in einen einzigen Vektor zu überführen. Wobei C die Menge der Wörter des Paragraphen beschreibt und c_i die jeweiligen Elemente der Menge.

$$av = \frac{1}{|C|} \sum_{i=0}^{|C|} c_i$$

Das Resultat ist ein Vektor derselben Dimensionalität, wie die der jeweiligen eingehenden Wortvektoren. Daraufhin wird der gemittelte Vektor durch n verschiedene Hidden-Layer geführt. Dabei beschreibt W_n die Gewichte des entsprechenden Hidden-Layer h_n sowie b_n den entsprechenden Bias. Die Anzahl der Hidden-Layer sowie die zugrundeliegende Aktivierungsfunktion $f(x)$, sind dem Modellierer überlassen.

$$\begin{aligned} h_1 &= f(W_1 * av + b_1) \\ h_n &= f(W_n * h_{n-1} + b_n) \end{aligned}$$

Der Output aus dem letzten Hidden-Layer bildet nach der Optimierung des Netzwerkes den distribuierten Paragraphvektor. Allerdings setzt dies erst ein Training des Netzes voraus. Dafür schließen sich den grundlegenden Hidden-Layern ein Classifier an, welcher aus beliebig vielen weiteren Schichten bestehen kann. Die Anzahl der Neuronen im letzten Layer wird dabei durch die Zielklassen der zugrundeliegenden Datenbasis bestimmt. Hierbei wird sich einer Softmax-Funktion als bedient, welche eine Wahrscheinlichkeitsverteilung über die Menge der Zielklassen wiedergibt.

$$E(h_n) = \frac{e^{W_{h_i}}}{\sum_j e^{h_{nj}}}$$

Im Anschluss wird mithilfe einer Verlustfunktion und einem Optimierungsverfahren das Netz angelernt, sowie die Gewichte der Hidden-Layer angepasst. Zu betonen ist, dass der sich anschließende Classifier nicht für die Inferenz der einzelnen distribuierten Repräsentationen verwendet wird, sondern lediglich dazu dient, um die Gewichte des DAN anzupassen (vgl. Iyyer et al., 2015b, S. 1681).

Dabei kann das Model durch eine Dropout-basierte Technik verbessert werden, indem für jeden Trainingsschritt, mit einer bestimmten Wahrscheinlichkeit, ein Wort aus dem Input-Layer entfernt wird (vgl. Iyyer et al., 2015b, S. 1685–1686).

Die Intuition eines DAN ist es, dass jeder der verschiedenen Netzsichten eine immer mehr abstraktere Repräsentation des Inputs berechnet. Dadurch werden kleine aber ausschlaggebende semantische Unterschiede stärker hervorgehoben und in den resultierenden Vektor inkludiert. Die untenstehenden **Abbildung XYZ** illustriert vier verschiedene Paragraphvektoren. Das linke Schaubild illustriert Paragraphvektoren, welche auf Basis des arithmetischen Mittels einzelner Wordvektoren, Paragraphvektoren erzeugt. Das Rechte Bild veranschaulicht Paragraphvektoren, welche auf Basis, der in **Abbildung XYZ** beschriebenen Architektur erzeugt wurden. Als Eingabevektoren sind hierbei Fast-Text-Embeddings verwendet worden.

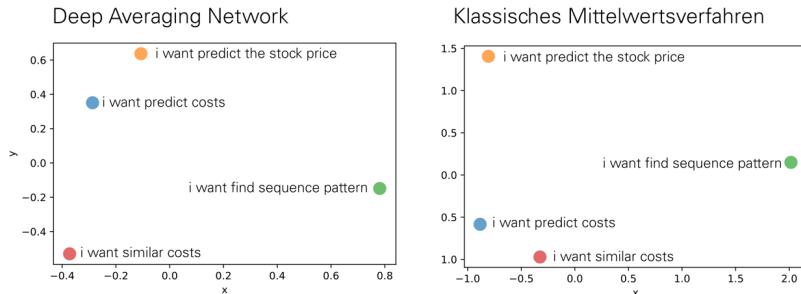


Abbildung 9: Gegenüberstellung DAN und Mittelwertsverfahren (Eigene Darstellung)

In Anlehnung an, die in **Kapitel XYZ** definierte Zielstellung, sind vier verschiedene Sätze der in **Kapitel XYZ** zugrundeliegenden Klassen abgebildet. Dabei unterscheiden sich die Sätze nur in kleinen semantischen Nuancen. Beispielsweise unterscheiden sich die beiden Sätze „*I want predict costs*“ und „*I want similar costs*“ nur in einem Wort zueinander. Obwohl beide Paragraphen auf zwei unterschiedliche Klassen hindeuten sind beide Paragraphen bei dem klassischen Mittelwertverfahren sehr dicht beieinander, was eine große Ähnlichkeit impliziert. Das Abstrahieren dieses Mittelwerts durch multiple darauf-folgende Hidden-Layer, grenzt die kleinen semantischen Unterschiede besser voneinander ab. Hierbei liegen Paragraphen einer Klasse näher beieinander als durch das reine Bilden eines Mittelwertes.

IVYER ET AL. (2018) heben zusätzlich hervor, dass DANs besonders gut bei „unscharfen“ Daten funktionieren, sowie Domänenübergreifend State-of-the-Art-Ergebnisse erzielen. So funktionieren diese Modelle gut mit Datengrundlagen, welche eine hohe syntaktische Divergenz aufweisen. Die geringe benötigte Rechenleistung sowie die Fähigkeit dieser Modelle syntaktische Divergenzen zu überwinden begründet die Verwendung des Verfahrens in der zugrundeliegenden Forschungsarbeit.

4.1.3 Universal Sentence Encoder

Der Universal Sentence Encoder (USE) sind zwei von Google propagierte Modelle der distributionellen Semantik, welche sich insbesondere auf das Gebiet des Transfer-Learning spezialisieren. Beide Modelle erzielen State-of-the-Art-Ergebnisse, in Anbetracht der Effizienz und Performance nachgelagerter NLP-Anwendungen. CER ET AL. (2018)

konstatieren dabei, dass die Applikation von Transfer-Learning, im Rahmen der Paragraph-Embeddings, andere Modelle der distributionellen Semantik übertreffen. Hinzu kommt, dass auf Basis geringer Datenmengen, diese Modelle vergleichbare Ergebnisse erzielen wie bereits erwähnte Modelle der distributionellen Semantik. Zu betonen sei dabei, dass diese Modelle besonders gut semantische Nuancen der Paragraphen inkludieren und somit State-of-the-Art-Ergebnisse im Erkennen textueller Ähnlichkeiten erzielen (vgl. Perone et al., 2018, S. 10). Die restriktierte Menge an verfügbaren Trainingsdaten, in Bezug auf differente NLP-Applikationen, unterstreicht die Relevanz dieser Modelle (vgl. Cer et al., 2018, S. 174). Die beiden von Google bereitgestellten Modelle kreieren dafür Vektoren mit jeweils 512 Dimensionen.

Die beiden von CHER ET AL. (2018) vorgestellten Modelle bieten die Möglichkeit zwischen Effizienz und Modellgenauigkeit abzuwählen. Zum einen basiert das leichtgewichtige Modell auf der von IYYER ET AL. (2015) deskribierten DAN-Architektur und verspricht ein performantes Training sowie schnelle Inferenz. Das trainierte DAN-Modell ist dabei auf Wörtern sowie N-Gramen bzw. Paragraphen trainiert. Zum anderen stellen die Autoren ein schwergewichtiges Transformer-Modell vor, welches zwar bessere Paragraph-Embeddings berechnet, jedoch rechenaufwändiger ist. Letztgenanntes basiert dabei auf der von VASWANI ET AL. (2017) vorgestellten Deep-Learning-Architektur, welche die Wortreihenfolge mit einbezieht, sowie speziellen Attention-Funktionen nutzt, um die distribuierten Repräsentation zu berechnen (vgl. Vaswani et al., 2017, S. 5999ff.). Allerdings steht der damit verbundene Rechenaufwand in keinerlei Bezug für die im Rahmen der Forschungsarbeit zur Verfügung stehenden Ressourcen, weshalb auf eine Anwendung, dessen verzichtet wird.

Für die final publizierten USE-Modelle verwenden CHER ET AL. (2018) eines von MIKOLOV ET AL. (2013) trainiertes Word2Vec-Modell. Im Rahmen des aktuellen Forschungsvorhabens soll das auf der DAN-Architektur³ fundierende Modell aufgegriffen und mit anderen vorgestellten Methodiken verglichen werden. Im Folgenden wird hierbei mit der Bezeichlichkeit USE auf das DAN basierte Modell referenziert.

Kommentiert [DH12]: Weiter Ausführen?

Kommentiert [DH13R12]: Change:

- ➔ Nutzung wegen schneller Inferenz
- ➔ Iyyer et al. Sagen DANs syntaktische Divergenz umgehen (S. 1683)
- ➔ Berechnung mehrerer Modelle nötig > Reduzierung Zeitaufwand

³ Eine Beschreibung zum originalen Modell findet sich unter folgenden Link:
<https://tfhub.dev/google/universal-sentence-encoder-lite/>

Allerdings soll hierbei auf eine umfassende Anpassung der Parameter sowie auf eine Modellevaluation verzichtet werden. Hierbei sei auf die Evaluationsergebnisse von PERONE ET AL. (2018) verwiesen, welche eine ausführliche intrinsische sowie extrinsische Evaluation der Modelle aufzeigen.

4.2 RNN-basierte Technologien

Weiterhin finden sich im Bereich des Deep Learning Architekturen, welche aufgrund ihrer Fähigkeit zur Verarbeitung von sequenziellen Daten, im Bereich des NLP starke Anwendung finden. Generell erreicht ein Wort seine Bedeutung durch die diesem vorhergegangenen Wörter. Exemplarisch sind hier die englischen Wörter „dog“ sowie „hot dog“ anzuführen, in welchen eine starke Abweichung des Sinnes des Wortes „dog“ durch das vorausgehende Wort „hot“ zu verzeichnen ist. Die Fähigkeit von RNNs sequentielle Daten zu verarbeiten erlaubt diesen Architekturen die sequentielle Struktur von Sprache und somit kontextuelle Abhängigkeiten zwischen Wörtern zu verarbeiten, indem Buchstaben, Wörter oder Sätze nacheinander in das Netzwerk eingelesen werden. Aufgrund dieser Vorteile resultieren in einem vermehrten Fokus des NLP-Bereichs auf RNN-basierte Technologien (vgl. Young, Hazarika, Poria, & Cambria, 2018, S. 65f.).

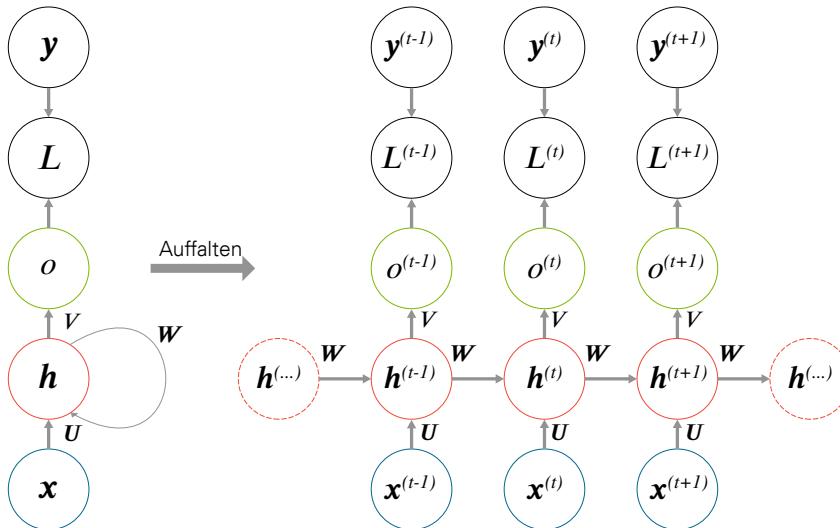


Abbildung 10: Berechnungsgraph eines Recurrent Neural Networks nach GOODFELLOW ET AL. (2016), S. 378

In Referenz zur vorherigen Forschungsarbeit illustriert **Abbildung XX** diese Architektur, wobei x die Inputsequenz der Länge τ darstellt, welche über differente Zeiteinheiten $t \in \{0, \dots, \tau\}$ in das Netzwerk eingelesen werden. Die oben angesprochene Kontextverarbeitung resultiert zudem aus der Weitergabe der Gewichtsmatrix W zwischen den einzelnen Zeitpunkten, welche als additionaler Input in die Verarbeitung von $x^{(t)}$ in $h^{(t)}$ ein geht. So entsteht der Input pro Zeitstempel $a^{(t)}$ sowie die Verarbeitung dieses aus den folgenden Formeln:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \quad (1)$$

$$h^{(t)} = \tanh(a^{(t)}) \quad (2)$$

mit den Gewichtsmatrizen W und U , der Aktivierung des vorherigen Zeitstempels $h^{(t-1)}$, dem aktuellen Input $x^{(t)}$ sowie dem Bias-Vektor b (vgl. Goodfellow et al., 2016, S. 378ff.).

Trotz dieser Vorteile kristallisieren sich ebenfalls Schwächen dieser Basisarchitektur heraus. Generell verfolgt diese zur Berechnung der partiellen Ableitungen der Zielfunktion

bzw. der Gradienten das Verfahren der Back-Propagation-Through-Time (BPTT). Hierfür wird das Netzwerk, analog zur **Abbildung XX** rechts, aufgefaltet, um anschließend, unter Betrachtung der einzelnen Zeitpunkte bzw. „*hidden states*“ als Ebenen des Netzwerkes, das Verfahren der Backpropagation von Feed-Forward-Netzwerken anzuwenden (vgl. Gruslys, Munos, Danihelka, Lanctot, & Graves, 2016, S. 4133). Dieses Vorgehen resultiert in einer Problematik mit längeren Inputsequenzen bzw. einer damit verbundenen, tieferen Architektur des Netzwerkes, da der berechnete Fehlerwert, welcher in die Anpassung der Gewichte des Netzwerks einfließt, in Relation zu dieser tiefen exponentiell wächst oder sinkt. Dies ist darauf zurückzuführen, dass in jedem Zeitpunkt des RNN gleiche Operationen mit der, über die Zeitpunkte konstanten, Gewichtsmatrix W durchgeführt werden. Diese wiederholte Multiplikation mit gleichen Werten führt letztendlich entweder zu sehr großen (sog. Exploding Gradients) oder zu sehr geringen Gradienten (sog. Vanishing Gradients) und somit zu ineffizienten Gewichtsanpassungen durch den applizierten Optimierungsalgorithmus. Erstere Problematik führt daher zu einem instabilen Training, da häufige, große Gewichtsanpassungen das Erreichen des globalen Minimums für den Optimierungsalgorithmus erschweren oder gar verhindern. Zweitere Problematik hingegen resultiert in einer Stagnation des Trainings, da nur minimale Anpassungen durchgeführt werden und hier eine Unsicherheit entsteht in welche Richtung die Gewichte angepasst werden müssen um die Kostenfunktion zu optimieren (vgl. Goodfellow et al., 2016, S. 289f.; Sussillo & Abbott, 2014, S. 1f.).

4.2.1 Long Short-Term Memory

Zur Adressierung dieses Sachverhalts publizieren HOCHREITER & SCHMIDHUBER (1997) die Architektur des LSTM in seiner ersten Version. Diese findet in den Folgejahren hoher Anklang und wird sukzessive durch andere Publikationen wie beispielsweise von GERS, SCHMIDHUBER & CUMMINS (2000) oder BAYER, WIERSTRÄ, TOGELIUS & SCHMIDHUBER (2009) angepasst und optimiert, was letztendlich in der heutigen Version des LSTM resultiert. Generell ersetzen die Autoren die singuläre Tanges-Funktion einer jeden Zelle eines klassischen RNNs (vgl. **Formel XX**) durch mehrere, komplexere Operationen, welche dem oben deskribierten Problem entgegenwirken. **Abbildung XX** gibt hierfür eine ablauforientierte Illustration einer LSTM-Zelle.

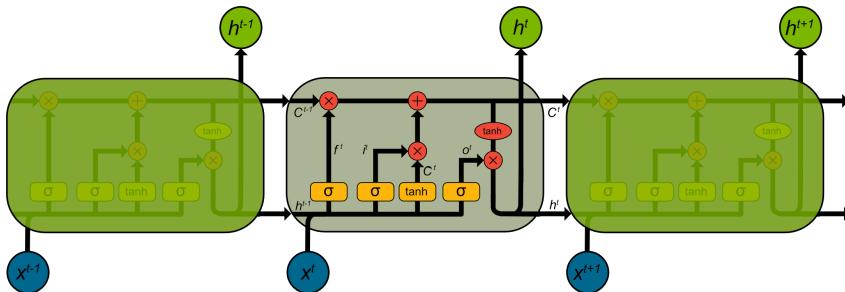


Abbildung 11: Illustration der Funktionsweise einer LSTM-Zelle (Eigene Darstellung nach XYXYX)

Kommentiert [m14]: Wie kann man Colahs Blog zitieren?
:D

Auffällig in dieser Architektur ist primär der sog. Cell State (CS) (z.B. Zellenstatus), der über alle Zeitpunkte bzw. durch alle Zellen hinweg relevante Informationen trägt. Hierbei existieren in jeder Zelle lineare Operationen zum Einfügen, Löschen oder Aktualisieren des CS. Die Autoren definieren divergente Gates (z.Dt. Tore), die in ihrer Gesamtheit für die Verwaltung des CS zuständig sind (vgl. Hochreiter & Schmidhuber, 1997, S. 1740f.).

Zunächst ist hierbei das sog. Forget Gate anzusprechen, welches durch GERS ET AL. (2000) zur Basisversion des LSTMs addiert wird. Dieses Gate, charakterisiert durch die erste Sigmoid-Funktion sowie die darauffolgende Multiplikation mit dem CS, legt fest, welche bisher bereich gespeicherten Informationen im CS gelöscht und somit „vergessen“ werden (vgl. Abbildung XX).

$$f^t = \sigma(W_f[h^{t-1}, x^t] + b_f) \quad (3)$$

Dies geschieht durch Transformation der konkatenierten Werte der letzten Aktivierung h^{t-1} sowie des neuen Inputs x^t mittels einer sigmoiden Funktion, welche Werte im Bereich von 0 und 1 ausgibt (vgl. Formel XX). Durch Multiplikation dieser Werte mit dem letzten CS werden Informationen gewichtet und eventuell gelöscht (vgl. Gers et al., 2000, S. 2454f.).

Als zweiter Schritt existiert ein sog. Input Gate, welches regelt, welche Informationen des neuen Inputs verwendet und somit im CS gespeichert werden. Hierbei ist das Gate zweigeteilt und berechnet zunächst, welche Informationen des CS generell aktualisiert werden (i^t) und multipliziert diese Informationen mit potentiellen Kandidaten C^t , die mittels einer Tangens-Hyperbolikus-Funktion aus dem Input berechnet werden (vgl. Formeln XX).

$$i^t = \sigma(W_i[h^{t-1}, x^t] + b_i) \quad (4)$$

$$C^t = \tanh(W_C[h^{t-1}, x^t] + b_C) \quad (5)$$

$$C^t = f^t * C^{t-1} + i^t * C^t \quad (6)$$

Resultierend errechnet sich der neue CS auf Basis dieser beiden Gates und beinhaltet somit neue, selektierte Informationen. Abschließend wird eine gefilterte Version des aktuellen CS von der Zelle ausgegeben.

$$o^t = \sigma(W_o[h^{t-1}, x^t] + b_o) \quad (7)$$

$$h^t = o^t * \tanh(C^t) \quad (8)$$

Wie hier zu erkennen erfolgt die Evaluation, welche Informationen ausgegeben werden analog zum Input Gate. Die finale punktweise tanh-Operation erfolgt um die Werte des CS in das Intervall $[-1, 1]$ zu skalieren (vgl. Goodfellow et al., 2016, S. 409ff.; Hochreiter & Schmidhuber, 1997, S. 1746ff.; Young et al., 2018, S. 66).

Abschließend ist additional auf den Vorteil der Variabilität derartiger, auf RNN basierten, Architekturen zu verweisen. In **Abbildung XX, XX und XX** werden sog. Many-to-Many-Architekturen illustriert, da sowohl Inputs als auch Outputs der Netze Sequenzdaten darstellen. Neben dieser Art besteht ebenso die Option lediglich einen Output bei zeitgleichem Sequenzinput zu verwenden. Diese Struktur wird demnach als Many-to-One-Architektur bezeichnet und findet beispielsweise zur Klassifikation von Texten oder zur Vorhersage eines Wertes auf Basis einer Zeitreihe, Anwendung. Analog sind diese Modelle ebenfalls als One-to-One- bzw. One-to-Many-Architekturen realisierbar (**QUELLE**). Im Rahmen der aktuellen Untersuchung sind diese Modelle vor allem in Form von Many-to-One sowie als One-to-One-Architekturen zu untersuchen, da diese zur Klassifikation von Textdaten verwendet werden können. In ersterer Variante sind hierfür alle Wort-Embeddings eines Satzes als Sequenz zu betrachten, wohingegen der Input in zweiter Variante beispielsweise durch ein Satz- bzw. Paragraph-Embedding realisiert werden kann.

4.2.2 Gated Recurrent Units

Neben der LSTM-Architektur hat sich ebenfalls die von CHO ET AL. (2014) propagierte Architektur der Gated-Recurrent-Units (GRU) etabliert, um dem Problem klassischer rekurrenter Netzwerke entgegenzuwirken. Hierbei dient die LSTM-Architektur als Vorlage für GRUs, welche bestimmte LSTM-Gates konkateniert (vgl. Chung, Gülcöhre, Cho, & Bengio, 2014, S. 4). Die nachfolgende Abbildung illustriert zunächst die grundlegende Funktionsweise einer solchen Architektur.

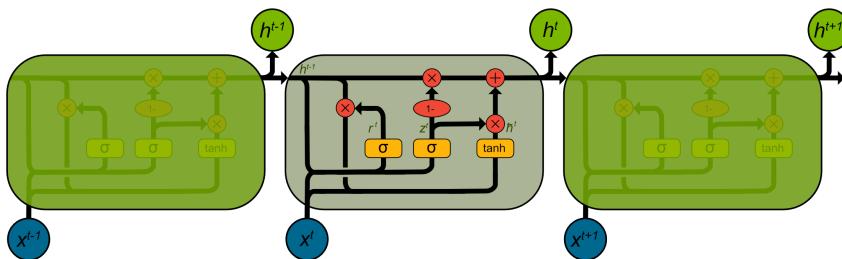


Abbildung 12: Illustration der Funktionsweise einer GRU-Zelle (Eigene Darstellung nach XYXYX)

Im Vergleich zu LSTM-Zellen verzichtet diese Architektur auf zwei verschiedene Zellstatus und kombiniert den Cell-State zusammen mit dem Hidden-State zu einem Zellstatus h_t , welcher den Informationsfluss über die einzelnen Zellen hinweg abbildet. Somit besitzen diese Zellen keine Kontrolle darüber, in welchen Masse die Zellinformationen weitergegeben werden (vgl. Chung et al., 2014, S. 4).

Hinzukommt, dass GRUs auf zwei separate Input- sowie Forget-Gates verzichten und diese in einem Update-Gate z_t kombinieren. Dabei bestimmt das Update-Gate wieviel Informationen vom Hidden-State h_t aktualisiert werden. Hierbei wird der Sigmoidfunktion σ das Produkt der Gewichtsmatrix W_z und den konkatenierten Vektoren $[h_{t-1}, x_t]$ zugeführt. Anschließend wird durch $(1 - z_t) * h_{t-1}$ bestimmt, welche Informationen nicht im Hidden-State h_t aktualisiert werden.

$$z_t = \sigma(W_z * [h_{t-1}, x_t])$$

Neu hingegen ist die Einführung eines Reset-Gates, welches bestimmt welche Informationen der vorherigen Zelle behalten oder verworfen werden. Dabei wird die entsprechende Gewichtsmatrix W_r mit den konkatenierten Vektoren $[h_{t-1}, x_t]$ multipliziert und letztendlich der Sigmoiden-Funktion σ zugeführt.

$$r_t = \sigma(W_r * [h_{t-1}, x_t])$$

Der Finale Zellstatus bzw. neue Hidden-State h_t wird somit durch das Zwischenergebnis \tilde{h}_t und dem Hidden-State h_{t-1} durch Vektoraddition bestimmt.

$$\tilde{h}_t = \tanh(W * [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Im Vergleich zu LSTM-Zellen besitzen GRUs weniger Operationen und konvergieren somit schneller zu einem Optimum. Nichtsdestotrotz hängt die Wahl der RNN basierten Architektur häufig von der zugrundeliegenden Problemstellung ab (vgl. Chung et al., 2014, S. 6f.) .

4.3 Keyword-Extraction-Methoden

Das folgende Kapitel verfolgt das Ziel ausgewählte Modelle der Keyword Extraction vorzustellen und zu deskribieren. Es fokussiert demnach die Erweiterung der bereits in der vorhergegangenen Forschungsarbeit aufgezeigten Methoden TFIDF, TFIGM und Text-Rank. Aufbauend hierauf lassen sich in aktueller Literatur eine Vielzahl an Keyword-Extraction-Algorithmen identifizieren. Die Auswahl der Algorithmen erfolgt hierbei auf den in der Literatur dargelegten Ergebnissen der jeweiligen Algorithmen sowie mit dem Fokus jeweils drei Algorithmen aus den Subkategorien der graphbasierten- sowie der Gewichtungsverfahren zu verwenden.

4.3.1 YAKE

CAMPOS ET AL. (2018) publizieren den Algorithmus YAKE und definieren fünf divergente Eigenschaften eines Wortes, die zur Bewertung dieser herangezogen werden. Hierbei

handelt es sich um statistische Größen, welche auf Basis der übergebenen Dokumente berechnet werden können und somit das Verfahren in die Kategorie der unüberwachten Keyword-Extraction-Methoden einordnen (vgl. Campos et al., 2018, S. 684).

Basierend auf der Annahme, dass groß geschriebene Wörter sowie Akronyme relevantere Informationen hinsichtlich des Textes beinhalten, führen die Autoren die Kennzahl W_{case} ein, die zur stärkeren Gewichtung dieser dient.

$$W_{case} = \frac{\max(TF(U(w)), TF(A(w)))}{\log_2(TF(w))} \quad (9)$$

Hierfür verwenden die Autoren das Maximum des Aufkommens eines Wortes als großgeschriebenes Wort $TF(U(w))$ oder eines Wortes als Akronym $TF(A(w))$ und setzen dies mit der logarithmierten Häufigkeit des Wortes im gesamten Dokument in Relation (vgl. Campos et al., 2018, S. 685).

Weiterhin inkludieren die Autoren die Position eines Wortes W_{pos} im Dokument in die Bewertung.

$$W_{pos} = \log_2(\log_2(2 + Median(Sen_w))) \quad (10)$$

Diese Kennzahl stellt durch den Wert Sen_w , der die Position der Sätze mit w definiert, sicher, dass Wörter, die früher im Korpus bzw. im Dokument auftreten eine höhere Relevanz erreichen. Gerade im Kontext der vorliegenden Arbeit und den damit einhergehenden wissenschaftlichen Publikationen ist anzunehmen, dass relevante Informationen über Problem- und Zielstellung bereits früh im Dokument erörtert werden (vgl. Campos et al., 2018, S. 685f.).

Als dritte Eigenschaft, setzen die Autoren das Maß der Häufigkeit eines Wortes W_{freq} ein, stellen diese allerdings in Relation mit der durchschnittlichen Häufigkeit von Wörtern addiert mit der Standardabweichung dieser. Weiterführend berechnen die Autoren inwieweit das betrachtete Wort die Charakteristiken eines Stopwords erfüllt und demnach weniger Relevanz aufweist.

$$W_{rel} = \left(0.5 + \left(\left(WL * \frac{TF(w)}{MaxTF} \right) + PL \right) \right) + \left(0.5 + \left(\left(WR * \frac{TF(w)}{MaxTF} \right) + PR \right) \right) \quad (11)$$

Hierbei beschreiben die Werte WL und WR die Relation von differenten Wörtern, die das Kandidatenwort umgeben zu der Gesamtanzahl dieser. PL und PR hingegen teilen

diese Menge an unterschiedlichen Wörtern im Kontext durch die maximale Termfrequenz im Dokument. Demnach erhalten Wörter mit stark variiertem Kontext höhere Werte dieser Kennzahl, da dies eine ähnliche Charakteristik zu Stopwords darstellt.

Als letztes Kriterium deskribieren CAMPOS ET AL. (2018) die relative Häufigkeit von Sätzen, die den Wortkandidaten beinhalten $W_{DiffSentence}$. Final führt dies zur Berechnung der Kandidatenbewertung mittels der untenstehenden Formeln, wobei (XX) für die Bewertung eines Wortes sowie (XX) zur Bewertung von N-Gramm-Kandidaten herangezogen wird.

$$S(w) = \frac{W_{rel} * W_{pos}}{W_{case} + \frac{W_{freq}}{W_{rel}} + \frac{W_{DiffSentence}}{W_{rel}}} \quad (12)$$

$$S(kw) = \frac{\prod_{w \in kw} S(w)}{TF(w) * (1 + \sum_{w \in kw} S(w))} \quad (13)$$

Abschließend zeigen Campos et al. (2018) in einem Vergleich dieser Methodik mit TFIDF, TextRank sowie RAKE starke Verbesserungen im Rahmen der Keyword Extraction.

4.3.2 PositionRank

In Addition zur graph-basierten Methode des TextRank nach MIHALCEA & TARAU (2004a), welches in vorangegangener Arbeit untersucht worden ist, ist der Algorithmus PositionRank nach FLORESCU & CARAGEA (2017) anzusprechen. Analog zu TextRank basiert der Algorithmus auf dem PageRank-Modell von PAGE, BRIN, MOTWANI & WINOGRAD (1999). Dabei liegt der allgemeine Vorteil von graph-basierten Verfahren bei der namensgebenden Verwendung von Graphen zur Repräsentation der Dokumentstruktur, da die Gewichtung der verschiedenen Knoten eines Graphs die Relevanz eines Knotens im gesamten Netzwerk wiederspiegelt (vgl. Mihalcea & Tarau, 2004a, S. 404).

Zur Selektion potentieller Schlüsselwörter extrahieren FLORESCU & CARAGEA (2017) Satzstrukturen, welche aus einem, keinem oder mehreren Adjektiven sowie mindestens einem darauffolgenden Nomen bestehen und somit dem regulären Ausdruck „(Adjektiv)*(Nomen)+“ entsprechen. Anschließend werden die entstehenden Kandidaten bzw.

deren Subwörter anhand des Bewertungsmechanismus bewertet, wobei die Summe dieser den Rang des gesamten Keywords abbildet (vgl. Florescu & Caragea, 2017, S. 1109).

Die Intention hinter PositionRank kristallisiert sich in der Positionierung des Wortes im Dokument heraus. Analog zu CAMPOS ET AL. (2018), basiert die Inkludierung der Wortpositionierung auf der Annahme, dass frühzeitig im Dokument auftretende Wörter mehr Informationsgehalt aufweisen und somit einen höheren Wert zur Extraktion der Semantik des Dokumentes innehaben. Zur Realisierung dieser Intention beziehen die Autoren alle inversen Positionen eines Wortes im Dokument mit in die Rangbewertung nach PageRank bzw. TextRank ein⁴. Hierbei berechnet sich der Wert der Positionsbewertung p eines Wortes an der zweiten, fünften und zehnten Position im Dokument wie folgt.

$$p = \frac{1}{2} + \frac{1}{5} + \frac{1}{10} = 0.8 \quad (14)$$

Die Summierung der einzelnen Werte dient zur additional stärkeren Gewichtung von hochfrequenten Wörtern bei zeitgleicher Beachtung deren Position. Abschließend erreichen die Autoren mithilfe dieser Integration signifikante Verbesserung der Extraktion gegenüber TextRank und TFIDF auf multiplen Datensets (vgl. Florescu & Caragea, 2017, S. 1108ff.).

4.3.3 TopicRank

Als ein weiterer Vertreter graphbasierter Verfahren wird TopicRank herangezogen, welches den Graphen anhand von Topics aufbaut anstelle von. Ähnlich wie zum TextRank-Algorithmus, besteht die grundlegende Intuition darin, einen Graphen aus den zugrundeliegenden Dokumenten aufzubauen, welcher die semantisch wichtigsten Textbausteine widerspiegelt. Im Kontrast zu klassischen Verfahren der Keyword Extraction, baut TopicRank den Graphen an zuvor erstellten Topics auf, welche durch ein spezifisches Clusteringverfahren erzeugt werden. Hierbei sind die Topics nicht durch einzelne Wörter des Corpus definiert, sondern durch die im Text enthaltenen Schlüsselwortphrasen Wörtern (vgl. Bougouin et al., 2013, S. 543ff.).

⁴ An dieser Stelle sei auf die vorangegangene Forschungsarbeit verwiesen, da die Wortbewertung nach TextRank hier nicht fokussiert wird.

Das Verfahren ist grundsätzlich als eine Erweiterung des TextRank-Algorithmus anzusehen, welches signifikante Verbesserungen im Bereich der Keyword Extraction aufzeigt (vgl. Bougouin et al., 2013, S. 548f.). Die folgende Abbildung erläutert die grundlegenden Schritte des Algorithmus, nach BOUGIN ET AL. (2013), bevor auf die detaillierte Berechnung des Graphen eingegangen wird.

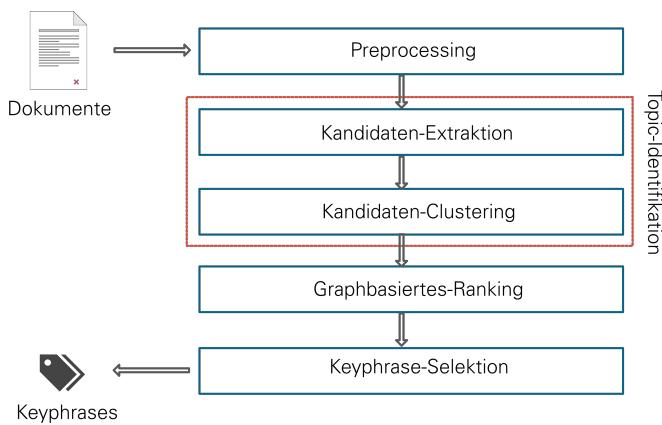


Abbildung 13: Verarbeitungsschritte des TopicRank-Algorithmus (Eigene Darstellung)

Bevor mit der eigentlichen Extraktion der relevanten Topics begonnen werden kann, wird der Textkorpus einer Datenvorverarbeitung unterzogen (Tokenization, POS-Tagging). Daraufhin beginnt die eigentliche Topic-Selektion, welche die längsten Nominalphrasen extrahiert und diese als potentielle Topic-Kandidaten behandelt⁵. Die automatisierte Gruppierung ähnlicher Schlüsselwortphrasen basiert hierbei auf einen agglomerativen Clusteringverfahren, mit dem Average-Linkage-Verfahren als Fusionierungsalgorithmus.

Basierend auf denen durch Clustering zusammengeführten Topics, erfolgt die Konstruktion des Graphen $G = (V, E)$, wobei V die Menge der Knoten und E die Menge der Kanten des Graphen repräsentiert. Hierbei repräsentieren Topics die einzelnen Knoten des

⁵ HULTH (2003) konstatiert, dass der Großteil von Schlüsselwortphrasen durch Nominalphrasen beschrieben werden. Aus diesem betrachtet TopicRank auch nur Nominalphrasen, um zur großen Topics zu vermeiden.

Graphen und die Kanten zwischen Topics t_i sowie t_j symbolisieren ihren semantischen Zusammenhalt. Dabei haben zwei Knoten eine starke semantische Beziehung, wenn diese dicht beieinander im Dokument liegen. Die Gewichte zwischen den Kanten ergibt sich daraus wie folgt:

$$w_{i,j} = \sum_{c_i \in t_i} \sum_{c_j \in t_j} dist(c_i, c_j)$$

$$dist(c_i, c_j) = \sum_{c_i \in t_i} \sum_{c_j \in t_j} \frac{1}{|p_i - p_j|}$$

Die Distanz $dist(c_i, c_j)$ ergibt sich dabei aus den Abständen der potentiellen Schlüsselwortphrasen eines Topics c_i und c_j im zugrundeliegenden Text.

Kommentiert [DH15]: Diskutieren

Im Kontrast zum klassischen TextRank-Algorithmus handelt es sich hierbei, um einen vollvernetzten Graphen, welcher einen besseren Gesamtüberblick über die Topic-Relationen liefert. Das Berechnen der relevantesten Schlüsselwortphrasen des Graphen erfolgt durchaus dem TextRank-Algorithmus bekannten Scoring-Verfahren:

$$S(t_i) = (1 - d) + d * \sum_{t_j \in V_i} \frac{w_{j,i} * S(t_j)}{\sum_{t_k \in V_j} w_{j,k}}$$

Wobei hier jedoch die Input- und Output-Beziehungen keine Rolle spielen, da es sich um einen ungerichteten Graphen handelt. Die Gewichtung einer Beziehung zwischen zwei Knoten t_i und t_j wird mit dem Gewicht w_{ij} symbolisiert. Abschließend repräsentiert d den sogenannten Dumping-Faktor, welcher die Wahrscheinlichkeit angibt von einem Knoten zum nächsten zu wandern. Der Standardwert dieses Faktors liegt bei 0.85 (vgl. Mihalcea & Tarau (2004), S. 404).

4.4 Ensemble Learning

Das Gebiet des Ensemble Learning (EL) ist als ein zentrales Forschungsareal im Machine Learning zu charakterisieren und untersucht die Möglichkeiten differente Algorithmen des ML zu kombinieren, um bessere Resultate zu erzielen. Hierbei liegt der Hauptfokus

auf prognostizierenden Methoden mit dem Ziel die Güte der Klassifikation durch die Beachtung differenter Modelle zu erhöhen. Diesbezüglich ist festzustellen, dass die Funktionalität dieses Vorhabens von der Unabhängigkeit der Fehler der einzelnen Classifier abhängt. Somit ist zu konstatieren, dass die Zusammenführung bei Modellen mit perfekt korrelierten bzw. identischen Fehlern keine Vorteile bringt, da die Vorhersage des EL-Modells lediglich die Ergebnisse der Classifier reproduziert. Im Allgemeinen sind die Methoden des Ensemble Learnings folgend der Taxonomie von RE & VALENTINI (2012) in nicht-generative sowie generative Methoden zu unterteilen. Nicht-generative Methoden umfassen hierbei diejenigen, welche die Ergebnisse bestehender Klassifikationsalgorithmen kombinieren und hieraus eine finale Vorhersage ableiten. Generative Methoden hingegen deskribieren Algorithmen, welche additional auf das Training der divergenten Modelle fokussiert sind und somit, durch Anpassung der Algorithmen oder der Trainingsdaten, die Genauigkeit dieser zu erhöhen versuchen (vgl. Goodfellow et al., 2016, S. 256f.; Re & Valentini, 2012, S. 567, 572).

Unter dem Gesichtspunkt der Anzahl an differenten Methoden, welche zur Einordnung von Problembeschreibungen im Kontext der aktuellen Untersuchung verwendet werden können, bietet das EL einen Ansatzpunkt zur Kombination dieser. Gerade hinsichtlich der stark differierenden Herangehensweisen zur Zielerreichung, der geringen Datenmenge sowie der unter **Kapitel XX** dargelegten Problematik unterschiedlicher linguistischer Strukturen von Trainings- und Anwendungsdaten, kristallisiert sich dieses Areal als vielversprechend heraus. Aufgrund dessen dient das folgende Kapitel zur Deskription von ausgewählten Methoden des EL. Hierbei ist zu erwähnen, dass auf nicht-generative Methoden sowie die generativen Methoden Bagging und Boosting eingegangen wird, diese allerdings nicht die Gesamtheit der EL-Methoden darstellen. Für eine umfangreichere Auflistung und Beschreibung derartiger Methoden sei auf die Publikation von RE & VALENTINI (2012) verwiesen, welche eine generelle Übersicht bereitstellen.

4.4.1 Nicht-generative Methoden des Ensemble Learnings

Die nicht-generativen Methoden des EL lassen sich weiterhin in Fusions- und Selektionsmethoden unterteilen. Im Folgenden werden primär erstere fokussiert, da Selekti-

onsmethoden das Ziel verfolgen den besten Algorithmus aus einer Sammlung an Algorithmen zu extrahieren. In der vorliegenden Untersuchung hingegen bestehen potentielle Vorteile durch die tatsächliche Zusammenführung der applizierten Methoden.

Zunächst ist das Prinzip der **Majority Vote** anzuführen, welches auf Basis aller Vorhersagen der Classifier die Mehrheit dieser als finales Ergebnis heranzieht. Hierbei sind Problematiken bei ungleichverteilter Modellgüte der Algorithmen sowie bei starker Unterscheidung der Vorhersagen festzustellen, da diese die Verlässlichkeit der finalen Auswahl negativ beeinflussen. Als zweiter, intuitiver und weit verbreiteter Ansatz ist auf das Verfahren des **Weighted majority voting** zu verweisen. Motiviert durch different ausgeprägte Gütewerte der Classifier werden die Vorhersagen dieser hierbei mit Gewichten belegt um diese in die Berechnung der finalen Vorhersage einfließen zu lassen. Häufig wird hierbei die Güte, beispielsweise in Form der Kennzahl der Accuracy, als Gewicht herangezogen (vgl. Kuncheva & Rodríguez, 2014, S. 262f.).

Neben weiteren arithmetischen Methoden wie beispielsweise die **Summenbildung**, die **Produktbildung** oder die Verwendung von **(gewichteten) Mittelwerten** der Ergebnisse existieren in diesem Areal weitere Methoden, die das Training eines additionalen Classifiers auf Basis der Ergebnisdaten fokussieren. Hierbei können Algorithmen wie beispielsweise die Naive Bayes Klassifikation oder Support Vector Machines verwendet werden.

4.4.2 Generative Methoden des Ensemble Learnings

Im Folgenden wird nach POLIKAR (2012) mit den Methoden **Bagging** und **Boosting** auf zwei der populärsten generativen Methoden des EL eingegangen. Insgesamt sind in aktueller Literatur eine Vielzahl weiterer Methoden zu identifizieren, welche hier aufgrund des Rahmens der vorliegenden Arbeit jedoch vernachlässigt werden.

Bagging (Bootstrap Aggregating)

Wie die Kategorie der generativen Methoden bereits aufzeigt, setzt das Bagging bereits in der Trainingsphase von Classifiern an und definiert die Variation der Trainingsdaten. So werden im Rahmen des Bagging n differente Trainingsdatensätze für n Classifier erstellt. **Abbildung XX** dient zur Illustration des Bagging-Ansatzes.

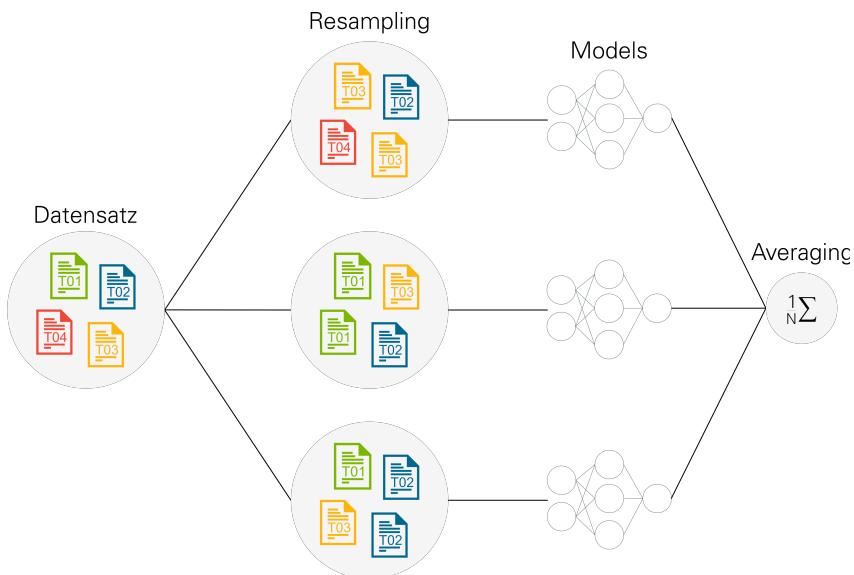


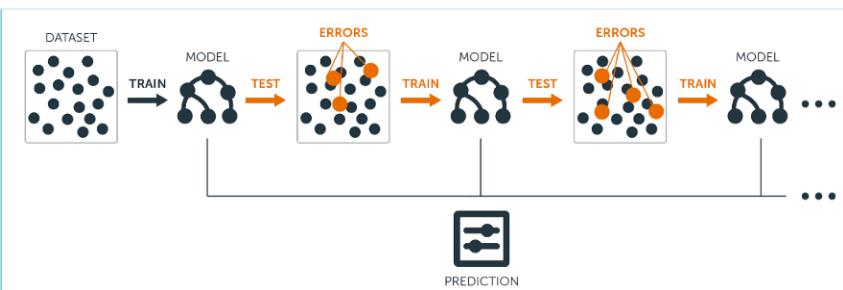
Abbildung 14: Illustration der Funktionsweise des Bagging-Ansatzes (Eigene Darstellung)

Final umfasst jeder dieser Datensätze die gleiche Anzahl an Elementen wie der originale und wird mittels sog. Sampling mit Ersetzung generiert (vgl. obige Abbildung). Sampling definiert das Vorgehen zur wahrscheinlichkeitsbasierten Vervielfachung von Elementen im originalen Datensatz. So werden differente Elemente mehrfach in den resultierende Datensatz integriert. Im aktuellen Fall werden hierbei andere Elemente durch diese Kopien ersetzt, was die Generierung unterschiedlicher Datensätze identischer Größe erlaubt. Diese Variation der Daten verfolgt das Ziel die oben angesprochene Unabhängigkeit der Fehler und somit unterschiedliche Modelle als Resultat der Trainingsphase zu erreichen. In Anschluss an diese Phase werden, wie der Name Bootstrap Aggregating bereits erkennen lässt, die finalen Ergebnisse durch den Mittelwert der einzelnen Modellergebnisse berechnet (vgl. Goodfellow et al., 2016, S. 257; Polikar, 2012, S. 11f.).

Boosting

Im Kontrast zu Bagging-Methoden, werden bei einer auf Boosting basierenden Strategie, verschiedene schwache Klassifikatoren zu einem starken Klassifikator zusammen-

geführt. Meist sind diese schwache Klassifikatoren einfach aufgebaut und berücksichtigen nur ein Merkmal des zugrundeliegenden Datenpools (vgl. Polikar, 2012, S. 35). Hierbei handelt es sich i.d.R. um ein sequentielles Vorgehen, um die multiple schwache Klassifikatoren zu kombinieren.



Kommentiert [DH16]: Abbildung machen

Abbildung 15: Vorgehen bei Boosting (Eigene Darstellung)

Wie in **Abbildung XYZ** illustriert, wird ein initiales Modell auf dem Trainingsdatensatz trainiert und getestet. Ausgehend von den falschen Vorhersagen sowie den additionalen zufälligen Datenpunkten aus dem ursprünglichen Datensatz, wird das darauffolgende Modell trainiert. Fundierend auf den falsch klassifizierten Datenpunkten, werden somit Modelle trainiert, welche speziell auf den Fehlern des Vorgängermodells trainiert werden.

Das Verwenden einer Boosting basierenden Strategie führt in der Regel zu besseren Ergebnissen, als Bagging basierte Ansätze und gilt als State-of-the-Art-Methode, um Ensemble-Learning-Modelle zu modellieren (vgl. Raphael Campos, Canuto, Salles, de Sá, & Gonçalves, 2017, S. 114). Allerdings, benötigen Boosting basierende Strategien ähnlich viele Daten wie Bagging-Methoden, um ausreichend schwache Klassifikatoren, auf den Fehlern der Vorgängermodelle zu trainieren. Hinzukommt, dass diese Methoden häufig dazu tendieren die Trainingsdaten auswendig zu lernen und sehr stark auf Ausreißer reagieren, was das Parametertuning dieser als essentiell gestaltet (vgl. B.Meshram & M. Shinde, 2015, S. 7). Aus Komplexitätsgründen und der fehlenden Datenbasis des aktuellen Forschungsvorhabens, wird dieser Ansatz nicht weiter in der Implementation des Prototypes betrachtet, jedoch vollständigkeitshalber mit aufgelistet.

4.5 Named-Entity Recognition

Das Areal der Named-Entity Recognition (NER) beschreibt ein Subareal der Information Extraction (IE) und fokussiert die Extraktion von Entitäten aus Textdaten. Globale Systeme finden dabei Entitäten, die generischen Konstrukten wie Personen, Organisationen, Orten oder Zeitangaben zuordenbar sind. Dennoch zeigen GRIDACH (2017) und MCCALLUM (2005) die Adaption in spezifischen Domänen mit dem Ziel Gene, Proteine und andere Biomedizinische Entitäten oder Namen von Hochschulkursen auszulesen. Das Generelle Vorgehen in diesem Areal ist zunächst die Extraktion von Textstellen, welche Entitäten repräsentieren um diese anschließend, unter der Hinzunahme von zusätzlichen Attributen, wie der Schreibweise oder deren POS-Tags, zu klassifizieren (vgl. Jurafsky & Martin, 2008, S. Ch. 22, S. 2). GOYAL, GUPTA & KUMAR (2018) deskribieren in ihrer Forschungsarbeit verschiedene Techniken, welche zur NER verwendet werden. So unterscheiden die Autoren zwischen regelbasierten- und lernbasierten Methoden, wobei sich letztere zudem in die Kategorien des überwachten, unüberwachten sowie semi-überwachten Lernens unterteilen. EFTIMOV, SELIAC & KOROŠEC (2017) spezifizieren weiterhin wörterbuchbasierte Vorgehen als weitere Kategorie von NER-Methoden.

4.5.1 Regelbasierte und wörterbuchbasierte Vorgehen

Wörterbuchbasierte Methoden definieren im Allgemeinen die Anwendung einer Wissensbasis als semantische Ressource, welche potentiellen Texten gegenübergestellt werden kann. Hierbei beinhaltet diese Ressource spezifische Terminologien der fokussierten Domäne sowie deren Synonyme, weshalb dieses Verfahren primär in stark abgegrenzten Domänen Anwendung findet. Zur Verbesserung der Güte dieses Vorgehens werden zur Gegenüberstellung der Ressource und neuen Textinhalten, neben dem strikten Vergleich der Zeichenketten, differente Heuristiken, wie bspw. die Permutation von Wörtern, angewendet. Aufgrund der Domänenunabhängigkeit des hier zu entwickelnden Artefakts und der resultierenden nicht gegebenen Anwendbarkeit von wörterbuchbasierten Methoden werden diese Heuristiken allerdings nicht weiter vertieft. Im Allgemeinen erweisen sich wörterbuchbasierte Methoden als Inflexibel und bieten den Nachteil, dass lediglich im Vorfeld spezifizierte Entitäten aus neuen Texten extrahiert werden können (vgl. Eftimov et al., 2017, S. 5).

Regelbasierte Verfahren im Gegenzug spezialisieren sich auf die Nutzung von regulären Ausdrücken (vgl. **Kapitel XX**), mittels denen terminologische Informationen und Charakteristiken der Entitäten inkludiert werden können. So werden Ausdrücke definiert, welche Muster wiederspiegeln die charakteristisch für die zu extrahierende Entitäten sind, wodurch domänen spezifisches Wissen inkludiert werden kann. Zeitgleich beschreibt dies den Nachteil dieser Methodik, da die Regeln eine schlechte Generalisierungskraft aufweisen und somit lediglich hinsichtlich einer speziellen Anwendung Funktionalität sichern können. Weiterhin setzt die Erstellung derartiger regulärer Ausdrücke die Kenntnis der linguistischen Struktur der verwendeten Sprache voraus, weshalb diese Methode ebenfalls auf diese Sprache restringiert ist. Summierend bietet diese Methodik, durch den Einbezug von domänen spezifischen Informationen, eine gute Möglichkeit zur Extraktion von Entitäten, ist allerdings auf eine Domäne bzw. Anwendung und Sprache beschränkt. Resultierend aus diesem Sachverhalt existieren in aktueller Literatur viele Methoden im Bereich der lernbasierten Methoden (vgl. Goyal et al., 2018, S. 25; Nadeau & Sekine, 2007, S. 6).

Kommentiert [m17]: Datenselektion und reinigung

4.5.2 Lernbasierte Verfahren der NER

Kommentiert [DH18]: Hybride Systeme in ein zwei sätzen mit rein

Im Vergleich zu den oben genannten Methoden basieren lernbasierte Verfahren auf Machine-Learning-Algorithmen, um automatisch Textentitäten zu extrahieren (**Quelle**). Dabei erzielen diese Verfahren i.d.R. besser Performance als klassische Wörterbuch oder regelbasierte Vorgehen (vgl. Goyal et al., 2018, S. 26). Grundlegend lassen sich lernbasierte Verfahren in vier Kategorien unterteilen, welche nachfolgend kurz charakterisiert werden.

Fundierend auf annotierten Daten, extrahieren überwachte NER-Verfahren die Entitäten eines Textes. Dabei sind die Daten mit positiven und negativen Entitäten annotiert, um eine bessere Diskriminierung zwischen tatsächlichen Entitäten und nicht-Entitäten zu erzeugen (vgl. Goyal et al., 2018, S. 25). Prinzipiell fußen diese Verfahren auf überwachten ML-Verfahren und folgen einem klar definierten iterativen Schema, welches in der nachfolgenden **Abbildung XYZ** illustriert ist.



Kommentiert [DH19]: Lesen und eventuell raus außer man macht es mit „Entity Extraction“

Wie zu erkennen, orientiert sich der Ablauf an einem klassischen Vorgehen überwachter ML-Verfahren. Allerdings ist der Schritt der Feature Extraction als essentiell bei der Extraktion von Textentitäten anzusehen, um die multidimensionalen Aspekte derer getreu zu erfassen. Dabei unterscheiden sich die zu extrahierenden Eigenschaften in drei differente Gruppen. Wörterbuch basierende Eigenschaften dienen als Look-Up-Tabelle, um Wortzugehörigkeiten zu erfassen. Hinzukommen Dokumenteneigenschaften, welche strukturelle und syntaktische Eigenschaften erfassen, um die Textentitäten eines zu grundeliegenden Korpus zu extrahieren. Die wortbasierten Eigenschaften konzentrieren sich beispielsweise darauf, semantische Aspekte einzelner Wörter zu inkludieren. Somit unterscheiden sich lernbasierte NER-Verfahren auch zu anderen NLP-Methoden, wie beispielsweise Embeddings, da diese nur wortbasierte Eigenschaften beachten. Für das letztendliche Training können verschiedene ML-Verfahren zur Anwendung kommen. So benutzen SAHA ET AL. (2010) eine SVM, um Entitäten aus einem Text zu extrahieren. Erweiterte NER-Systeme nutzen jedoch verschiedene schwergewichtige Modelle entlang des illustrierten Prozesses. Beispielsweise können bidirektionale LSTMs dazu verwendet werden, Dependenzbeziehung zu extrahieren und diese Dokumenteneigenschaften für das spätere Training des Klassifikators verwendet werden (vgl. Kiperwasser & Goldberg, 2016, S. 323).

Neben dem überwachten Verfahren etablieren sich auch unüberwachte Verfahren der NER, welche sich hauptsächlich auf Clustering- und Assoziationsanalysen stützen. Zum einen appliziert man Clustering-Verfahren, um kontextuelle Ähnlichkeiten einzelner Entitäten zu extrahieren. Zum anderen werden Assoziationsanalysen dafür genutzt, um Beziehungen zwischen potentiellen Entitätskandidaten zu entdecken und diese anhand verschiedener Algorithmen zu bewerten (vgl. Goyal et al., 2018, S. 26).

Eine Kombination beider eben genannter lernbasierter Verfahren bilden Semi-Supervised-Ansätze. Hierbei wird aus einer Mischung von unüberwachten und überwachten Verfahren ein NER implementiert. Die Intuition hierbei ist es, ein überwachtes NER-Modell zu verwenden, welches einen kleinen Teil annotierter Daten verwenden, um die restlichen nicht-annotierten Daten zu annotieren. Beispiele hierfür liefern THENMALAR ET AL. (2015).

Nichtsdestotrotz ist das Training eines angemessenen NER-Modells mit vielen Herausforderungen verbunden. Beispielsweise ist das annotieren der Daten mit einem exorbitanten Aufwand verbunden. Zwar gibt es große Datensätze annotierter Daten, allerdings

fehlen diesen, domänen- und sprachspezifische Konstrukte. Hinzukommen andere Herausforderungen wie verschachtelte Entitäten oder die Mehrdeutigkeit der natürlichen Sprache, welche die Entwicklung eines NER-Systems erschweren (vgl. Goyal et al., 2018, S. 23).

Aus diesem Grund beschäftigt sich die vorliegende Arbeit mit einem rein regelbasierten Ansatz, um potentielle Analyseentitäten aus den vorliegenden Problemstellungen zu extrahieren. Das detaillierte Vorgehen wird im folgenden **Kapitel XYZ** deskribiert.

5 Prozess der Datenbeschaffung

Das folgende Kapitel verfolgt das Ziel eine Übersicht über die durchgeführten Schritte zur Beschaffung von Daten bzw. zu deren Selektion zu geben und fokussiert demnach die Beantwortung der Forschungsfrage I. Übergeordnet wird in der vorliegenden Arbeit, der in **Abbildung XX** dargestellte Prozess verfolgt, welcher weiterführend deskribiert wird.



Abbildung 16: Prozess der Datenbeschaffung und -selektion (Eigene Darstellung)

Wie in obenstehender Abbildung zu erkennen erfolgt zunächst das Crawling von wissenschaftlichen Datenbanken zur Schaffung einer Ausgangsbasis. Anschließend werden die gefundenen Daten überprüft und selektiert. Dies dient zum einen dem Zweck Rauschen in den Daten zu minimieren sowie zum anderen zur Auswahl von adäquaten Sätzen und Textteilen, die im Rahmen von vokabular- und häufigkeitsbasierten Herangehensweisen fokussiert werden. Abschließend finden mehrere Methoden der Data Augmentation Anwendung um die Datenbasis additional künstlich zu erweitern und die finalen Datensätze zu kreieren.

5.1 Crawling von Datenbanken

Zunächst werden zur Generierung einer Datengrundlage differente Datenbanken automatisiert mittels Crawlern nach bestimmten Suchbegriffen durchsucht, und die Resultate dieser Suche in Form von Textdateien gesichert. Generell beschreiben Crawler Programme bzw. Systeme, welche das Internet strukturiert nach Informationen durchsuchen, diese sammeln und es somit ermöglichen die rapid wachsende Menge an Inhalten

im Internet leichter zu assimilieren. Dies wird durch das Auslesen von Website-Quelltexten sowie der Folge von Hyperlinks zwischen Seiten erreicht (vgl. Dhenakaran & Thirugnana, 2011, S. 265). Im aktuellen Kontext werden die Datenbanken „ArXiv“ der Universität Stanford, der Verbund an Datenbanken von „Elsevier“ sowie das Portal „Medium“ durchsucht. Hierbei ist anzumerken, dass für erstere Datenbanken Python-Programmierschnittstellen der jeweiligen Organisationen Verwendung finden, wohingegen „Medium“ mittels eines klassischen Crawlers durchsucht wird.

Zu Beginn werden hierfür differente Suchbegriffe definiert, welche im Rahmen der technischen Möglichkeiten zwischen den differenten Datenbanken standardisiert werden. Aufgrund der Fokussierung auf Textdaten, welche Informationen über die differenten Algorithmenklassen wie beispielsweise „*Clustering*“ oder „*Classification*“ beinhalten, finden diese kategorischen Namen direkt als Suchbegriffe Anwendung. Zur Filterung der Ergebnisse werden diese lediglich in den Feldern „Title“, „Abstract“ und „Keywords“ appliziert und mit dem Wort „*data*“ kombiniert um Veröffentlichungen, welche diese Begriffe in anderem Kontext verwenden zu ignorieren. Weiterhin wird spezifiziert, dass der Titel der Veröffentlichung ebenfalls die jeweilige Algorithmusklasse beinhalten muss. Aufgrund der Kombination der Klassen „*Regression*“ und „*Classification*“ zur Klasse „*Prediction*“ wird im Kontext der betroffenen Suchanfragen die Option integriert, dass das Wort „*Prediction*“ im Titel des Dokumentes auftreten kann. Hieraus entstehen exemplarisch folgende Suchanfragen an die Datenbanken:

1. *Title – Abstr – Key(Clustering + AND + data) + AND + Title(Clustering)*
2. *Title – Abstr – Key(Classification + AND + data) + AND + Title(Classification + OR + Prediction)*

Additional werden Suchanfragen konstruiert, welche direkt auf Definitionen der Algorithmen abzielen. Dies resultiert aus der Verwendung von Algorithmen des Topic Modeling sowie der Keyword Extraction im Rahmen der Implementierung, da diese durch ihren Fokus auf das verfügbare Vokabular und Häufigkeiten von Wörtern eine starke Abhängigkeit zur Qualität der bzw. zum Rauschen in den Daten aufweisen. Hierfür werden die

Suchanfragen um den Term „*is*“ im ersten Teil der Query erweitert. Exemplarisch ergeben sich somit weiterhin die folgenden Anfragen⁶:

3. *Title – Abstr – Key(Clustering + is + AND + data) + AND + Title(Clustering)*

4. *Title – Abstr – Key(Classification + is + AND + data) + AND + Title(Classification
+ OR + Prediction)*

Final werden die jeweiligen Ergebnisse der Suchanfragen mittels Jupyter Notebooks festgehalten und dokumentiert. **Abbildung XX** zeigt einen Ausschnitt dieser Dokumentation und stellt die Quellenanzahl pro Publikationsjahr für das Crawling der Elsevier-Datenbanken dar.

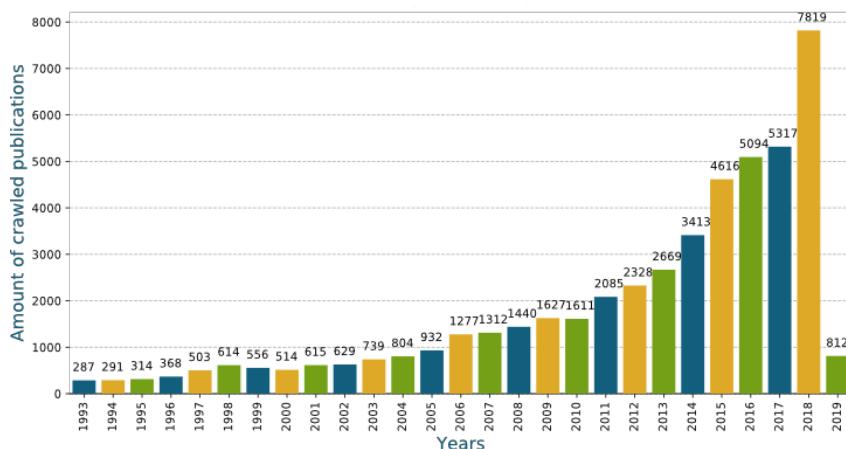


Abbildung 17: Crawling-Ergebnisse in Elsevier (Eigene Darstellung)

Summierend werden, bereinigt von Duplikaten, mittels dieses Vorgehens ca. 90.000 wissenschaftliche Veröffentlichungen bzw. deren Abstracts erlangt. Anschließend an die Schaffung einer Datenbasis gilt es die gefundenen Dokumente zu evaluieren und unbrauchbare Daten zu exkludieren.

⁶ Eine komplette Auflistung der verwendeten Suchanfragen findet sich im Anhang unter XXX

5.2 Datenevaluierung und -selektion

Generell ist zu konstatieren, dass die Menge der Daten, sowie die zugrundeliegende thematische Vielfalt der abgefragten Datenbanken eine nachfolgende Datenbereinigung dieser begründet. Hierbei wird zunächst ein rein auf Regeln und Logik basierter Datenbereinigungsschritt erläutert, um daraufhin einen auf Noise-Clustering basierten Ansatz zum Entfernen von Ausreißern zu deskribieren.

5.2.1 Regelbasierte Datenbereinigung

Um die Texte von unnötigem Rauschen zu bereinigen, empfiehlt sich ein rein Logik basierter Ansatz zur Vorverarbeitung der Daten. Im Rahmen des NLP werden hierbei häufig Technologien wie Stemming, Lemmatization, das Entfernen von Stopwords oder reguläre Ausdrücke verwendet, um semantisch irrelevante lexikalische Bausteine aus dem Text zu entfernen (vgl. Jurafsky & Martin, 2008, S. 2f.). Allerdings empfiehlt sich bei der Verwendung von distribuierten Repräsentation nur eine sehr verhaltene Datenvorverarbeitung (vgl. Jose Camacho-Collados & Pilehvar, 2018, S. 44).

Basierend auf den extrahierten Daten in **Kapitel XYT**, wird im Rahmen der Arbeit, Textrauschen im Folgenden als alle Duplikate, Zahlen, Abkürzungen, einzelne Buchstaben, Sonderzeichen und Leerraumzeichen definiert. Des Weiteren werden Formeln sowie Variablen (XX) mithilfe von regulären Ausdrücken aus dem Text gefiltert.

Kommentiert [DH20]: Unten die zwei Formeln

`r"(\s +\$ +\d +\$*)|(\s +\$ *\d +\$+)"` (15)

`r"(w *\W +\d+)"` (16)

Hinzukommt, dass Dokumente, welche eine Länge von sieben Satzzeichen unterschreiten automatisch aussortiert werden. Additional werden Sätze auf ihren relativen Anteil an Rauschen überprüft und bei einem Schwellwert von über 45% entfernt.

Durch die angewandte regelbasierte Reinigung sind insgesamt **XXYZ Dokumente** aus dem Korpus entfernt worden.

5.2.2 Entfernen von Ausreißern

Wie anfänglich bereits erwähnt ist anzunehmen, dass aufgrund der Datenvielfalt erhebliche semantische Divergenzen zwischen den Dokumenten einer Zielklasse auftreten können. Wie in der untenstehenden Abbildung deutlich zu erkennen, reicht das alleinige Entfernen mittels regulärer Ausdrücke nicht aus, um irrelevante Dokumente aus dem Datenpool zu entfernen. Mithilfe der in **Kapitel XYZ** angesprochenen distribuierten Paragraphrepräsentation, lassen sich die einzelnen Dokumente in dicht besetzte Vektoren überführen. Die grundlegende Eigenschaft von solchen distribuierten Repräsentationen ist, dass nahestehende Vektoren zueinander mehr semantische Ähnlichkeit aufweisen, als Vektoren die eine größere Distanz zueinander besitzen (vgl. P. Wang et al., 2016, S. 808). Die untenstehende **Abbildung 18** illustriert den zugrundeliegenden Datenpool, auf Grundlage der im **Kapitel XYZ** beschriebenen Deep-Averaging-Networks.

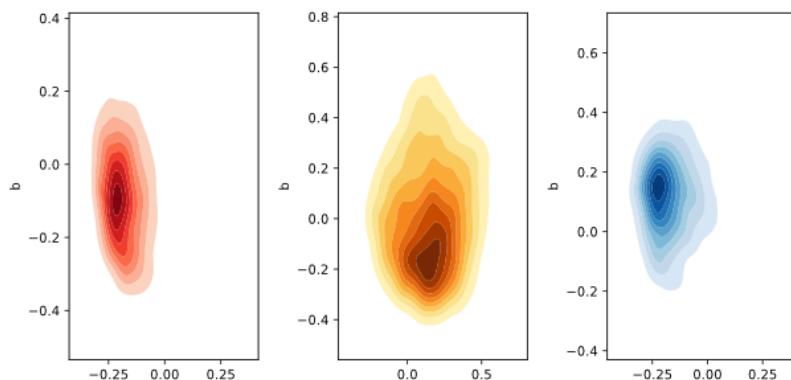


Abbildung 18: Density Graph des Datenpools (Eigene Darstellung)

Die oben illustrierten Diagramme veranschaulichen die Datenverteilung der einzelnen Dokumente, in ihrer distribuierten Form. Die dunkleren Areale symbolisieren ein relatives dichtes Aufkommen semantisch ähnlicher Dokumente und die helleren, dünner besetzten Flächen, die mit einer höheren semantischen Diskrepanz zur Ausgangsklasse. Interpretierbar ist, dass ausgehend von der distributionellen Hypothese, die dicht besetzten Areale die Bedeutung der Zielklasse besser verinnerlichen als dünn besetzte. Verdeutlicht ist dies analog in den untenstehenden Streudiagrammen.

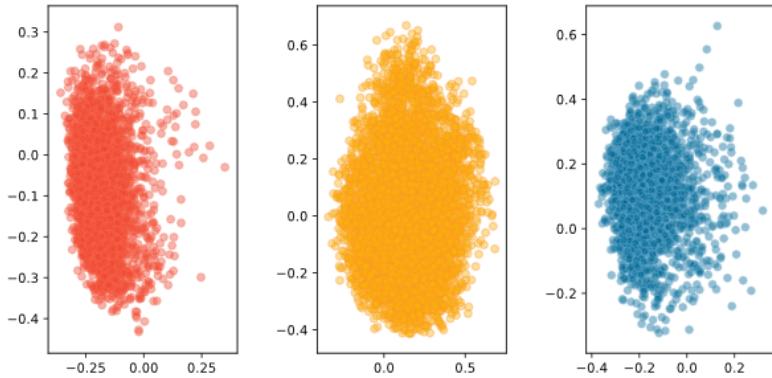


Abbildung 19: Streudiagramme des Datenpools (Eigene Darstellung)

Bedingt durch die semantische Sensitivität der Embeddings, können die Ausreißer darauf aufbauende Machine-Learning sowie Klassifikationsmodelle negativ beeinflussen (vgl. Rehm, Klawonn, & Kruse, 2007a, S. 489; P. Wang et al., 2016, S. 806). REHM ET AL. (2007) beschreiben das Entfernen von Ausreißern sogar als essentiellen Schritt, um ein adäquates Vorhersagemodell zu entwerfen.

Das Gruppieren von ähnlichen Textdaten ist Gegenstand des Text Clustering, welches sich verschiedenen Clustering-Methoden bedient, um ähnliche Textdokumente zueinander zu gruppieren (vgl. Liu, Liu, Chen, & Ma, 2003, S. 488). Als ein Teilgebiet des Clustering lässt sich die Kategorie des Noise-Clustering (NC) identifizieren. Dabei wird im Laufe des Clustering-Prozesses ein Noise-Cluster gebildet, welches alle Datenpunkte beinhaltet, die sich keiner Klasse zuordnen lassen. Die Idee lässt sich auf verschiedene Clustering-Algorithmen übertragen, wobei deren Anwendung vom jeweiligen Anwendungsfall abhängt (vgl. Dave, 1991, S. 657–658; Rehm et al., 2007a, S. 458).

Als NC-Algorithmus wird im Folgenden das dichtebasierte Verfahren DBSCAN verwendet, um semantisch irrelevante Elemente zu identifizieren. Das Verfahren verspricht effizient Cluster beliebiger Form zu identifizieren, sowie Ausreißer in einem eigenen Noise-Cluster zu separieren (vgl. Ester, Kriegel, Sander, & Xu, 1996, S. 228). Die grundlegende Idee des Algorithmus ist es, dass für jeden Datenpunkt im Radius Eps eine mi-

nimale Anzahl an Punkten (*MinPts*) vorhanden sein muss, um zu einem Cluster zu gehören (vgl. Ester et al., 1996, S. 228–230). Auf eine ausführliche Erläuterung des Algorithmus soll im Rahmen der Arbeit aus Komplexitätsgründen verzichtet werden.

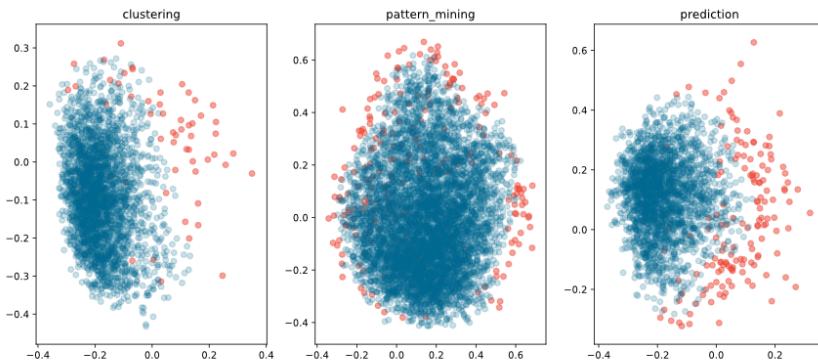


Abbildung 20: Durch DBSCAN identifizierte Ausreißer (Eigene Darstellung)

Die obige Abbildung illustriert die gefundenen Ausreißer. Anzumerken ist hierbei, dass die hohe initiale Dimensionalität der distribuierten Textrepräsentationen ein NC als diffizil gestaltet. Aus diesem Grund sind die Datenpunkte mithilfe des Verfahrens der *Principle Component Analysis* (PCA) auf 3 Dimensionen reduziert worden und anschließend dem DBSCAN-Verfahren unterzogen. Ein zugrundeliegender Streukoeffizient von **XYZ** zeigt dabei, dass die ursprüngliche Streuung der Daten gut wiedergespiegelt wird.

Wie die roten Datenpunkte veranschaulichen, erlaubt das Verfahren gut semantische Ausreißer zu filtern. Die Sensitivität des DBSCAN-Verfahrens gegenüber den Parametern *Eps* und *MinPts*, macht es schwer alle Ausreißer exakt zu filtern. Additional ist zu erkennen, dass in seltenen Fällen Datenpunkte innerhalb der Dichtezentren erkannt werden, was ebenfalls auf die komplexe Parameterauswahl des Verfahrens zurückführen ist (vgl. Halkidi, Batistakis, & Vazirgiannis, 2001, S. 21).

Kommentiert [m21]: I hope so xD

Kommentiert [DH22R21]: stimmt

Ergebnisse und Evaluation

Das Ziel der aufgezeigten Methodik, um Ausreißer zu filtern, dient zum einen semantische in sich geschlossene Klassen zu erzeugen. Zum anderen soll jedoch auch die Separierbarkeit zwischen den Klassen erhöht werden, um eine sich anschließende Textklassifikation zu erleichtern.

Kommentiert [m23]: Keyword: Hyperspectral Data. Vielleicht findest du sowsas, dass mit zunehmender Dimensionalität die Separierbarkeit der Klassen erschwert wird.

Der *Davis-Bouldin-Index* ist eine Clustering-Validationsmetrik, welche alleinig auf den Clustern selbst basiert und nicht auf der Basis annotierter Daten. Der Index ist ein Indikator dafür, wie gut Cluster separierbar sind. Ein geringes Ergebnis lässt dabei auf eine bessere Separierbarkeit der Klassen schließen. Die grundlegende Intention sei in der nachfolgenden Formel veranschaulicht:

$$DB = \frac{1}{K} \sum_{i=1}^k \max_{j=1, \dots, k, i \neq j} \left(\frac{diam(c_i) + diam(c_j)}{d(z_i, z_j)} \right)$$

$$diam(c_i) = \sqrt{\frac{1}{n_i} \sum_{x \in c_i} d(x, z_i)^2}$$

Hierbei beschreibt k die Anzahl der zugrundeliegenden Klassen, n die Anzahl aller Punkte einer Klasse, z_i den Zentroid einer Klasse c_i sowie $d(x_i, x_j)$ die Distanz zwischen zwei Punkten. Hinzukommt, dass der Ausdruck $diam(c_i)$ die gemittelte euklidische Distanz aller Punkte eines Clusters zu ihrem Klassenschwerpunkt darstellt. Vereinfacht lässt sich konstatieren, dass der Ausdruck die gemittelte Gleichheit eines Clusters c_i und seinem ähnlichsten Cluster c_i widerspiegelt.

Der Index wird im Rahmen der NC dazu verwendet die beste Parameterkombination des DBSCAN-Verfahrens zu ermitteln. Die nachfolgende Tabelle hält einen Ausschnitt der Ergebnisse fest.

Tabelle 2: Festlegen der Clustering-Parameter

Iteration	Eps	MinEps	Dave-Bouldin-Index
21			
22			
23			

Kommentiert [m24]: Tabelle!

Hierbei ist iterativ eine Kombination aus den beiden Parametern *Eps* und *MinPts* getestet worden, um anschließend den *Davis-Bouldin-Index* als Entscheidungsgrundlage heranzuziehen. Die finale Parameterkombination des NC-Verfahrens wird somit durch den geringsten Dave-Bouldin-Index bestimmt.

Neben der algorithmischen Evaluation der Daten empfiehlt sich eine manuelle Begutachtung der Daten. So wird der zugrundeliegende Text der Ausreißer stichpunktartig auf

semantische Aussagekraft überprüft. Die semantisch sinnvollen Texte werden demzufolge aus dem Noise-Cluster entfernt und erneut der Datenbasis zugeführt. Nichtsdestotrotz ist dieser Evaluationsansatz rein subjektiver Natur und soll nicht näher deskribiert werden.

5.3 Data Augmentation

Anschließend an die Bereinigung der Ausgangsdaten erfolgt die Expansion dieser mittels Methoden der Data Augmentation. Das Areal der Data Augmentation umfasst Methoden, welche zur Reduktion der Gefahr des Overfitting von Machine-Learning-Modellen dienen, indem die Datenmenge künstlich expandiert wird und somit die Variabilität des Datensatzes erhöht wird. Gerade im Bereich der Computer Vision bzw. der Bildverarbeitung mittels Deep Learning erreichen diese Methoden hohe Popularität und werden häufig appliziert. Hierbei werden Bilddateien, beispielsweise durch horizontale und vertikale Spiegelung, Änderung der Farbgebung, Rotation, Ausschneiden von Bildelementen oder Manipulation von Pixeln variiert um robustere und besser angepasste Modelle zu erreichen (vgl. Perez & Wang, 2017, S. 2ff.).

Oppositionell zum Areal der Computer Vision ist im Bereich des NLP keine Popularität von Data Augmentation zu vermerken. Dies resultiert daraus, dass ein Großteil der Augmentationsmethoden die Variation von unabhängigen Eigenschaften, wie beispielsweise den Pixeln eines Bildes, der bestehenden Daten fokussieren und somit keine eventuell vorherrschenden Zusammenhänge verändert werden. Dieses Vorgehen ist bei einer textuellen Datenbasis, durch die zugrundeliegende Semantik, bestehende Relationen sowie die Ambiguität von Sprache als diffizil zu charakterisieren, da die Änderung von einzelnen Buchstaben in Textdaten den Sinn bzw. die Aussage der Daten verändern kann (vgl. Fadaee, Bisazza, & Monz, 2017, S. 567; X. Zhang & LeCun, 2015, S. 3f.).

5.3.1 Methoden zur Substitution und Reihenfolgenanpassung

Resultierend aus dieser Problematik publizieren ZHANG & LECUN (2015) die Methodik, bestehende Textbausteine bzw. Wörter in Texten durch Synonyme mithilfe eines online

Thesaurus der entsprechenden Sprache zu substituieren. Dies erlaubt die Veränderung der Datenbasis ohne eine Verfälschung bestehender semantischer Relationen. Hierfür verwenden die Autoren ein verteilungsbasiertes Vorgehen zur Wahl der Anzahl der zu ersetzenen Wörter sowie des jeweiligen Synonyms (vgl. X. Zhang & LeCun, 2015, S. 3f.).

Ein weiterer Ansatz findet sich in der Variation von Satz- bzw. Paragraphenfolgen in längeren Dokumenten. Dieses Vorgehen bietet eventuelle Qualitätsverbesserungen in Paragraph Embeddings, die beispielsweise mit Doc2Vec trainiert werden können. Gleichwohl lässt sich dieses Verfahren auf Wortebene transferieren indem Wortfolgen in Sätzen randomisiert werden. Diese Worddrehungen sind allerdings als problematisch zu charakterisieren. Dies resultiert einerseits aus der Kontextsensitivität von Embeddings, da diese auf Basis der umliegenden Wörter trainiert werden sowie andererseits aus der Situation, dass die Wortabfolge fürhäufigkeits- bzw. verteilungsbasierte Modelle irrelevant ist und somit keine neuen Dokumente generiert werden können. Letztendlich ist demnach lediglich ein potentieller Vorteil hinsichtlich Paragraph-Embeddings zu vermerken.

Additional finden sich in der Literatur ebenfalls komplexere Methoden zur Augmentierung von textuellen Datensets unter der Anwendung von Machine Learning sowie Deep Learning. So spezifizieren WANG & YANG (2015) ein Vorgehen, welches, analog zu erstem Ansatz, auf das Substituieren von Wörtern im zugrundeliegenden Text fokussiert ist. Hierbei applizieren die Autoren allerdings Embedding-Modelle um Vektorrepräsentationen der enthaltenen Wörter zu erlangen. Anschließend ersetzen die Autoren die Wörter durch das jeweilig nächste Wort im Vektorraum (K-Nearest-Neighbor) und erreichen somit neue Sätze (vgl. W. Y. Wang & Yang, 2015, S. 2559f.). Hierbei ist allerdings eine starke Abhängigkeit zur Qualität des Embedding-Modells zu vermerken, da diese entscheidend für die Sinnhaftigkeit der Substitution und somit für die Qualität der Ergebnisse ist.

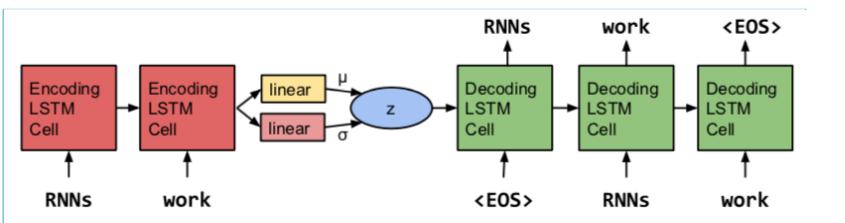
5.3.2 Generation von Textdaten

Neben Methoden zur Variation von bestehenden Daten, lassen sich in aktueller Literatur ebenfalls generative Methoden extrahieren. Diese versuchen auf Basis bekannter Daten neue Daten zu generieren. Hierbei ist beispielsweise auf Markov-Language-Models

bzw. N-Gramm-Modelle zu verweisen, mithilfe welcher Textdaten generiert werden können. In Referenz der dieser vorausgegangen Arbeit und der darin beschriebenen differenten Wahrscheinlichkeiten basieren Markov-Modelle im Kontext der Text-Generierung auf konditionalen Wahrscheinlichkeiten. Die Markov-Annahme beschreibt das Prinzip, dass die Wahrscheinlichkeit eines Wortes lediglich von wenigen vorangegangenen Wörtern abhängt und nicht durch alle vorangegangenen Wörter einer Sequenz bestimmt ist. So werden zur Vorhersage des nächsten Wortes nicht die verketteten, bedingen Wahrscheinlichkeiten aller Wörter verwendet, sondern lediglich die der vorherigen zwei Wörter (vgl. Jurafsky & Martin, 2008, S. Ch. 4, S. 3ff.).

Kommentiert [DH25]: Eventuell mit Formel untermauern und dafür Bild unten raus

In Hinsicht der Generierung von Text findet sich ebenfalls die Möglichkeit Deep Learning Methoden wie beispielsweise Recurrent Neural Networks oder Long-Short-Term-Memory-Netzwerke zu applizieren. Zur Erreichung dieses Ziels existiert eine große Bandbreite an Möglichkeiten, weshalb im aktuellen Kontext lediglich ein kurzer Einblick in diese gegeben wird. Exemplarisch implementieren SUTSKEVER, MARTENS & HINTEN (2011) eine Variante eines RNN, ein sog. *Multiplicative Recurrent Neural Network*, zur Generierung von Text auf Ebene einzelner Buchstaben. Ebenso publizieren PAWADE, SAKHAPARA, JAIN, JAIN & GADA (2018) ein Vorgehen zur Generierung von neuartigen Texten auf Basis bereits bekannter, unter Zuhilfenahme eines LSTMs auf Wortebene. Oppositionell zum vorherigen Beispiel versucht das Modell demnach das jeweils nächste Wort einer Sequenz auf Basis der bereits erstellten Wortsequenz zu generieren. BOWMAN ET AL. (2016) illustrieren weiterführend die Verwendung eines Encoder-Decoder-Modells zur Generierung von Text und fokussieren hierbei sog. *Variational-Autoencoder-Modelle*. Die nachfolgende Abbildung gibt eine strukturelle Übersicht einer solchen Architektur.



Kommentiert [m26]: Eigentlich will ich noch irgendwie rein bringen, dass Jorge et al die evaluiert haben und die gut scheinen

Kommentiert [DH27]: Tschau

Abbildung 21: Struktureller Aufbau eines Variational-Autoencoder-Modells nach BOWMAN ET AL. (2016)

Charakteristisch für derartige Modelle sind die Outputs des Encoder-Modells, welche einer spezifischen Verteilung folgen. Diese Anpassung erlaubt es unbekannte Vektoren zu erstellen bzw. Vektoren aus dieser Verteilung zu ziehen und jene an das dekodierende Modell zu übergeben um resultierend neue Texte zu generieren (vgl. Bowman et al., 2016, S. 11).

Weiterhin findet sich im Areal der Machine- bzw. Neural Machine Translation (NMT) (z.Dt.: Maschinelles Übersetzen) ein weiterer Ansatz zur Augmentierung von Textdaten. Resultierend aus der geringen Verfügbarkeit von qualitativen Datensätzen, welche Übersetzungspaare zweier Sprachen umfassen, findet das Prinzip der Back-Translation (z.Dt.: Rückübersetzung) Einzug in das Feld der NMT. Generell beschreibt dieses Vorgehen die Rückübersetzung von Sätzen, welche bereits in ihre Zielsprache übersetzt worden sind. Aufgrund der Vielfältigkeit potentieller Übersetzungsmöglichkeiten entstehen hieraus neue Sätze, welche eine additionale Datenbasis darstellen können. So beschreiben SENNINICH, HADDOW & BIRCH (2016) sowie DOMHAN & HIEBER (2017) mögliche Implementierungen dieser Back-Translation in Sequence-to-Sequence-Modelle um monolinguale Datensätze in multilinguale zu überführen und diese somit zusätzlich zum Training eines NMT-Modells zu verwenden (**QUELLE: Coloumbe?!**).

Kommentiert [m28]:

5.3.3 Applizierte Methoden der Data Augmentation

Im Rahmen der vorliegenden Arbeit werden drei differente Methoden zur Augmentierung der erlangten Textdaten angewandt. So wird unter anderem ein Vorgehen zur Substitution von Nomen und Verben durch Synonyme festgelegt. Zur Qualitätssicherung der Substitution werden Kriterien definiert, welche durch das Substitutionswort erfüllt werden müssen. **Tabelle XX** fasst diese Kriterien zusammen und gibt somit einen Überblick über die Methode.

Tabelle 3: Kriterien zur Augmentierung von Daten mittels Substitution

Kriterium	Beschreibung
<i>POS-Tag Übereinstimmung</i>	Das vorhandene sowie das Substitutionswort müssen über denselben Part-of-Speech-Tag verfügen.

<i>Ähnlichkeitsgrenzwert</i>	Beide Wörter müssen eine gewisse semantische Ähnlichkeit zueinander aufweisen.
<i>Wortdistanz</i>	Analog zum Ähnlichkeitswert müssen die Wörter eine gewisse Distanz auf Buchstabenebene aufweisen.
<i>Zufallswert</i>	Randomisierter Grenzwert der über die generelle Substitution eines Wortes entscheidet.

Generell wird sich in der Substitution auf Substantive und Verben fokussiert sowie Stopwords wie „is“, „the“ und „a“ ausgeschlossen. Zudem findet WordNet als Quelle für potentielle Substitutionskandidaten Anwendung.

Das erste Kriterium in obiger Tabelle stellt hierbei sicher, dass Substantive und Verben lediglich mit gleichartigen Wörtern ersetzt werden. Additional wird ein Grenzwert bzgl. der Ähnlichkeit beider Wörter spezifiziert, durch welchen sichergestellt werden kann, dass keine starke Variation der Semantik vorgenommen wird. Hierfür wird das Maß der Kosinus-Ähnlichkeit auf Basis, mittels der Architektur FastText trainierten, Embeddings appliziert. FastText bietet dabei, oppositionell zu anderen Embedding-Architekturen, das Plus, dass unbekannte Wörter, aufgrund der Fokussierung auf Buchstaben-N-Gramme, ebenfalls in Vektoren überführt werden können. Hierdurch kann sichergestellt werden, dass adäquate Vektoren für potentiell nicht trainierte Wörter abgeleitet und folglich korrekte Ähnlichkeitswerte ermittelt werden können.

Um keine Synonyme einzusetzen, welche lediglich eine sehr gering differierende Schreibweise aufweisen erfolgt weiterhin die Spezifizierung einer Mindestdistanz zwischen den beiden Wörtern. Diese wird in der aktuellen Untersuchung durch die Levenshtein-Distanz berechnet, da diese im Vergleich zur Hemming-Distanz unterschiedliche Wortlängen zulässt und gleichzeitig auf Buchstabenebene agiert. Die Distanz zweier Wörter s und w beschreibt hierbei die Mindestanzahl an Einfüge-, Ersetz- oder Löschoperationen, die zur Transformation von s in w notwendig sind (vgl. Vijaymeena & Kavitha, 2016, S. 20). Der Grenzwert wird mit dem Wert zwei belegt um vor allem leichte Abwandlungen der ursprünglichen Wörter, wie beispielsweise durch andere Endungen, zu vermeiden.

Aufgrund der deterministischen Eigenschaften dieses Vorgehens hinsichtlich gleicher Wörter in differenten Sätzen wird abschließend ein Schwellwert hinzugefügt, welcher durch eine randomisierte Zahl überschritten werden muss. Dies erfolgt um die Varianz

in den substituierten Sätzen zu erhöhen. **Tabelle XX** illustriert zwei Beispiele, welche anhand dieser Methodik erstellt worden sind.

Tabelle 4: Exemplarische Substituierung von Synonymen in Sätzen

Original	Substituiert
<i>The main purpose of clustering is to find the structure of unlabeled data</i>	<i>The main determination of clustering is to recover the structure of unlabeled datum</i>
<i>Clustering is the most commonly used analysis technique for customer segmentation</i>	<i>Clump is the most commonly used analysis proficiency for customer division</i>

Als eine weitere Methode zur Expansion der Datenbasis wird ein Markov-Modell zur Generierung von Texten auf Basis der bereits verfügbaren Daten erstellt. Anzumerken ist hierbei, dass jegliche Methoden zur Augmentierung auf den Basisdaten, die nach der oben beschriebenen Selektion vorhanden sind, ausgeführt werden. So werden die bereits mit Synonymen ersetzen Daten nicht zur Implementierung des Markov-Modells herangezogen, um eine Kumulation der Fehler aller Methoden zu vermeiden.

Zunächst werden zur Implementierung des Modells sämtliche bestehende Sätze durch Beginn- und Endsignaturen pro Satz versehen um anschließend **Trigramme** der bestehenden Sätze zu bilden.

$$s = \text{Clustering is an unsupervised method}$$

$$(< E >, < E >, \text{Clustering}), (< E >, \text{Clustering}, \text{is}), \dots, (\text{method}, < E >, < E >)$$

Darauffolgend werden die konditionalen Wahrscheinlichkeiten des jeweils letzten Wortes, unter der Bedingung, dass das vorausgehende **Bigramm** gegeben ist, berechnet.

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})} \quad (17)$$

Auf Basis dieser Wahrscheinlichkeiten lassen sich für gegebene Satzanfänge neue Sätze generieren, um somit die Datenbasis künstlich zu expandieren. In der aktuellen Arbeit werden hierfür die jeweils ersten beiden Wörter der bestehenden Dokumente herangezogen. Exemplarisch finden sich in folgender Tabelle vier solcher Sätze.

Tabelle 5: Beispiel für generierte Sätze mittels eines Markov-Modells

Gegebenes Bigramm Generierter Satz

„The degree“	„The degree of dissimilarity between groups“
„Clustering can“	„Clustering can be employed to identify subgroups of genes according to the ith group“
„This clustering“	„This clustering defines separations of the elements of different variances clearly suboptimal“
„The goal“	„The goal of cluster analysis is dendogram“

Wie an **Tabelle XX** zu erkennen garantieren die generierten Sequenzen keine syntaktische Korrektheit. Diese wird in der aktuellen Untersuchung allerdings vernachlässigt, da diese vor allem für vokabularbasierte Verfahren nicht von Relevanz ist und die alleinige Anwendung von Wahrscheinlichkeiten durch das Markov-Modell keine Garantie syntaktischer Korrektheit zulässt.

Abschließend erfolgt, analog zur Datenselektion, eine Evaluierung der augmentierten Daten anhand von Paragraph-Embeddings der generierten Sätze. Hierbei findet die Architektur eines *Deep Averaging Networks* Anwendung (vgl. **Kap. XX**). **Abbildung XX** illustriert die Verteilung der mittels *Principal Component Analysis* (PCA) reduzierten Vektoren in Form von Dichtediagrammen.

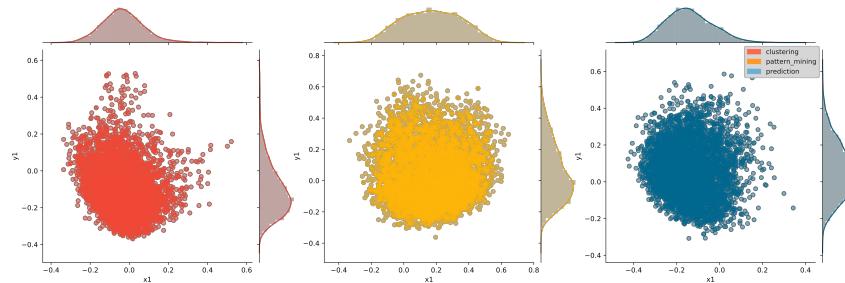


Abbildung 22: Dichtediagramm der augmentierten Daten des gefilterten Datensatzes (Eigene Darstellung)

Wie hieran zu erkennen liegt in der Klasse „pattern_mining“ eine höhere Streuung der Daten vor. Diese ist vermeintlich auf die zugrundeliegende Qualität der Daten zurückzu-

führen, da diese, aufgrund des Datenmangels, Informationen bzgl. der Sub-Areale „Sequential Pattern Mining“ und „Association Rule Mining“ beinhalten. Hieraus resultieren weitere Schritte zur Filterung dieser Daten, um generierte Daten schlechter Qualität zu entfernen und so den Streuradius zu reduzieren. Hierfür werden erneut die Methoden der regelbasierten Reinigung von Daten sowie des Noise Clustering angewendet.

5.4 Erstellung von Datensätzen

Das folgende Kapitel illustriert die finale Zusammenstellung der Trainings- und Validierungsdaten für die vorliegende Untersuchung. Hierfür wird zunächst auf die Trainingsdaten und anschließend auf die Validierungsdaten eingegangen.

5.4.1 Erstellung der Trainingsdatensätze

Die aus sämtlichen, oben aufgezeigten Schritten, resultierenden Daten werden final in zwei differente Datensets geteilt, die zum Training potentieller Methoden verwendet werden. Zum einen wird ein Datensatz erstellt, welches Kurzzusammenfassungen der gefundenen Publikationen umfasst. Dies folgt aus der Annahme, dass relevanter Kontext zu den Algorithmenklassen sowie Informationen hinsichtlich deren jeweiligen Anwendung gebündelt im Abstract zur Verfügung stehen.

Zum anderen wird ein Datensatz angelegt, der eine gefilterte und optimierte Version aller Daten darstellt. Dabei enthält dieser Datensatz definitionsähnliche Inhalte um eine möglichst hohe Datenqualität zu erzielen. **Abbildung XX** illustriert das schematische Vorgehen zur Erstellung dieses Datensatzes.



Abbildung 23: Prozess zur Generierung des gefilterten Datensatzes (Eigene Darstellung)

Hierbei wird die Menge der Sätze im zugrundeliegenden Datensatz nach spezifischen Schlüsselwörtern durchsucht. Diese Schlüsselwörter erstrecken sich exemplarisch über

die Satzteile „Clustering is“, „Clustering defines“ oder „Clusters are“ und sichern somit einen definitionsähnlichen Inhalt⁷. Weiterhin werden die extrahierten Sätze in Vektoren überführt um anschließend die Kosinusähnlichkeit zwischen den Sätzen und einem prädefinierten Topic zu errechnen (vgl. **Abbildung XX**).

Kommentiert [m29]: Sowas hier rein!

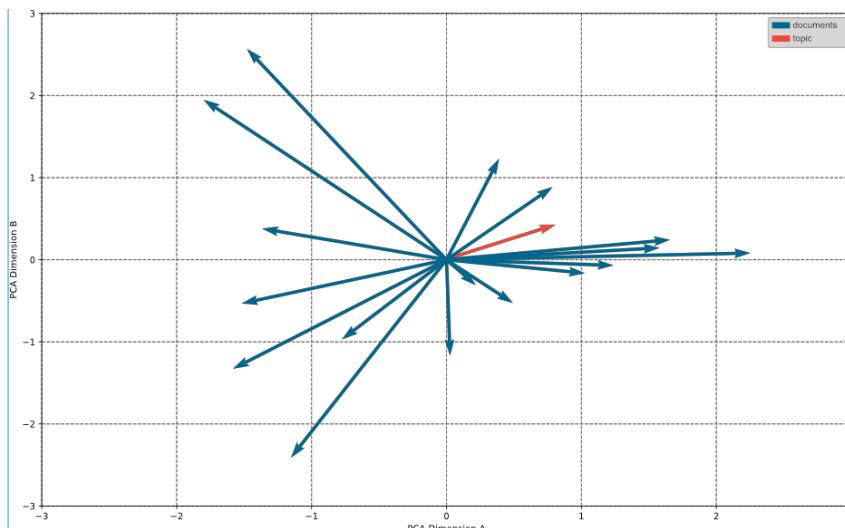


Abbildung 24: Illustration der Kosinusähnlichkeit zur Datenauswahl (Eigene Darstellung)

Dieses Topic resultiert aus der vorherigen Applikation von Keyword-Extraction-Methoden auf manuell zusammengetragene Definition des jeweiligen Algorithmus aus der Literatur. Abschließend dient das verwendete Ähnlichkeitsmaß zur Selektion der ähnlichsten Sätze. In Referenz zur obenstehenden Illustration beschreibt diese Ähnlichkeit die bzgl. ihres Winkels zueinander am nächsten liegenden Vektoren, wobei die Top 25 Prozent dieser in den Datensatz übertragen werden.

Final gibt **Tabelle XX** eine grobe Übersicht über die beiden erstellten Datensätze, sowie eine kurze Beschreibung dieser.

Tabelle 6: Übersicht über die erstellten Datensätze

⁷ Eine vollständige Liste der verwendeten Suchbegriffe findet sich im Jupyter Notebook „dataset overview.ipynb“ auf dem beigefügten Datenträger.

Datensatz	Beschreibung
<i>Abstract-Datensatz</i>	Dieser Datensatz enthält Abstracts aller gefundener Dokumenten nach initialer Selektion.
<i>Gefilterter Datensatz</i>	Dieser Datensatz enthält definitionsähnliche Dokumente, welche aus der Gesamtdatenbasis extrahiert worden sind.

Unter Anwendung der unter **Kapitel XX** deskribierten Methoden werden diese Datensätze additional in augmentierter Form erzeugt und abgelegt, sodass eventuelle Unterschiede zwischen augmentierten sowie den originalen Daten erkannt werden können, um somit auf die Wirksamkeit der Augmentierung zu schließen.

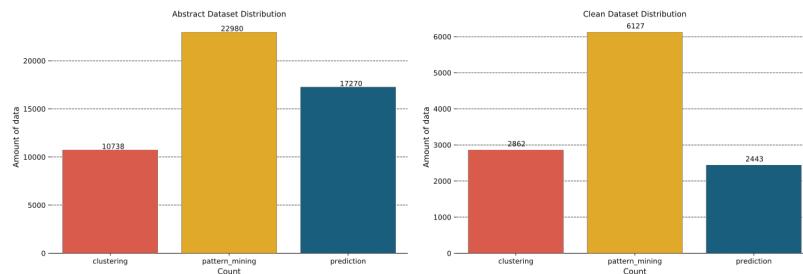


Abbildung 25: Klassenverteilung der finalen Datensätze (Eigene Darstellung)

Obenstehende Abbildung gibt zudem einen Überblick über die Klassenverteilung in den finalen Datensätzen. Hier zu erkennen ist, dass eine starke Ungleichverteilung der Klassen vorliegt, weshalb der Einfluss dieses Bias auf potentielle Algorithmen ebenfalls weiter untersucht werden muss.

5.4.2 Erstellung der Validationsdatensätze

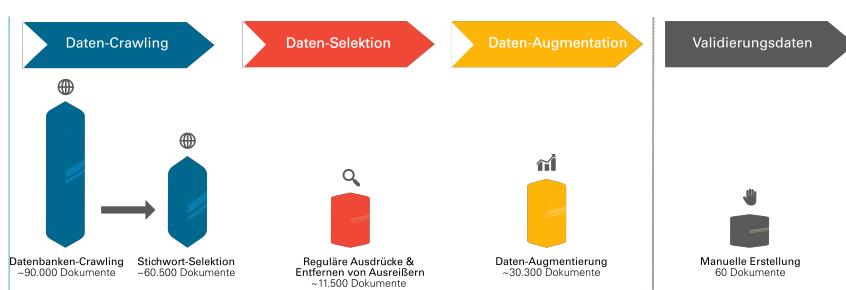
Die in **Kapitel XYZ** zugrundeliegende syntaktische Divergenz der Daten, begründet die separate Erstellung eines Validierungsdatensatzes, neben der klassischen Aufteilung in Trainings- und Testdaten. Zwar beinhalten die gefilterten Daten definitionsähnliche Kon-

Kommentiert [m30]: Widerspricht sich das jetzt mit den Aussagen bei Ergebnisse Eval Classifier?

Kommentiert [DH31R30]: Vorgehen in Eval beschrieben – Oder einfach raus und sagen dass die geschnitten werden

strukte und bündeln die Semantik der zugrundeliegenden Klassen durch ihr abgegrenztes Vokabular sowie ihre Terminologie, allerdings tun dies klassische Problemstellungen nicht (vgl. **Kapitel XYZ**).

Für die Erstellung der Validationsdatensätze gilt im Rahmen des aktuellen Forschungsvorhabens, die Anforderung, dass alle manuell erstellt werden und nicht maschinell manipuliert werden. Hinzukommt, dass diese Daten ausschließlich für das finale Testen der Generalisierungskraft von Methoden sowie des resultierenden Recommender-Systems verwendet werden und nicht für das Training von Modellen. Als Resultat finden sich insgesamt 60 verschiedene Problemstellungen im elektronischen Anhang der Arbeit wieder, mit welchen das System evaluiert wird. Hierbei sind die Validierungsdaten auf den zugrundeliegenden Klassen gleichverteilt, um mögliche Abweichungen der Modellvorhersage zu vermeiden. Die nachfolgende Abbildung illustriert die für die Arbeit fundamentalen Datentöpfe, sortiert nach ihrer Einsatzmöglichkeit im beschriebenen Vorhaben.



Kommentiert [DH32]: Überarbeiten und Daten-Crawling anpassen. Prozess mit Filterung darstellen

Darstellen das Eval Daten separiert betrachtet werden

Oben Prozessbausteine wie EPK? Und Rahmen weg?

Abbildung 26: Übersicht der Datenmenge in verschiedenen Verarbeitungsschritten (Eigene Darstellung)

Wie zu erkennen ist die ursprüngliche Datenbasis auf rund 30.300 semantisch relevante Daten reduziert worden. Dabei wird für das Modelltraining der augmentierte Datensatz (3) sowie der selektierte Datensatz (2) verwendet, abhängig von den resultierenden Modellgenauigkeiten. Der Validierungsdatensatz ist separat von allen anderen Datensätzen zu betrachten und wird im weiteren Verlauf der Arbeit nicht weiterbearbeitet.

6 Konzeption und Implementierung des Prototyps

In Referenz zur zweiten forschungsleitenden Frage ordnet sich das folgende Kapitel in der Phase **Design und Development** im eingangs illustrierten Design-Science-Prozess ein und fokussiert demnach die Entwicklung des Artefakts. Hierbei wird sich wie in **Kapitel XX** dargelegt, an einer Variation des Vorgehensmodells zur Softwareentwicklung nach PRESSMAN (2015) orientiert. Additional findet in der Phase der Anforderungsdefinition eine Unterteilung in funktionale- und nicht-funktionale Anforderungen sowie eine Priorisierung dieser nach IEEE STD 830-1998 (1998) statt. Im Sinne einer Strukturierung des Kapitels wird zunächst auf die Anforderungsdefinition eingegangen. Anschließend wird die Entwicklung des Frontend sowie des Backend diskutiert um abschließend die finale Programmstruktur zu illustrieren und das entwickelte System den Anforderungen gegenüberzustellen.

6.1 Anforderungsdefinition

In Anbetracht des in **Kapitel XYZ** beschriebenen Forschungsvorhabens, dient das folgende Kapitel der Charakterisierung und Definition von Anforderungen, welche an den zu entwickelnden Prototyp gestellt werden. Hierbei ist zu unterstreichen, dass sich lediglich auf den zu entwickelnden Prototypen bezogen wird und nicht auf die zu implementierenden Methoden dessen.

Grundlegend wird sich dabei auf das in **Kapitel XYZ** definierte Forschungsvorhaben bezogen und daraus die Anforderungen für den zu entwickelnden Prototyp abgeleitet. Hervorzuheben ist hierbei, dass es sich um eine initiale Anforderungsaufnahme handelt, welche in weiterführenden Forschungsvorhaben durchaus erweitert werden kann.

Des Weiteren erfolgt eine Kategorisierung der Anforderungen in funktionale und nicht-funktionale Anforderungen. Erstere beschreiben im Allgemeinen alle Aktionen, die der Prototyp gewähren muss, um die Funktionalität dessen zu gewährleisten. Die nicht-Funktionalen Anforderungen hingegen, repräsentieren Eigenschaften und Systemattribute des Systems, welche keine spezifischen Funktionalitäten betreffen (vgl. S. Robertson & Robertson, 2012, S. 9f.). Hinzukommt, dass die Anforderungen nach IEEE Std 830-1998 (1998) in drei Klassen priorisiert sind, um den Entwicklungsprozess zu vereinfachen. Dabei stellen essentielle Anforderungen die Kernfunktionalität der Applikation dar, ohne welche sie nicht ordnungsgemäß zu applizieren wäre. Die konditionalen Anforderungen erweitern das System in seiner Funktionsweise, allerdings würde durch das Fehlen einer konditionalen Anforderung das System weiterhin seine Kernfunktionalität erfüllen. Wohingegen optionale Anforderungen lediglich eine Erweiterung für den Nutzer darstellen, jedoch keinerlei Einfluss auf die eigentliche Funktionalität der Applikation haben.

Kommentiert [DH33]: Quelle Richard

Die nachfolgende Tabelle illustriert die in der Anforderungsanalyse gefundenen Anforderungen sowie ihre Kategorisierungen und Priorisierungen.

Tabelle 7: Prototypbezogene Anforderungsdefinition

A_ID	Kategorie	Priorisierung	Anforderung
A1	Funktional	Essentiell	Zuordnung einer Problemstellung zu einer Verfahrensklasse
A2	Funktional	Essentiell	Ausgabe empfohlener Datenstruktur für Verfahrensklasse
A3	Funktional	Essentiell	Ausgabe einer personalisierten, exemplarischen Analyse
A4	Funktional	Konditional	Bessere Zuordnung als reines Raten (Cohens Kappa > 0.5)
A5	Funktional	Konditional	Extraktion potentieller Analyseentitäten aus Problemstellung
A6	Funktional	Konditional	Deterministische Zuweisung zu einer Verfahrensklasse
A7	Funktional	Optional	Eingabe einer kurzen sowie einer langen Problembeschreibung
A8	Funktional	Optional	Entgegenwirken von syntaktischer Divergenz
A9	Funktional	Optional	Ausgabe von Wahrscheinlichkeiten der Zuordnung

A10	Nicht-Funktional	Essentiell	Einfachheit
A11	Nicht-Funktional	Essentiell	Erweiterbare Systemarchitektur
A12	Nicht-Funktional	Konditional	Usability des Systems
A13	Nicht-Funktional	Konditional	Plattformunabhängigkeit

Wie aus dem Forschungsvorhaben hervorgeht, ist die Kernaufgabe der Applikation, Problemstellungen des Machine-Learning bzw. der Data Science einer Verfahrensklasse zuzuordnen (A1), weshalb diese Anforderung als essentiell einzustufen ist. Neben der Zuordnung wird anschließend eine personalisierte, exemplarische Analyse als Ausgabe an den Nutzer geliefert (A3). Dabei ist es das Ziel, gleichzeitig eine Empfehlung bezüglich der zu verwendeten Datenstruktur an den Nutzer zu übergeben (A2). Dies dient unter anderem dazu, dem Data Scientist einen ersten Einblick, in die für die Problemstellung zugrundeliegende Datenbasis zu vermitteln und somit die Kommunikation beider Interessengruppen zu simplifizieren. Die eben charakterisierten Anforderungen bilden hierbei die Kernfunktionalität der Anwendung. Das Fehlen konditionaler oder optionaler Funktionalitäten führt zwar zu einem qualitätsmindernden Ergebnis, jedoch würde die Applikation das Ziel des aktuellen Forschungsvorhabens grundlegend erfüllen.

Kommentiert [m34]: Hatte ich anders geschrieben oben
– Sollten wir also angleichen 😊

Konditional besteht die Anforderung, dass multiple Entitäten aus der Eingabeproblembeschreibung extrahiert werden (A5), um eine detaillierte Explikation der Datenstruktur zu generieren. In der Regel ist es die Aufgabe der NER, Entitäten wie Personen, Organisationen, etc. aus dem Text zu extrahieren (vgl. Yadav & Bethard, 2018, S. 2145). Im Rahmen des aktuellen Forschungsvorhabens sollen diese jedoch durch potentielle Datenattribute der Problembeschreibung definiert sein. Der zu entwickelnde Prototyp soll somit in der Lage sein, mögliche nominale oder kategorische Variablen aus der Problembeschreibung zu extrahieren und diese in der personalisierten Analyse zur Verfügung stellen.

Hinzukommt, dass der Anwender beim Fehlen einer ausführlichen Problembeschreibung und zur besseren semantischen Abgrenzung die Möglichkeit haben soll eine Kurzbeschreibung einzugeben (A7). Dies resultiert aus der Annahme, dass eine zusammengefasste Version der Problemstellung ebenfalls deren Semantik aggregiert. Durch die Zusammenfügung der beiden Beschreibungen kann somit der Kern der Aussage vermehrt in das Dokument übernommen werden, um somit die Zuweisung einer Klasse zu erleichtern.

Des Weiteren soll das System in seiner Zuordnung besser funktionieren als pures Raten (A4). Dazu wird im nachfolgenden **Kapitel XYZ** das entsprechende Evaluationsvorgehen sowie die -Metrik deskribiert.

Damit die Ergebnisse der Zuordnung nachvollziehbar für den Nutzer dargestellt werden, empfiehlt sich die Ausgabe der Zuordnungswahrscheinlichkeit zu einer Verfahrensklasse (A8). Hinzukommt, dass die in **Kapitel XYZ** angesprochene Problematik der syntaktischen Divergenz die Zuordnung einer Problembeschreibung zu einer Verfahrensklasse erschwert. Basierend auf diversen Methodiken der distributionellen Semantik, ermöglicht das System optional, syntaktische Divergenzen zu umgehen (A9).

Eine einfache Bedienbarkeit, Systemerweiterbarkeit und Plattformunabhängigkeit bilden die nicht-funktionalen Anforderungen des Prototyps (A10-13). Die Systemerweiterbarkeit soll durch die Anwendung objektorientierter Programmierung, in der Implementation der Software gewährleistet werden. Hinzukommt, dass webbasierte Technologien sowie eine Kompilierung der Software in einem Docker-Container die Plattformunabhängigkeit gewährleisten.

Kommentiert [DH35]: Löschen wenn nötig.

6.2 Frontend Konzeptionierung

Im Rahmen der Frontend-Konzeptionierung wird die Entwicklung der Nutzeroberfläche fokussiert, was sich ebenfalls in der Phase des Quick Design nach Pressman (2015) wiederfindet. Hierbei wird sich der Methode der Wireframes bedient, welche zur groben Modellierung dient und somit zur Transformation von Ideen in Visualisierungen herangezogen wird. Generell beschreibt ein Wireframe ein Modell, welches die Eckpunkte und relevanten Charakteristiken eines Objektes darstellt (vgl. Bagali & Waggenspack, 1995, S. 339).

The wireframe illustrates the initial user interface design. At the top, there are two tabs: "Tabs" and "Main". The "Main" tab is active, revealing a form area. This form includes an "Input" section containing three text input fields: "Title", "Short description", and "Long description". Below these fields is a "Submit" button. At the bottom of the main content area is a "Footer" section.

Abbildung 27: Wireframe der initialen Nutzeroberfläche (Aufgenommen auf Wireframes.cc)

In obenstehender Abbildung ist diesbezüglich ein konzeptioneller Entwurf der Hauptseite des Prototyps dargestellt. Wie hieran zu erkennen wird in diesem Konzept, in Referenz zu den Anforderungen A7 und A10, ein einfaches Design gewählt, welchen schnell zu Überblicken ist. Mittig auf der Startseite findet sich demnach direkt die Möglichkeit wieder, eine Problembeschreibung einzugeben und diese mit einem Klick durch das System evaluieren zu lassen. Hierbei unterteilt sich die Eingabe in die drei Eingabefelder *Titel*, *Kurzbeschreibung* sowie *längere Beschreibung*, wobei lediglich ersteres und letzteres als verpflichtende Eingaben markiert sind.

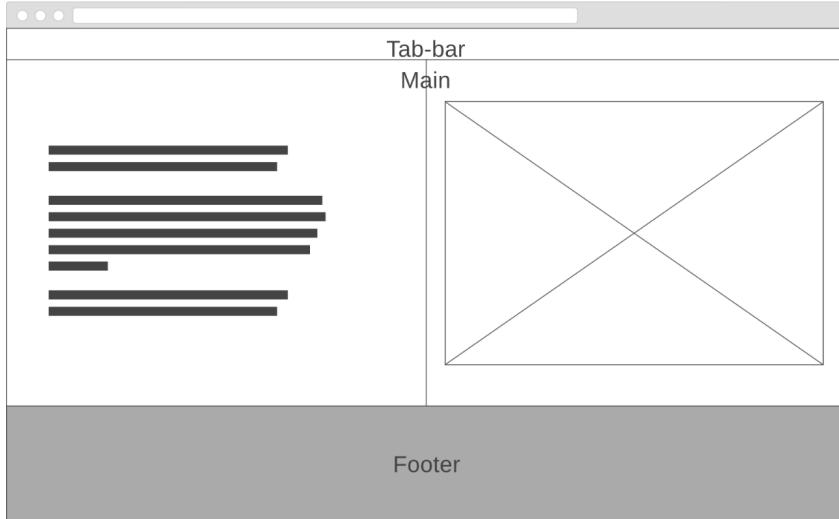


Abbildung 28: Wireframe der Ergebnisseite des Prototyps (Aufgenommen auf Wireframes.cc)

Weiterhin wird dieses Design ebenfalls auf der Ergebnisseite des Prototyps fokussiert. Obenstehende Abbildung illustriert diese Seite, welche zur Visualisierung der finalen Empfehlung dient. Hierbei wird auf der linken Seite erneut die eingegebene Problemstellung sowie die generelle Zuweisung dieser zu einer Verfahrensklasse ausgegeben. Die andere Seite hingegen fokussiert die Ausgabe einer exemplarischen Analyse in Form eines Diagramms sowie der benötigten Datenstruktur.

6.3 Entwicklung des Backend

Im Rahmen der Entwicklung der Funktionalitäten sowie der Zusammenhänge des Systems, werden drei differente Ebenen des Prototyps identifiziert, die sowohl eine ablauforientierte wie auch eine strukturelle Sicht auf das System ermöglichen (vgl. **Abbildung XX**).

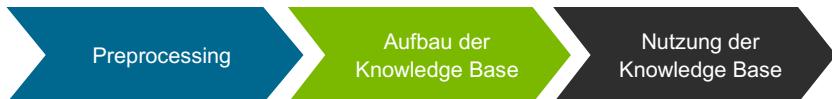


Abbildung 29: Illustration der identifizierten Entwicklungsebenen (Eigene Darstellung)

Wie in obiger Abbildung zu erkennen wird hierfür zunächst auf die Vorverarbeitung der Daten eingegangen, um alle hierfür benötigten Operationen zu implementieren. Anschließend erfolgt der Aufbau der Wissensbasis (engl.: Knowledge Base) (KB) sowie die Integration der hierfür nötigen Methoden in den Programmkontext. Diese KB stellt hierbei die Sammlung aller Informationen und Konstrukte dar, die zur Berechnung und Ausgabe einer Empfehlung verwendet werden. Final wird die Ebene zur Nutzung der erstellten Basis fokussiert und implementiert. An dieser Stelle sei angemerkt, dass dieser Prozess hinsichtlich der Durchführung nicht als streng sequentiell zu erachten ist, da nicht alle Methoden einzelner Phasen eine generelle Abhängigkeit zur vorherigen Phase aufweisen. Beispielsweise werden Methoden zum Aufbau der Knowledge Base appliziert, die keine Notwendigkeit zur vorhergehenden Aufbereitung der Daten aufweisen. Im Sinne der Übersichtlichkeit stellt dieser Prozess ebenfalls die Struktur des folgenden Kapitels dar, wobei der Fokus auf der Illustration der Vorgehensweise sowie der Implementierung liegt. Das Erstellen bzw. das Training der differenten Modelle sowie die resultierende Auswahl der jeweiligen Algorithmen wird in **Kapitel XX** aufgezeigt und beschrieben.

6.3.1 Vorbereitung und Vorverarbeitung der Daten

Im Rahmen der ersten Ebene des Systems besteht das Ziel in der Vorverarbeitung und Säuberung der Ausgangsdaten um darauf aufbauend Methoden zur Extraktion von semantisch relevanten Informationen zu applizieren und damit die Knowledge Base kreieren. Hierbei ist anzumerken, dass die differenten Algorithmen in der darauffolgenden strukturellen Ebene auf unterschiedlich prozessierte Ausgangsdaten zurückgreifen, weshalb im Rahmen der Implementierung ein Spektrum an möglichen Vorverarbeitungsmethoden des NLP berücksichtigt wird. **Tabelle XX** zeigt hierfür die Zuordnung von Vorverarbeitungsmethoden zu Methoden der zweiten Ebene der oben dargestellten Struktur.

Tabelle 8: Zuordnung von Vorverarbeitungsmethoden zu Methoden des KB-Aufbaus

Preprocessing	Keyword Extraction	Topic Modeling	Vector Space Modeling
<i>Stemming / Lemmatization</i>	X	X	
<i>Entfernen von Stopwords</i>	X	X	
<i>POS-Tagging</i>	X		
<i>Entfernen von Punktierungen</i>	X	X	X
<i>Entfernen von Zahlen</i>	X	X	X

Wie hieraus zu erkennen, werden alle, die in der vorherigen Forschungsarbeit deskribierten, Methoden verwendet um die Ausgangsdaten zu prozessieren. Hierbei resultiert die Zuordnung der Methodengruppen auf Basis der Natur bzw. der Funktionalität dieser. So zeigen CAMACHO-COLLADOS & PILHEVAR (2018), dass eine starke Vorverarbeitung und Bereinigung von Textdaten keine signifikanten Qualitätsverbesserungen im Areal von Embeddings hervorbringen, weshalb hier ebenfalls auf die Anwendung multipler Methoden verzichtet wird. Zur Vermeidung der Abbildung ähnlicher Wörter im Vektorraum, welche sich lediglich durch das Hinzufügen von Zahlen oder Punktierungen unterscheiden (z.B. „clustering“, „clustering,“ und „-clustering“), werden diese Zeichen allerdings im Rahmen des Preprocessing entfernt.

Weiterführend sind alle der oben deskribierten Methoden im Bereich der Keyword Extraction anwendbar, da durch eine Filterung der Ausgangsdaten klarere Topics als Resultat der Anwendung derartiger Algorithmen zu erwarten sind. Hierbei ist speziell zu erwähnen, dass die Methode der Lemmatization gegenüber der des Stemming bevorzugt wird. Dies resultiert daraus, dass mittels Stemming Wörter durch Abschneiden von Endungen reduziert werden und hierdurch keine Garantie auf korrekte Wörter gegeben werden kann. Exemplarisch ist hier die Reduktion von „organize“ zu „organ“ anzuführen, welche den Wortsinn verändert oder die Änderung von „reply“ zu „rep“, wodurch kein existentes Wort mehr repräsentiert wird. Dies führt bei häufigkeitsbasierten Verfahren, zu welchen die meisten Keyword-Extraction-Algorithmen gezählt werden können, zur Problematik inkorrektener Häufigkeiten von Wörtern sowie zur Erweiterung des Vokabulars mit solchen. Ähnlich charakterisiert ist die Vorverarbeitung in Hinsicht auf

Topic-Modeling-Modelle, speziell auf den Algorithmus LDA, da hier ebenfalls Operationen auf Häufigkeiten in Form von Wahrscheinlichkeitsverteilungen durchgeführt werden.

Gegeben diesen Sachverhalt, ist im Rahmen der Implementierung zu konstatieren, dass zur verschiedenseitigen Verarbeitung von Daten eine einheitliche Datenbasis erforderlich ist, weshalb im Rahmen der Implementierung zunächst eine Korpus-Klasse definiert wird, die zur Sammlung der Daten aus verschiedenen Quelldateien sowie zur einheitlichen Bereitstellung dieser dient.

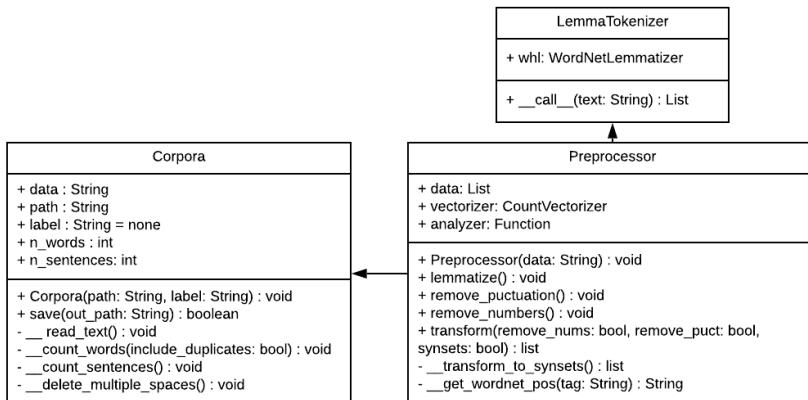


Abbildung 30: Klassenübersicht zur Datenvorverarbeitung (Eigene Darstellung)

Wie in **Abbildung XX** zu erkennen bildet die *Corpora*-Klasse die Basis, auf welcher die weiterhin implementierte *Preprocessor*-Klasse operieren kann. Hierbei inkludiert diese Klasse die Methode *transform*, welche durch Parametereinstellungen zu unterschiedlichen Arten der Vorverarbeitung angewendet werden kann. Die additionale Klasse *LemmaTokenizer* dient indes zur Lemmatisierung der Ausgangsdaten im Rahmen des Preprocessors.

6.3.2 Aufbau der Knowledge Base

Die Grundlage für die Nutzung des deskribierten Recommender-Systems bildet die Knowledge-Base, welche die unstrukturierten Textdaten des Empfehlungssystems in

strukturierter Form speichert (vgl. **Kapitel XYZ**). Dabei soll diese die benötigten Fakten repräsentieren, welche nötig sind, um geeignete Empfehlungen für den Nutzer zu prognostizieren. Die Herausforderung besteht hierbei in der Entwicklung eines konzeptionellen Entwurfs, welcher die Applikation aller aufgezeigten Methoden unterstützt. Der grundlegende Aufbau des Systems ist in der folgenden Abbildung, in Form eines Entity-Relationship-Diagramm dargestellt, welches die einzelnen Entitäten, Relationen, Attribute und Schlüssel eines konzeptionellen Datenschemas veranschaulicht (vgl. Unterstein, 2013, S. TBD).

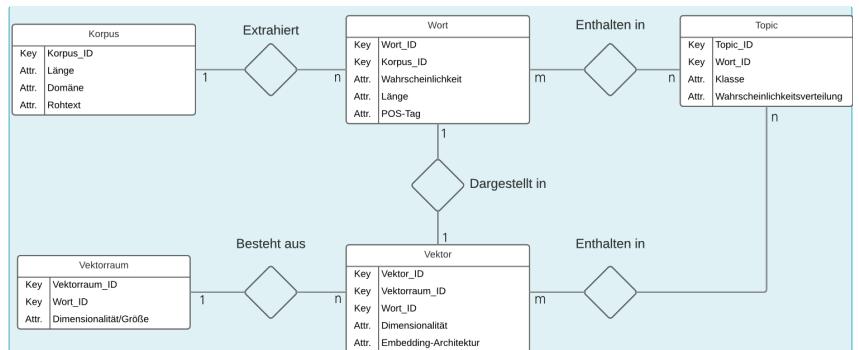


Abbildung 31: Konzeptioneller Entwurf der Knowledge Base

Hervorzuheben ist, dass es sich hier um einen ersten Entwurf der Knowledge Base handelt und nicht um ein konkretes Schema zur Implementierung dieser. Dabei sollen nur die Entitäten modelliert werden, welche essentiell für die in **Kapitel XYZ** aufgezeigten Methoden sind.

Der Korpus ist die Basis für alle weitere Entitäten des Systems. Die vorverarbeiteten Daten bilden den Kern des Artefakts, welcher in Form von Tokens, Lemmas und POS-Tags vorliegt. Zusätzlich wird die Länge des Textes als nützliche Metainformation, für sich anschließende Analysen erfasst. Auf Basis des Korpus werden die einzelnen Wörter in Form ihres vorverarbeiteten Tokens in der Wort-Entität abgespeichert. Im Vergleich zur Korpus-Entität, stehen diese aber nicht mehr in Bezug zueinander und liefern somit keinen Kontext für Modelle der distribuierten Semantik. Hinzukommt, dass im Zusammenspiel mit der Fremdschlüsselbeziehung, die Auftrittswahrscheinlichkeit der Wörter mit festgehalten wird, da die Berechnung dieser, bei großen Textmengen mit

Kommentiert [DH36]: Welche Notation?

Es gibt:
Chen
Min-Max (Iso)
UML
Krähfuß

Die jetzige ist ein Mix aus Chen und ??

SeitenzahlL!

**Kommentiert [DH37]: Extrahier -> beinhaltet/inkludiert
Vektor -> inkludiert semantik von -> Wort
Topic -> repräsentiert durch -> Vektor
Vector Space -> represents -> Korpus**

hohen Rechenaufwänden verbunden sein können und aus diesem Grund bereits in der Knowledge Base vorgehalten werden. Durch die Fremdschlüsselbeziehung erklärt sich ebenfalls das Domänenattribut in der Korpus-Entität. Zum einem kann es als Label verstanden werden, um den Korpus einer Klasse zuzuweisen. Zum anderen unterstützt es die Unterscheidung von Polysemen, da Wörter auf ihre spezifische Domäne zurückgeführt werden können. Hierbei wird den Polysemen ein klarer Semantischer Bezug gegeben und diese einem klaren Konzept der domänenpezifischen Terminologie zugeordnet. Die Diskriminierung in Domänen hilft der in **Kapitel XYZ** linguistischen Problematik entgegenzuwirken.

Basierend auf der Wort-Entität werden die Wörter mithilfe eines Modells der distributiven Semantik in ihre entsprechende Vektorform überführt, die in der vorliegenden Arbeit als Grundlage für das Erlernen eines Klassifikators dienen. Die Gesamtheit aller Vektoren einer Embedding-Architektur bilden den Vektorraum, auf welchen beispielsweise Analogien oder Ähnlichkeiten der distribuierten Repräsentationen abgeleitet werden können. Hinzukommt, dass mögliche Diskrepanzen der Modellgenauigkeit sich auf die entsprechende Embedding-Architektur zurückführen lassen.

Die Topic-Entität repräsentiert die einzelnen Resultate des Topic-Modelling sowie Keyword-Extraction-Modelle. Aufgrund dessen, dass die Modelle des Topic-Modelling anhand einer Wahrscheinlichkeitsverteilung der Wörter bestimmen, wie die Topics sich zusammensetzen, wird diese anhand eines Attributes mit abgebildet. Im Falle von Keyword-Extraction-Algorithmen müsste es sich hierbei, um eine Gleichverteilung handeln. Die relevantesten Wörter, werden im Falle der Keyword-Extraction-Algorithmen, anhand des im Wort enthaltenen Score-Attributs bestimmt. Additional können einzelne Topics ebenfalls in Vektorrepräsentationen überführt werden, um die semantischen Kernkonzepte dieser zu erfassen und miteinander zu vergleichen.

Um die Methoden des aktuellen Forschungsseminars dem konzeptionellen Entwurf zuzuordnen wird sich nachfolgender Tabelle bedient.

	Korpus	Wort	Vektor	Vektorraum	Topic
<i>Keyword-Extraktion</i>	X	X	-	X	X
<i>Topic-Modelling</i>	X	X	-	X	X
<i>Klassifikation mithilfe von Embeddings</i>	X	-	X	X	(X)

Die im Rahmen der Arbeit vorgestellten Keyword-Extraktionsmethoden basieren auf graphbasierten (TopicRank, etc) oder rein statistischen Algorithmen. Im Vergleich zu rein Statistik basierenden Keyword-Extraktoren benötigen graphbasierte Verfahren den ganzen zugrundeliegenden Korpus, um die Schlüsselwörter eines Textes zu extrahieren. Wohingegen Verfahren wie TF-IDF oder TF-IDGM rein auf den Wahrscheinlichkeiten basieren, welche sich aus der Wort-Entität ableiten lassen ([Quelle](#)). Hinzukommt, dass die Keyword-Extraction-Methoden dafür verwendet werden, um einzelne Topics bezüglich der differenten Verfahrensklassen zu erstellen und diese optional in distribuierte Vektoren überführt werden, um die semantischen Nuancen eines Topics hervorzuheben, sowie diese im Vektorraum zu vergleichen.

Kommentiert [DH38]: Quellen

Das Topic-Modelling beschränkt sich im aktuellen Forschungsvorhaben auf das deskribierte LDA-Verfahren. Dabei berechnet das Verfahren, basierend auf konditionalen Wahrscheinlichkeiten, die Wortzugehörigkeit zu einem Topic sowie zu einem Dokument, was die Einordnung in die Wort- sowie Korpus-Entität begründet. Ähnlich wie bei den Keyword-Extraction-Verfahren können auch hier Topics in distribuierte Repräsentationen überführt werden, um anschließend diese im Vektorraum zu vergleichen.

Die Klassifikation von Embeddings vereinigt zum einen die Anwendung klassischer Machine-Learning-Klassifikatoren, wie beispielsweise SVM oder K-Nearest-Neighboor-Klassifikationen. Zum anderen aber auch Deep-Learning-Ansätze wie die in **Kapitel XYZ** deskribierten LSTM- oder GRU-Architekturen. Hinzukommt, dass deren Eingabe durch Embedding-Modelle determiniert wird und diese initial durch den Korpus angelernt werden müssen. Dabei unterstützt der oben illustrierte konzeptionelle Entwurf, dass Modelle der distribuierten Semantik auf mehr als einem Korpus trainiert werden können, um bei-

spielsweise die Generalisierungsfähigkeit dieser zu verbessern. Basierend auf den resultierenden Embeddings, ergibt sich der Vektorraum auf welchen die differenten Klassifikatoren applizieren. Hinzukommt, dass im Falle einer hohen zugrundeliegenden Klassenanzahl, optional die Topic-Verteilungen für das modellieren eines Klassifikators mit herangezogen werden kann.

6.3.3 Nutzung der Knowledge Base

Die Nutzung der KB charakterisiert sich durch die Ausgabe der finalen Empfehlung zum Nutzer. Wie eingangs bereits deskribiert umfasst diese die Einordnung der übergebenen Problemstellung durch das System, eine Illustration der benötigten Datenstruktur sowie eine exemplarische Analyse, welche personalisiert mittels Entitäten der Problemstellung erstellt wird. Zunächst wird zur Deskription des Vorgehens auf den Kern der Empfehlung in Form der Zuordnung der Problemstellung eingegangen.

Einordnung der Problemstellung zu einer Verfahrensklasse

Insgesamt werden zur Einordnung von potentiellen Problemstellungen durch den Prototyp drei differente Vorgehen angewandt, welche final im Rahmen eines Verfahrens des *Ensemble Learnings* zusammengeführt werden. **Abbildung XX** gibt hierfür eine strukturelle Übersicht über die verwendeten Methoden.

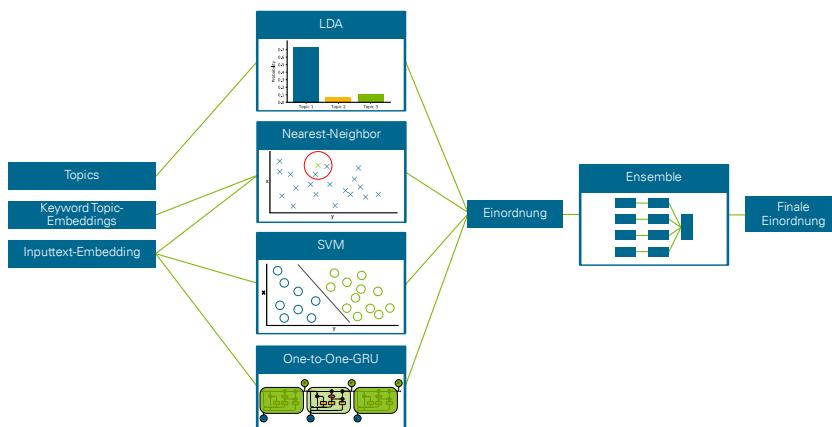


Abbildung 32: Illustration des Vorgehens in der Klassifikation (Eigene Darstellung)

Zunächst ist, in Referenz zu vorherigem Kapitel, auf das Modell LDA zu verweisen, welches indirekt, durch die Erkennung der Themenverteilung in unbekannten Dokumenten, ebenfalls zur Klassifikation neuer Problemstellungen herangezogen werden kann. Hierbei wird zur Zuweisung einer Verfahrensklasse das am stärksten im Dokument vertretene Topic herangezogen. Diese Art der Anwendung basiert auf der Annahme, dass das Modell die Semantik der Zielklassen in Form von Topics inkludiert und weiterhin deren Anzahl mit der Anzahl an Zielklassen übereinstimmt. Im Falle eines Modells mit mehr Topics als zuzuordnenden Data-Mining-Verfahrensklassen besteht additional die Option, einen weiteren Klassifikationsalgorithmus zu applizieren und diesen auf Basis der eruierten Topicverteilungen zu trainieren. Aufgrund der schlechten Modellgüte dieser Kombination wird diese hier allerdings vernachlässigt (vgl. **Kapitel XX**).

Weiterhin werden im Rahmen der Nutzung der Knowledge Base ebenfalls Klassifikationsalgorithmen aus dem Bereich des Maschine Learning verwendet um Vektorrepräsentierte Problemstellungen einer Verfahrensklasse zuzuordnen. Hierbei zeigen BISHOP (2006) und GOODFELLOW ET AL. (2016) multiple potentielle Algorithmen. In der aktuellen Untersuchung wird sich aufgrund dieser Vielfalt einerseits auf die Anwendung einfacherer Methoden mit K-Nearest-Neighbor-Klassifikationen und Support Vector Machines (SVM) sowie auf Methoden des Deep Learning mit RNN-basierten Modellen beschränkt.

Ersterer Algorithmus findet hier zur Eruierung der Klasse des nächsten Vektors im Vektorraum Anwendung, wobei der Input durch ein Paragraph-Vektor des Inputtextes charakterisiert ist. Weiterhin repräsentieren die diesem Vektor gegenübergestellten Vektoren, distribuierte Repräsentationen der mittels Keyword-Extraction-Algorithmen errechneten Topics⁸. Die Überführung der eingegebenen Problemstellung in einen Vektor erfolgt hier analog zum Vorgehen bei der Berechnung der Topic-Embeddings in der Phase des Aufbaus der KB. So werden die einzelnen Wörter durch ein Fast-Text-Modell in Wortvektoren überführt und diese mithilfe eines DAN-Modells in Paragraph-Embeddings transferiert.

⁸ Diese werden im weiteren Verlauf als Topic-Embeddings bezeichnet.

Zeitgleich bilden diese Vektoren den Input zur Einordnung einer Problemstellung im Rahmen klassischer Textklassifikationsaufgaben. Im Kontext dieses Areals wird sich der Methoden der SVM sowie der Gated-Recurrent-Units bedient.

Kommentiert [m39]: Check if One to One or Many to One

> Classification based on Embeddings

> SVM

> One to One LSTM/GRU

> Many to One LSTM/GRU

Kommentiert [m40]:

Wie aus diesen Beschreibungen zu entnehmen, werden multiple Algorithmen angewendet, welche jeweils Vorhersagen bzgl. der Zielklasse einer Problembeschreibung berechnen. Zur Zusammenführung dieser Prädiktionen wird das Areal des Ensemble Learning (vgl. **Kapitel XX**) herangezogen. Da die hier verwendeten Methoden differente Architekturen aufweisen, auf den gleichen Ausgangsdaten trainiert werden und eine Restriktion hinsichtlich der Datenmenge vorliegt, wird sich einem nicht-generativen Ansatz des EL bedient. So wird zur Zusammenführung der vorhergesagten Klassenwahrscheinlichkeiten das Prinzip des *Weighted Averaging* angewendet, wobei die Gewichte der einzelnen Prädiktoren durch deren jeweilige Modellgenauigkeit bestimmt wird. **Abbildung XX** zeigt dies exemplarisch anhand von vier potentiellen Modellen.

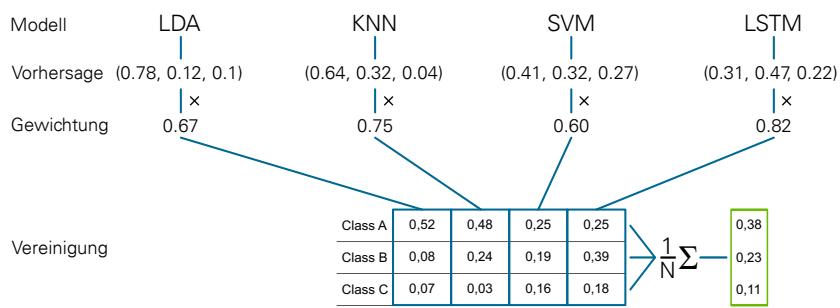


Abbildung 33: Weighted Averaging im Rahmen des Ensemble Learning (Eigene Darstellung)

Wie der Darstellung zu entnehmen werden mittels gewichteter Mittelwertbildung alle Vorhersagen der Modelle zusammengeführt um resultierend eine Aussage zu erlangen.

Neben dieser Zuweisung im Rahmen einer Klassifikation besteht, wie unter **Kapitel XX** bereits aufgezeigt, die Anforderung, eine personalisierte, exemplarische Analyse an den Nutzer zurückzugeben. Diese dient zur Evaluation für den Nutzer, sodass dieser seine Vorstellungen mit dieser Analyse abgleichen kann. Zur Erreichung dieses Ziel wird die unter **Kapitel XX** dargelegte Named-Entity-Recognition verwendet, mittels welcher Entitäten aus dem übergebenen Text extrahiert werden. Zunächst werden die Wörter der Problemstellung, für die aktuelle Untersuchung, mittels POS-Tagging und dem Prinzip des Sentence-Parsings in einen spezifischen Syntaxbaum überführt, der die syntaktischen Bestandteile des Satzes aufzeigt. Anschließend werden die relevanten Entitäten mittels regelbasiertem Chunking extrahiert. Im Allgemeinen werden durch regelbasiertes Chunking reguläre Ausdrücke definiert, die die zu suchende Wortfolgenstruktur wiederspiegeln. Vor dem Hintergrund Analyseentitäten zu extrahieren und der Annahme, dass diese zumeist in Form von Nomen oder Verb-Nomen Kombinationen vorliegen, wird das folgende Suchmuster definiert:

CHUNK: {< NN.>< NN.*>}*

CHUNK: {< DT.>< NN.*>}*

CHUNK: {< V.>< N.*> +}*

Wie hieran zu erkennen werden alle aufeinander folgende Nomen, Nomen, welche auf einen Artikel folgen sowie die Abfolge von Verb und mindestens einem Nomen extrahiert. **Abbildung XX** dient zur Illustration eines exemplarischen Resultats einer derartigen Extraktion.

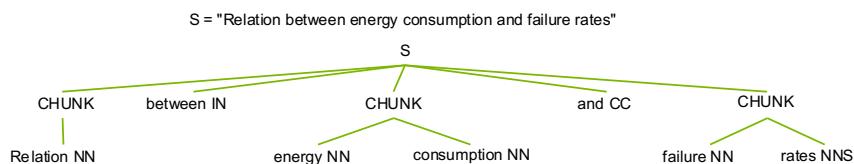


Abbildung 34: Illustration des Vorgehens zur Entitätextraktion (Eigene Darsellung)

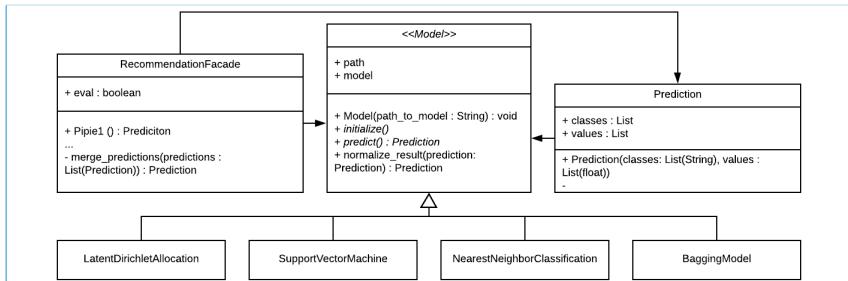
In obenstehender Abbildung ist zu erkennen, dass durch dieses Vorgehen nicht lediglich relevante Entitäten extrahiert werden, sondern ebenfalls alleinstehende Nomen des Satzes. Resultierend aus diesem Sachverhalt werden die Entitäten, ähnlich zum Vorgehen von Keyword-Extraction-Methoden, mittels einer Funktion gewichtet und nach dieser sortiert. Die Bewertung der einzelnen Chunks c wird hierbei zweiteilig durchgeführt.

Zum einen wird die Position des Chunks im Satz P_c berücksichtigt, sodass später auftretenden Entitäten eine höhere Relevanz zugeordnet wird. Dies resultiert aus der Annahme, dass der erste Teil eines Satzes einen mehr einleitenden Charakter aufweist, darauffolgend allerdings das Problem beschrieben wird. Zum anderen wird die Häufigkeit der Entität F_c im Satz bzw. im Dokument mit einbezogen. Final wird die Summe dieser Werte in Relation zur Gesamtlänge der Problemstellung N gesetzt, sodass sich die folgende Bewertungsformel ergibt.

$$\text{score}(c) = \frac{P_c + F_c}{N} \quad (18)$$

Anschließend an diese Sortierung obliegt dem Nutzer die finale Auswahl der Entitäten mithilfe eines implementierten Dialogfensters. Anzumerken ist hierbei, dass aktuelle Algorithmen im Areal des NER durch eine Kombination dieser Methodik mit Klassifikationsalgorithmen des Machine Learning implementiert werden und im Zuge dessen ebenfalls weitere Kriterien, wie exemplarisch die Schreibweise eines Wortes, in die Klassifikation einfließen lassen. Generell ist zu konstatieren, dass diese Algorithmen den lernbasierten bzw. den hybriden NER-Systemen zuordenbar sind und somit eine ausreichend große Datenbasis, mit idealerweise annotierten Daten, benötigt wird (vgl. Jurafsky & Martin, 2008, S. Ch. 22 S.4, 10ff.). Aufgrund der Inexistenz eines derartigen Datensatzes wird hier das oben beschriebene, regelbasierte Vorgehen zur Extraktion und Bewertung verwendet.

Zur Implementierung dieses Ensembles an Modellen und Architekturen wird eine abstrakte Klasse *Model* definiert, welche als Elternklasse aller oben beschriebenen Klassifikationsmodelle dient. So definiert diese Klasse die Mindestfunktionalitäten der einzelnen Modellklassen. Hierbei ist anzumerken, dass zur Initialisierung eines Objektes vom Typ Modell ein Pfad auf das jeweils bereits trainierte Modell angegeben wird. Das Training der jeweiligen Modelle erfolgt demnach außerhalb der illustrierten Programmstruktur, weshalb jedes Objekt dieses Typs die Methode *initialize()* implementieren muss um die jeweiligen Modellformate zu lesen und die Modelle zu initialisieren. Weiterführend wird zur Vereinheitlichung des Formats der Modellvorhersagen eine Klasse *Prediction* implementiert, mit deren Hilfe eine derartige Standardisierung erreicht werden kann.



Kommentiert [m41]: Wie wollen wir NER implementieren -> Müsste hier noch mit rein

Abbildung 35: Klassenstruktur der benötigten Klassen zur Verfahrenszuweisung (Eigene Darstellung)

In Anschluss an diese Implementierung wird das *Facade-Pattern* aus der Softwareentwicklung verwendet. Dieses Musters erlaubt es generell eine einheitliche Schnittstelle auf einer höheren Ebene zu definieren. Hierbei liegt die Motivation in der Reduktion von Abhängigkeiten im Subsystem sowie die hieraus resultierende vereinfachte und weniger fehleranfällige Nutzung des Subsystems durch diese Schnittstelle (vgl. Gamma, Helm, Johnson, & Vlissides, 2015, S. 237f.). In der vorliegenden Implementierung wird eine Fassade für die Klassenzuweisung (Klasse *RecommendationFacade*) implementiert, die die Initialisierung, Anwendung und Zusammenführung aller Klassifikationsalgorithmen steuert.

6.4 Programmstruktur

Das folgende Kapitel gibt nach der Illustration einzelner Komponenten erneut eine Übersicht über die gesamte Programmstruktur. Im Zuge dessen ist im **Anhang in Abbildung XX** erneut das gesamte Klassendiagramm dargestellt. Zur Verknüpfung des Frontends mit dem Backend wird sich für das Framework Flask entschieden, welches eine Anbindung eines Python-Backends an ein Frontend in HTML, CSS, Jinja und Sass erlaubt und somit unter anderem die clientseitige, dynamische Anzeige von Berechnungen ermöglicht. Weiterhin wird die Technologie *Asynchronous Java Script* (AJAX) Rahmen einer Middleware verwendet, um einen asynchronen Datenaustausch zwischen Nutzer und System zu ermöglichen ohne die Webapplikation neu laden zu müssen. **Abbildung XX** illustriert diesen konzeptionellen Aufbau.



Abbildung 36: Struktureller Aufbau des Prototyps (Eigene Darstellung)

Wie dieser Beschreibung zu entnehmen wird sich in der Entwicklung des Systems an dem Architektschema *Model-View-Controller* orientiert. In diesem beschreibt das Model-Objekt das Anwendungsobjekt und umfasst demnach die interne Programmstruktur. Das View-Objekt hingegen repräsentiert die grafische Benutzeroberfläche und umfasst somit die visuelle Repräsentation des Programms. Das Controller-Objekt dient weiterführend zur Verbindung dieser beiden Objekte und steuert die Reaktionen auf Nutzer-eingaben im View-Objekt. Der Vorteil dieser Methodik zeigt sich in der Entkopplung der Programmlogik und der Benutzeroberfläche, was zu höherer Flexibilität und Wiederverwendbarkeit der Software führt (vgl. Gamma et al., 2015, S. 30).

Die Hauptimplementierung des Systems findet sich somit in der Model-Ebene, in welcher ein wiederverwendbares Python-Paket mit dem Titel „rs_helper“ implementiert wird, das nach folgender Struktur konzeptioniert ist:

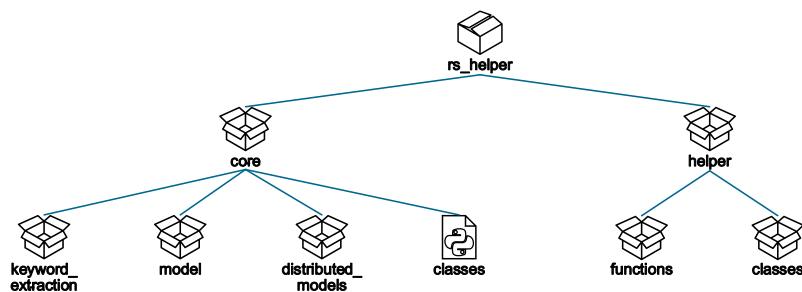


Abbildung 37: Struktur des implementierten Python-Pakets (Eigene Darstellung)

Wie in **Abbildung XX** zu erkennen, werden differente Subpakte definiert, welche zur allgemeinen Struktur des Programms beitragen. Hierbei wird zunächst eine thematische Abgrenzung mit den Subpaketen „core“ und „helper“ vorgenommen, wobei erstere alle generischen Klassen enthält, die hinsichtlich der Lauffähigkeit des Systems essentiell sind. Das Subpaket „helper“ hingegen beinhaltet Methoden und Klassen, welche primär im Rahmen des Trainings sowie der Evaluation, der unter **Kapitel XX** deskribierten, methodischen Herangehensweisen hinzugezogen werden und fortführend verwendet werden können.

6.5 Anforderungsgegenüberstellung

Das folgende Kapitel dient zur Gegenüberstellung der unter **Kapitel XX** aufgestellten Anforderungen hinsichtlich des entwickelten Prototyps. Hierbei wird zunächst auf die funktionalen- und anschließend auf die nicht-funktionalen Anforderungen eingegangen.

Tabelle XX zeigt hierfür die Bewertung der definierten Anforderungen.

Tabelle 9: Gegenüberstellung der Anforderungen

A_ID	Anforderung	Bewertung
A1	Zuordnung einer Problemstellung zu einer Verfahrensklasse	✓
A2	Ausgabe empfohlener Datenstruktur für Verfahrensklasse	✓
A3	Ausgabe einer personalisierten, exemplarischen Analyse	✓
A4	Bessere Zuordnung als reines Raten (Cohens Kappa > 0.5)	✓
A5	Extraktion potentieller Analyseentitäten aus Problemstellung	○
A6	Deterministische Zuweisung zu einer Verfahrensklasse	✓
A7	Eingabe einer kurzen sowie einer langen Problembeschreibung	✓
A8	Entgegenwirken von syntaktischer Divergenz	○
A9	Ausgabe von Wahrscheinlichkeiten der Zuordnung	✓
A10	Einfachheit	✓
A11	Erweiterbare Systemarchitektur	✓
A12	Usability des Systems	✓

A13	Plattformunabhängigkeit	<input checked="" type="checkbox"/>
-----	-------------------------	-------------------------------------

✓: Gegeben; O: Teilweise gegeben

Als primäre Anforderung ist die Zuweisung einer Problemstellung zu einer Verfahrensklasse aufgestellt worden (A1). Die Applikation verschiedener Klassifikationsverfahren des Machine- und Deep Learning sowie des Topic Modeling nach dem oben deskribierten Vorgehen ermöglichen eine derartige Zuweisung. Im Zuge dessen wird eine deterministische Zuweisung mit Wahrscheinlichkeitswerten in A6 und A9 bevorzugt, welche durch die Verwendung fertig trainierter, deterministischer Algorithmen und der Ausgabe von einzelner Zuweisungswahrscheinlichkeiten nach Zusammenführung aller Vorhersagen im Rahmen des Ensemble Learning erreicht werden kann. Hierbei ist anzumerken, dass A6 lediglich im Rahmen des Prototyps gegeben ist, allerdings, bei der Integration der Ähnlichkeit einer Problemstellung zu vergangenen Analysen in den Zuweisungsprozess, nicht gewährleistet werden kann, da dies eine dynamische Datenbasis zur Folge hat. Weiterhin besteht die Anforderung eine Zuordnung zu erreichen, welche eine höhere Vorhersagegüte als Raten aufweist, was einer Genauigkeit von mehr als 50 Prozent entspricht (A4). Im Rahmen des aktuellen Prototyps kann diese Anforderungen aufgrund der differenten Modellgenauigkeiten als gegeben angesehen werden. An dieser Stelle sei allerdings auf das Folgende Kapitel verwiesen, welches die Evaluation aller verwendeten Methoden deskribiert.

Weiterhin beschreibt A8 die Anforderung, dass der unter **Kapitel XX** dargelegten syntaktischen Divergenz in Paragraphen entgegengewirkt werden muss. Als Resultat dieser Anforderung wird sich in der Durchführung der Untersuchung auf die Architektur des DAN zur Inferenz von Paragraph-Embeddings spezialisiert, was dieser Problematik entgegenwirkt. Anzumerken ist jedoch, dass diese Architektur nicht Bestandteil aller aufgestellten Vorgehen bzw. Einordnungen ist, was letztendlich dazu führt, dass diese Anforderung als teilweise gegeben bewertet wird.

Neben dieser Zuweisung fokussiert A3 die Ausgabe einer personalisierten, exemplarischen Analyse an den Nutzer um die berechnete Verfahrensklasse zu illustrieren. Dabei zeigt die Anforderung eine starke Zusammengehörigkeit mit A5 auf, da die hierbei extrahierten Analyseentitäten in der beispielhaften Analyse visualisiert werden sollen. Hinreichlich dieser Anforderungen zeigt **Kapitel XX** die applizierten Methoden zur Erlangung der nötigen Informationen und **Abbildung XX** einen Ausschnitt des Prototyps mit

Kommentiert [m42]: Abbildung mit Screen mit Beispielanalyse in 6.3.3

inkludierter Beispielanalyse. Hierbei werden die potentiellen Analyseentitäten durch ein regelbasiertes Vorgehen im Rahmen des NER sowie durch eine Gewichtung der resultierenden Entitäten erlangt. Dennoch ist Anforderung A5 als teilweise gegeben zu betrachten, da die finale Auswahl der Entitäten durch den Nutzer im Rahmen eines Dialogfensters erfolgt. Parallel wird dem Nutzer, in Referenz zu **Abbildung XX**, eine empfohlene Datenstruktur bzgl. der berechneten Verfahrensklasse dargeboten, welche ebenfalls über die extrahierten Entitäten verfügt (A2).

Als letzte funktionale Anforderung sollen potentielle Nutzer sowohl eine ausführliche sowie eine kurze Beschreibung des Problems übergeben können (A7). Wie bereits in der Frontend Konzeptionierung illustriert, besteht diese Möglichkeit im programmierten Prototyp, weshalb A7 als gegeben bewertet werden kann.

Im Rahmen der nicht-funktionalen Anforderungen ist A11 als relevanteste zu charakterisieren. Durch die Verwendung objektorientierter Programmierung die einfache Erweiterbarkeit des Systems gewährleistet. So besteht die Option neue Modelltypen und -architekturen im Rahmen einer Kindklasse der abstrakten Klasse *Model* zu implementieren. Analog können weitere Embedding-Modelle im Paket *distributed_models* als Kindklasse von *EmbeddingModel* integriert werden. Final sichert die Anwendung des *Facade-Patterns* die problemlose Integration dieser neuen Klassen in den Programmablauf. Die Konzeptionierung des Systems im Architektschema *Model-View-Controller* ermöglicht additional die Anpassung des Backend ohne Auswirkungen auf die vordergründige Applikation, was die fehlerfreie Erweiterung erleichtert.

A10 und A12 hingegen fokussieren die Einfachheit und Nutzbarkeit der Software. Erstes bezieht sich hierbei auf das anwendungsbezogene Verständnis des Nutzers. Zur Erreichung dieser Anforderung findet eine einfache Struktur im Frontend der Software Anwendung (vgl. **Kapitel XX**), welche für den Nutzer möglichst einfach zu überblicken ist. So wird auf der initialen Seite direkt die Möglichkeit gegeben potentielle Problemstellungen zu übergeben, welche durch einen Klick evaluiert werden können. A12, und somit die Nutzbarkeit, welche additionale Aspekte wie beispielsweise die Fehleranfälligkeit und Schnelligkeit des Systems integriert, erweist sich hingegen diffizil in der Bewertung. Generell ist die Usability eines Systems von multiplen Faktoren abhängig und beeinflusst. Sie kann daher nicht vollständig im Rahmen der vorliegenden Arbeit untersucht, sondern sollte in einer anschließenden Untersuchung mit einem ausreichenden Pool an Nutzern evaluiert werden. Zur Teilbewertung dieser Anforderung wird jedoch

am Ende eines jeden Zyklus des ausgewählten Prototyping-Vorgehensmodells untersucht inwieweit diese Aspekte potentiell verbessert werden können. Beispielsweise werden unterschiedliche Bereiche des Systems durch komplexe Datentypen standardisiert um die Kommunikation zwischen verschiedenen Klassen weniger Fehleranfällig zu gestalten. So werden beispielsweise alle implementierten Modell- oder Keyword-Extraction-Klassen forciert standardisierte Objekte zurückzugeben. Die Klassen *Prediction* und *Keyword* repräsentieren dabei solche Objekte, welche anschließend fehlerfrei durch weiterführende Operationen prozessiert werden können. Weiterhin dient das verfolgte Vorgehen des kontinuierlichen Refactoring, durch die Reduktion der Komplexität, zur Senkung der Fehleranfälligkeit sowie zeitgleich zur Erhöhung der Performanz des Systems.

Weiterhin werden, in Referenz zu A13, durch die Implementierung des Systems als webbasierte Technologie potentielle Abhängigkeiten des Systems hinsichtlich unterschiedlicher Betriebssysteme eliminiert. Zusätzlich wird das System in Form eines Docker-Containers gespeichert wodurch ebenfalls der Installationsprozess unabhängig der Plattform durchgeführt werden kann.

7 Evaluation der Methoden sowie des Gesamtsystems

Das folgende Kapitel verfolgt das Ziel eine Übersicht über die applizierten Methoden zur Modell- und Systemevaluation zu geben und bildet somit die Evaluationsphase des anfangs definierten Design-Science-Research-Prozesses ab. Hierbei wird zunächst auf die Evaluation der angewendeten Embedding-Architekturen eingegangen um anschließend die vorgenommene Evaluation von Topic-Modellen sowie von Klassifikationsalgorithmen zu deskribieren.

7.1 Evaluation von Embeddings

Der in **Kapitel XYZ** deskribierte Forschungsstand verdeutlicht, dass in den letzten Jahren viele neue hochqualitative Modelle der distribuierten Semantik entstanden sind. Allerdings herrscht kein eindeutiger Konsens über die Evaluation dieser Verfahren (vgl. Bakarov, 2018, S. 1). Die mittlerweile große Ubiquität von distribuierten semantischen Repräsentationen, hat differente Evaluationsmethoden dieser zum Vorschein gebracht. So lässt sich allgemein zusammenfassen, dass sich zwei differente Interessengruppen gebildet haben. Zum einen liegt das Interesse darin die Performance der distribuierten Repräsentationen zu bewerten und zum anderen rücken linguistische Aspekte dieser in den Vordergrund. Erst genannte messen die Performance anhand sich der anschließenden Aufgabe und die zweitgenannten versuchen rein linguistische Aspekte zu messen (vgl. Schnabel, Labutov, Mimno, & Joachims, 2015, S. 298). Somit soll im Folgenden die Evaluierung der distribuierten Repräsentation in zwei Kategorien erfolgen. Zunächst wird die extrinsische Methodik deskribiert, welche überwachte Verfahren des Machine Learning nutzt, um die Qualität der Repräsentationen zu bewerten (vgl. Bakarov, 2018, S. 3). Dabei bestimmt die resultierende Performance des Verfahrens die Qualität der zugrundeliegenden Embeddings. Nichtsdestotrotz ist eine pure extrinsische Evaluation nicht

Kommentiert [DH43]: Eigentlich müsste hier noch ein Kapitel zur Hyperparameterauswertung hin. Das ist einfach zu geil für den Anhang und wir können Parameterempfehlungen für fortföherende Forschungsvorhaben liefern → Müsste Patrick doch feiern, oder?

ausreichend, um die Qualität der Embeddings zu messen. Eine domänenspezifische Terminologie sowie die Herausforderung Korrelationen zwischen den verschiedenen Machine-Learning-Verfahren aufzuweisen verlangt weitere semantische Evaluierungsmethoden (vgl. Bakarov, 2018, S. 6). Hierbei werden intrinsische Methoden verwendet, um semantische und syntaktische Beziehungen der distribuierten Repräsentationen zu messen (vgl. Schnabel et al., 2015, S. 298).

Im Rahmen der Arbeit steht vordergründig die Performance des zu entwickelnden Systems im Vordergrund. Aus diesem Grund wird zunächst eine extrinsische Evaluation der Methoden vollzogen und anschließen die besten Modelle anhand von intrinsischer Evaluierungsmethoden evaluiert.

7.1.1 Extrinsische Evaluation

Extrinsische Evaluationsmetriken basieren auf der guten Anwendbarkeit von Embeddings in überwachten Machine-Learning-Verfahren. Bei einer rein extrinsischen Evaluation dieser, wird angenommen, dass distribuierte Repräsentation eines Verfahrens auch gute Ergebnisse in anderen nachfolgenden NLP-Anwendungen erzielen (vgl. Bakarov, 2018, S. 3). Die ubiquitäre Anwendung distribuierter Repräsentationen erlaubt eine Evaluierung anhand verschiedenster NLP-Aufgaben. Beispielsweise haben TSVETKOV ET AL. (2015) extrinsische Evaluationen auf Basis distribuierter semantischer Modelle verwendet, um deren Performance anhand von Textklassifikationsaufgaben zu messen. Im Bereich der Sentiment-Analysen haben SCHNABEL ET AL. (2015) mithilfe von extrinsischer Evaluation die Performance distribuierter Repräsentationen evaluiert. Das Spektrum an möglichen NLP-Anwendungen ist jedoch durchaus komplex, weshalb aus Komplexitätsgründen nicht auf weitere Verfahren eingegangen wird (vgl. Bakarov, 2018, S. 4ff.).

Für das Auffinden der optimalen Modellkombinationen wird ein GridSearch⁹ ähnliches Vorgehen gewählt. Hierbei wird auf Basis der Kombination verschiedenster Parameter-

⁹ Grid-Search ist eines der klassischen Methoden, um die idealen Hyperparameter eines Modells zu identifizieren. Dabei bedient sich das Verfahren der Exhaustionsmethode (engl.: *Brute-Force*), auf Basis manuell erstellter Parameterkonfigurationen, um die beste Hyperparameterkonfigurationen für ein Modell zu identifizieren (vgl. Bergstra & Bengio, 2012, S. 282ff.).

ausprägungen jeweils ein separates Embedding-Modell trainiert und anschließend anhand eines überwachten Klassifikationsverfahrens evaluiert. Die Parameterkombination mit der höchsten Modellgenauigkeit stellt somit die Ausgangsbasis für das finale Modelltraining dar. Das zugrundeliegende Klassifikationsverfahren stellt hierbei eine multinomiale Support-Vector-Machine dar, mit einem linearen - und radialen Kernel als Ausgangsbasis und einer Penalty-Value von 12. Der Gamma-Wert der SVM-Modelle wird dabei durch die Anzahl der Features N und der Standardabweichung des Datensatzes bestimmt und wie folgt berechnet:

$$\frac{1}{N * \sqrt{\frac{1}{N} * \sum_{i=1}^n (x_i - \bar{x})^2}}$$

Es ist zu betonen, dass die SVM als Klassifikator rein arbiträr gewählt wurde und jedes andere Klassifikationsverfahren ebenfalls für die extrinsische Evaluation geeignet wäre. Die Datengrundlage bilden hierbei die in **Kapitel XYZ** deskribierten selektierten Daten. Für die Operationalisierung der Ergebnisse wird sich die Modellgenauigkeit herangezogen, um die Performance des Modells zu berechnen. Dabei wird diese durch das Verhältnis an richtig vorhergesagten Klassen zu allen gegebenen Klassen bestimmt. Das grundlegende Vorgehen zur Evaluation der distribuierten Modelle ist in [Anhang XYZ](#), in Form eines Prozesses deskribiert.

Kommentiert [m44]:

FastText

In Bezug auf das oben erläuterte Vorgehen ergeben sich insgesamt 32 verschiedene Modellmöglichkeiten¹⁰. Aufgrund der sich ergebenden Rechenintensität, der möglichen Modellkombinationen wird sich auf vier essentielle FastText-Parameter beschränkt. Die nachfolgende Abbildung illustriert ausgewählte Korrelationen zwischen den Modellparametern sowie den resultierenden Modellgenauigkeiten, basierend einer SVM mit linearer Kernelfunktion.

¹⁰ Die möglichen Parameterkombinationen finden sich im beigefügten Jupyter-Notebook: „**Titel**“

Ausgewählte Spearman-Korrelationen

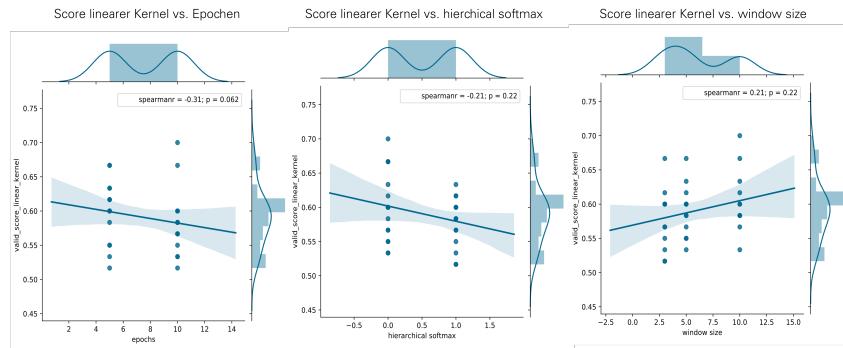


Abbildung 38: Ausgewählte Spearman-Korrelationen zwischen Modellgenauigkeit und Modellparametern (FastText) (Eigene Darstellung)

Auf Basis, der in **Anhang XYZ** abgebildeten Evaluationsergebnissen, lässt sich konstatieren, dass eine höhere Filtergröße der Embedding-Modelle, zu höheren Modellgenauigkeiten führt. Hinzukommt, dass eine steigende Anzahl an Epochen oder ein höherer Wert bei Hierarchischen-Softmax-Verfahren zu geringeren Modellgenauigkeiten führt.

Basierend auf der distributionellen Hypothese sollten semantisch ähnliche Wörter bzw. Paragraphen im Vektorraum näher zueinander liegen als unähnliche (vgl. Y. Zhang et al., 2016, S. 5). Die nachfolgende Abbildung illustriert mithilfe einer klassischen BoW-Kompositionsfunktion (vgl. **Kapitel XYZ**) die Paragraphvektoren des selektierten Datensatzes (vgl. **Kapitel XYZ**).

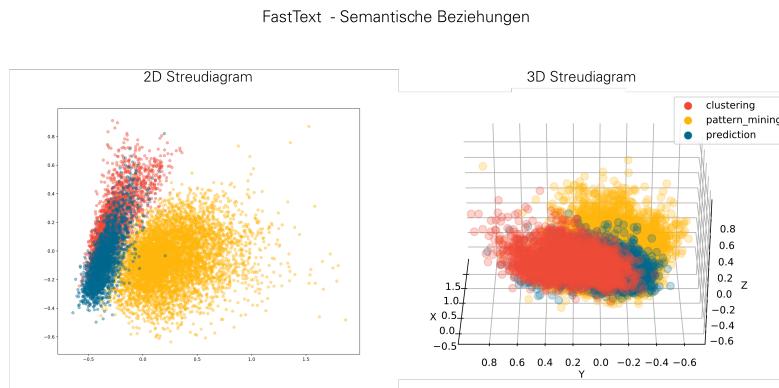


Abbildung 39: Scatter Plots des besten FastText Modells auf Grundlage XX (Eigene Darstellung)

Wie oben zu erkennen erfolgt, zwar eine semantische Diskriminierung der Daten zwischen den Klassen. Allerdings gibt es starke semantische Überschneidungen zwischen der „Clustering“- und der „Prediction“-Klasse. Die in **Kapitel XYZ** beschriebene intrinsische Evaluation, untersucht die semantische Aussagekraft der fünf besten evaluierten FastText-Modelle mithilfe geeigneter Evaluationsmetriken.

Deep-Averaging-Networks

In Angesicht des oben erläuterten Vorgehens, ergeben sich insgesamt 480 verschiedene Parameterkombinationen für das Training von DAN-Modellen. Auf Basis der errechneten Modellgenauigkeiten und den zugrundeliegenden Parameterkombinationen ergeben sich Korrelationen zwischen den gewählten Parametern und der resultierenden Modellgenauigkeit. Alle Spearman-Korrelations-Koeffizienten zwischen den Parameter und der resultierenden Modellgenauigkeiten sind im **Anhang XYZ** aufgezeigt.

Die Abbildung im **Anhang XYZ** illustriert, dass es nahezu keinerlei Korrelationen zwischen den eigentlichen Hyperparametern des Modells gibt. Allerdings ist eine leichte negative Korrelation zwischen der Hidden-Layer-Anzahl und der Modellgenauigkeit (*valid_score_linear_kernel*) sowie der Word-Dropout-Wahrscheinlichkeit und der Modellgenauigkeit zu konstatieren.

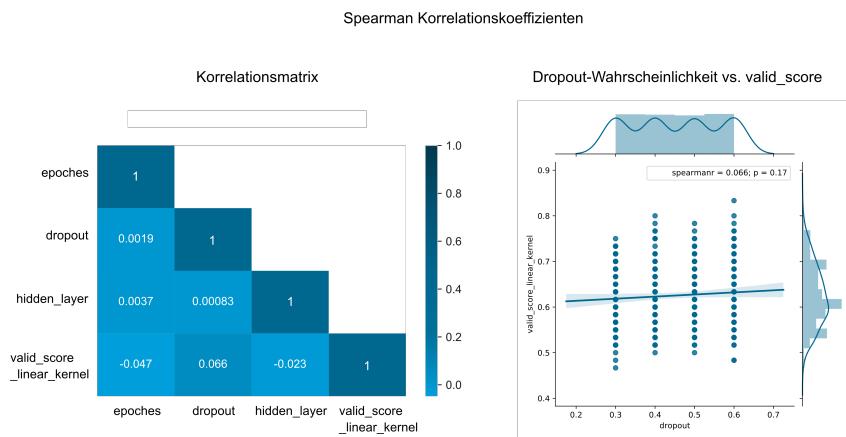


Abbildung 40: Ausgewählte Spearman-Korrelationen zwischen Modellgenauigkeit und Modelparametern (DAN) (Eigene Darstellung)

Dies bedeutet, dass eine steigende Anzahl an Hidden-Layern eine negative Auswirkung auf potentielle Modellergebnisse hat. Die oben dargestellte Korrelationsmatrix, veranschaulicht, dass die besten Modelle eine höhere Wort-Dropout-Wahrscheinlichkeit besitzen als Modelle mit einer geringeren Wahrscheinlichkeit. Dies unterstreicht die Ergebnisse von IYER ET AL. (2015b), welche bessere Modellgenauigkeiten durch höhere Wort-Dropout-Wahrscheinlichkeiten erzielen. Zurückführen lässt sich dies, auf eine potentielle bessere Generalisierungsfähigkeit, der resultierenden distribuierten Repräsentationen.

Eines der elementaren Gründe warum DANs Anwendung in der vorliegenden Arbeit finden, ist die Fähigkeit syntaktische Divergenzen zu überbrücken (vgl. Iyyer et al., 2015b, S. 1687). Die folgende Abbildung illustriert beispielhaft die diskriminierenden Eigenschaften eines DANs. Hierbei sind die besten Modelle, kategorisiert nach ihrer Layer Anzahl, von links nach rechts dargestellt. Dabei wird sich nur auf die Schichten bezogen, welche zwischen den Average-Layer und den sich anschließenden vollvernetzten Klassifikations-Layern liegen.

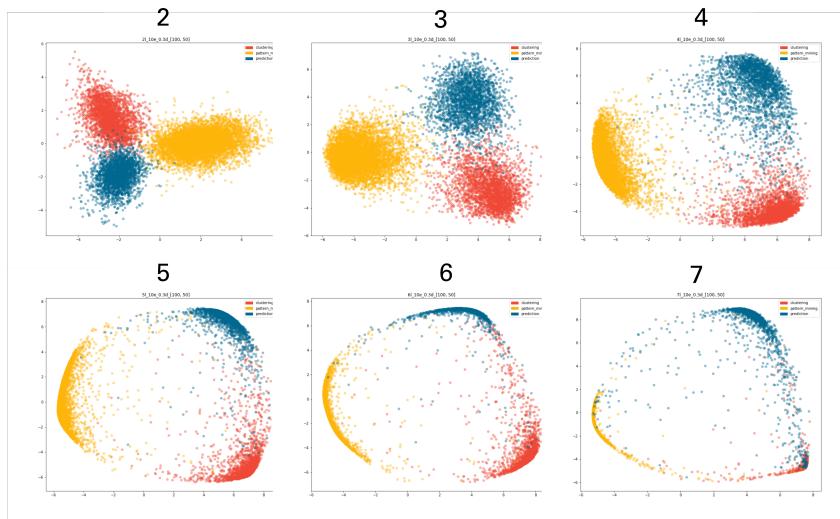


Abbildung 41: Deep-Averaging-Netzwerke mit mutipler Layer-Anzahl (Eigene Darstellung)

Im Vergleich zu **Abbildung XYZ** ist gut zu erkennen, dass durch das Verwenden eines zusätzlichen DAN-Modells klassische Mittelwertverfahren besser separiert werden, was sich profitabel in nachfolgenden Klassifikationsverfahren erweisen kann. Hinzukommt, dass die Streudiagramme die Erkenntnisse aus der Korrelationsanalyse unterstreichen. So separieren Modelle mit wenigen Hidden-Layern, vor dem Klassifikations-Layern, mehr in sich geschlossene konvexe Cluster, mit wenigen semantischen Ausreißern. Wohingegen Modelle mit einer hohen Hidden-Layer-Anzahl mehr nicht-konvexen Cluster bündeln und auf den ersten Blick mehr semantische Ausreißer beinhalten. Die Anzahl an potentiellen semantischen Ausreißern, beeinträchtigen sich anschließende überwachte Verfahren des Machine Learning, weshalb es relevant ist, diese in der Erstellung von DANs zu vermeiden (vgl. Rehm, Klawonn, & Kruse, 2007b, S. 489).

Kommentiert [DH45]: FastText

7.1.2 Intrinsische Evaluation

Im Rahmen der intrinsischen Evaluation finden differente Kennzahlen Anwendung, welche Aussagen über die Qualität der jeweiligen Modelle liefern ohne diese in speziellen Applikationen anzuwenden. Neben der bereits in **Kapitel XX** untersuchten **Perturbation**

wird ebenfalls die sog. *Purity* (z.Dt. Reinheit) der, durch die Anwendung des Embedding-Modells auf die Ausgangsdaten erlangten, Cluster fokussiert. Die Intuition bei einer derartigen Kategorisierung von Wörtern ist es, distribuierte Repräsentationen ihren zugrundeliegenden Gruppen durch Clustering-Algorithmen zuzuordnen. Jene Repräsentationen, werden anhand eines Clustering-Modells in n verschiedene Gruppen unterteilt, wobei n die Anzahl der natürlich zugrundeliegenden Wortkategorien des Korpus darstellt. Die letztendliche Qualität der Embeddings wird mithilfe einer extrinsischen Clustering-Evaluationsmetrik bestimmt (vgl. Baroni, Dinu, & Kruszewski, 2014, S. 241; Schnabel et al., 2015, S. 301). Die Herausforderung besteht hierbei zum einen in der Auswahl des richtigen Clustering-Verfahrens, zum anderen im Finden einer geeigneten Evaluationsmetrik (vgl. Bakarov, 2018, S. 12). Besonders in Hinsicht der hochdimensionalen Daten ist das zu wählende Clustering-Verfahren von essentieller Bedeutung, um die Funktionalität der Clustering-Algorithmen und somit die Belastbarkeit der Ergebnisse zu gewährleisten (vgl. Pavithra, 2017, S. 289ff.).

Im Rahmen der vorliegenden Arbeit wird sich auf ein partitionierendes Verfahren beschränkt. Der Vorteil eines partitionierenden Verfahrens liegt hierbei in der notwendigen, initialen Bestimmung der Anzahl der zu erstellenden Cluster, welche hier durch die Menge der abgezielten Verfahrensklassen bestimmt werden kann und somit erlaubt additional die Unterscheidbarkeit der Klassen im Vektorraum einzuschätzen. Im Rahmen des aktuellen Forschungsvorhabens wird sich hierbei des KMeans-Algorithmus bedient, da er simpel in seiner Anwendung ist und weniger rechenintensiv als andere partitionierende Verfahren (vgl. Bhukya & Ramachandram, o. J., S. 46ff.).

Bei der Auswahl der Evaluationsmetrik, orientiert sich die vorliegende Forschungsarbeit an dem Vorhaben von BARONI ET AL. (2014), wobei die Evaluationsmetrik der Purity (z.Dt. Reinheit) Aussage über die Reinheit der extrahierten Gruppierungen gibt. Im Falle einer perfekten Wiedergabe der zugrundeliegenden Labels durch die berechneten Cluster liegt die Reinheit dieser bei 100%, wohingegen dieser Wert bei zunehmender Unreinheit gegen Null konvergiert. Nachfolgend sei die Formel für die Berechnung der Cluster-Purity dargelegt:

$$purity(C, K) = \frac{1}{N} \sum_k \max_j |c_k \cap k_j| \quad (19)$$

Sei $C = \{c_1, c_2, \dots, c_K\}$ die Menge an extrahierten Clustern und $K = \{k_1, k_2, \dots, k_j\}$ die Menge der zugrundeliegenden Klassen, so wird die Reinheit aus der Summe der Objektanzahl der am häufigsten vorkommenden Klasse eines Clusters berechnet, multipliziert mit $1/N$ wobei N gleich die Summe aller Datenpunkte beschreibt.

Weiterführend wird ein Ähnlichkeitsbasiertes Bewertungskriterium für Embedding-Modelle herangezogen. Hierbei werden die Ähnlichkeiten zwischen aufgestellten Beispielsätzen in Form von Paragraph-Embeddings in einer Matrix visualisiert um Aufschluss über die Differenzierungsfähigkeit des Embedding-Modells zu erlangen. **Abbildung XX** zeigt eine derartige Matrix, welche für jedes trainierte Modell berechnet wird.

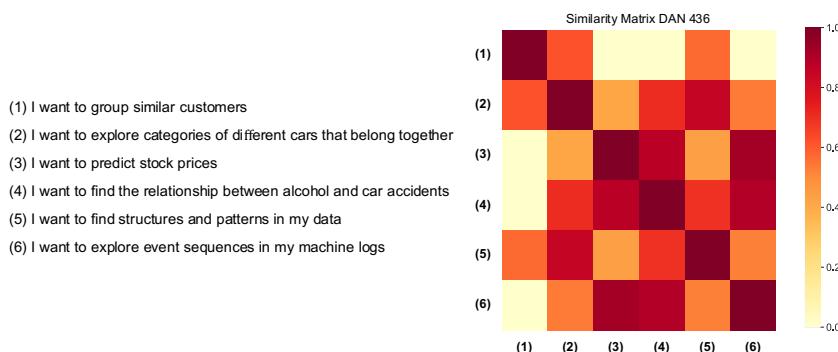


Abbildung 42: Ähnlichkeitsmatrix zur Evaluierung von Embedding-Modellen (Eigene Darstellung)

Abschließend erfolgt in Anlehnung an BOJANOWSKI ET AL. (2017) additional eine Modellbewertung anhand von Ähnlichkeitswerten vordefinierter Wortpaare. Hierfür wird die *Word-Similarity 353 Test Collection* nach FINKELSTEIN ET AL. (2001) verwendet, da dieser Datensatz Wortpaare mit bereits manuell annotierten Ähnlichkeitswerten umfasst.

$$ME = \sum_{i=1}^N \frac{x_i - \bar{x}_i}{N} \quad (20)$$

$$r_{SP} = \frac{\sum_{i=1}^N ((rg_{x_i} - rg_{\bar{x}})(rg_{y_i} - rg_{\bar{y}}))}{\sqrt{\sum_{i=1}^N (rg_{x_i} - rg_{\bar{x}})^2} \sqrt{\sum_{i=1}^N (rg_{y_i} - rg_{\bar{y}})^2}} \quad (21)$$

In Anschluss hieran werden diesen Werten, die mittels des Modells errechneten Werte gegenübergestellt um sowohl den Mean Error (z.Dt.: gemittelter Fehler) (**XX**) sowie die

Spearman Rangkorrelation (**XX**) der Ähnlichkeitswerte nach den oben illustrierten Formeln zu errechnen. Diese können weiterführend zur Bewertung der Modelle herangezogen werden (vgl. Bojanowski et al., 2017, S. 139ff.).

Anzumerken ist hierbei, dass die Ergebnisse der extrinsischen Evaluationsschritte sowie der oben deskribierten Ähnlichkeitsmatrix eine höhere Relevanz für die aktuelle Untersuchung aufweisen. Dies resultiert zum einen aus der Zusammensetzung des verwendeten Datensatzes nach FINKELSTEIN ET AL. (2001), da dieser eine große Bandbreite abdeckt und somit allgemeingefasste Wortpaare beinhaltet. Zum anderen zeigt die Ähnlichkeitsmatrix die Differenzierbarkeit der Embeddings hinsichtlich der untersuchten Verfahrensklassen und somit potentielle Auswirkungen auf die Güte der fortlaufend angewendeten Klassifikationsalgorithmen.

7.1.3 Ergebnisse der Evaluation

Insgesamt werden im Rahmen der vorliegenden Untersuchung 36 verschiedene FastText-Modelle sowie 480 Deep-Averaging-Netzwerke trainiert und gegenübergestellt. Wie im **Anhang unter XX** zu sehen, ist das Vorgehen hierbei als Zyklus zu verstehen, da aufbauend auf den Ergebnissen der Evaluation von FastText-Modellen, verschiedene DAN-Modelle mit unterschiedlichen Hyperparametern trainiert werden. Der Umfang dieser Trainingsschritte sowie deren Evaluation resultiert aus der Inferenz von Paragraph-Embeddings mittels des DAN-Modells auf Basis von Wordvektoren des Fast-Text-Modells sowie dem Sachverhalt, dass diese ausschlaggebend für die Qualität weiterer Methoden im Bereich der Klassifikation sind. Die folgende Tabelle gibt eine Übersicht der verwendeten Hyperparametern, welche aufgrund der jeweiligen Kombinationen dieser in den Anzahlen der trainierten Modelle münden.

Tabelle 10: Übersicht der trainierten Parameterkombinationen für Embedding-Modelle

Parameter	Mögliche Werte
FastText	
<i>Epochen</i>	[5, 10]
<i>Minimale Häufigkeit</i>	[3, 5, 8]
<i>Fenstergröße</i>	[3, 5, 10]
<i>Hierarchische Softmax-Funktion</i>	[0, 1]

Deep-Averaging-Networks

<i>Epochen</i>	[10, 50, 100]
<i>Anzahl an Layern</i>	[2, 3, 4, 5, 6, 7, 8]
<i>Dropout-Wahrscheinlichkeit</i>	[0.2, 0.3, 0.5]
<i>Form des Classifiers</i>	[[200], [100,50], [200,100]]

Wie eingangs bereits dargelegt ergeben sich insgesamt 36 FastText Modelle, welche auf Basis der intrinsischen sowie der extrinsischen Evaluation bewertet werden, wobei extrinsischen Ergebnissen eine höhere Relevanz zugestanden wird. **Tabelle XX** illustriert die Ergebnisse der Evaluation der einzelnen FastText-Modelle am Beispiel der besten vier Modelle, welche primär anhand der Güte des Modells der angeschlossenen Klassifikationsaufgabe auf den definierten Validierungsdaten ausgewählt werden. Dies resultiert zum einen aus der bereits dargelegten höheren Relevanz der extrinsischen Ergebnisse. Zum anderen dienen die Ergebnisse der intrinsischen Evaluation der Kontrolle der Modelle hinsichtlich deren internen Struktur, damit ausgewählte Modelle keine starken, internen Inkonsistenzen aufweisen.

Tabelle 11: Ergebnisübersicht der Ähnlichkeitsevaluation von Embedding-Modellen

Modell	EP	WS	MC	HS	Valid linear	Valid RBF	P	ME	r_{SP}
<i>fasttext_31</i>	10	10	5	0	0,70	0,633	0.752	0,232	0,344
<i>fasttext_12</i>	5	10	3	0	0,667	0,717	0.752	0,224	0,344
<i>fasttext_1</i>	5	3	5	0	0,667	0,633	0,764	0,203	0,342
<i>fasttext_25</i>	10	5	5	0	0,667	0,567	0.761	0,225	0,319

EP: Epochen; WS: Fenstergröße; MC: Min. Häufigkeit; HS: Hierarchische Softmax-Funktion; P: Purity

Wie in dieser Tabelle zu erkennen, erreichen die Modelle *fasttext_31* und *fasttext_12* die besten Ergebnisse in Klassifikationsaufgaben, wohingegen in den intrinsischen Kennzahlen lediglich sehr geringe Unterschiede zu verzeichnen sind.

Wie bereits aufgezeigt erfolgt auf Basis der Auswahl der besten fünf FastText-Modelle das Training der DAN-Modelle. Parallel zu den eben deskribierten Modellen werden hier

die Ergebnisse der extrinsischen Evaluation zur Auswahl der besten Architektur herangezogen sowie die der intrinsischen zur Kontrolle der internen Struktur. **Tabelle XX** gibt eine Übersicht über die besten Modelle hinsichtlich deren Generalisierungskraft¹¹.

Tabelle 12: Übersicht der Evaluationsergebnisse von DAN-Modellen

Modell	FastText	EP	DR	SH	HL	Valid linear	Valid RBF	P	ME	r_{sp}
DAN_436	fasttext_12	50	0.6	[100, 50]	5	0,833	0,733	0,975	0,307	0,199
DAN_411	fasttext_12	10	0.5	[100, 50]	5	0,783	0,75	0,971	0,332	0,205
DAN_279	fasttext_31	50	0.6	[200]	4	0,8	0,7	0,973	0,302	0,099
DAN_246	fasttext_12	10	0.4	[200]	4	0,8	0,733	0,967	0,329	0,18

EP: Epochen; DR: Dropout; SH: Form des Classifiers; HL: Hidden Layer; P: Purity

Hieran ist gut zu erkennen, dass durch die Anwendung der DAN-Architektur ein Unterschied in der Modellgüte von ca. 13% erreicht werden kann. Zeitgleich ist ein Anstieg in der Purity und der damit verbundenen Zusammengehörigkeit der Klassen im Vektorraum von ca. 21% zu verzeichnen. Auf der anderen Seite ist eine Verschlechterung hinsichtlich des gemittelten Fehlers von Wortähnlichkeiten sowie deren Rangkorrelation auffällig. Dies ist primär auf die Natur eines DAN-Modells zurückzuführen, da dieses als ungeordnete Kompositionsfunktion das Ziel verfolgt Paragraph-Embeddings zu kreieren. Die Bewertung einzelner Wörter, wie es in den Ähnlichkeitsbewertungen der Fall ist, ist demnach nicht im Fokus der Architektur.

Final wird sich zur weiteren Implementierung für das FastText-Modell *fasttext_12* sowie das DAN-Modell *DAN_436* entschieden. Dies Wahl des FastText-Modells resultiert hier aus der vergleichsweise hohen Generalisierungsfähigkeit des Modells sowie aus dem Sachverhalt, dass die besten der trainierten DAN-Modelle auf Basis des dieses Modells agieren. Das *DAN-Modell 436* zeigt sich zudem in der Güte des Classifiers der anschließenden Klassifikationsaufgabe als bestes Modell.

¹¹ Eine vollständige Übersicht der trainierten FastText- und DAN-Modelle findet sich im elektronischen Anhang der Arbeit im Dokument „ wieder.

7.2 Evaluation von Topic-Modellen

Das folgende Kapitel dient zur Darstellung der verwendeten Evaluationsmethoden in Hinsicht auf Topic-Modelle. Generell ist hierbei anzumerken, dass sich die Evaluation von Topic-Modellen als diffizil herausstellt, da ein Topic ein reales semantisches Konstrukt repräsentiert, die tatsächliche Passgenauigkeit der Wörter allerdings nicht erfasst werden kann. Somit bestehen, ähnlich zum Areal der Embeddings, in der Literatur eine Vielzahl an Kennzahlen, die zur Bewertung der berechneten Topics herangezogen werden können, allerdings die Ähnlichkeit der Wörter in einem Topic fokussieren und somit dessen Kohärenz bewerten. Ob letztendlich das erwartete semantische Konstrukt allgemeingültig und korrekt abgebildet wird ist demnach nur indirekt zu bewerten.

7.2.1 Überblick über Evaluationsmethoden

NEWMAN, LAU, GRIESER & BALDWIN (2010) geben in ihrer Publikation eine Übersicht über mögliche Bewertungskriterien von Topics, die teilweise auf differenten externen Quellen basieren. Die meisten deskribierten Metriken folgen hierbei demselben Vorgehen und fokussieren die Bewertung der Ähnlichkeit $S(w_i, w_{i+1})$ aller Wortpaare $t_{pairs} = ((w_1, w_2), (w_1, w_3), \dots, (w_n, w_{n-1}))$ eines Topics t (vgl. David Newman et al., 2010, S. 102ff.). Die Hauptunterschiede der Methoden bestehen in der Art der Bewertung der Ähnlichkeit von Wörtern. So spezifizieren LEACOCK, CHODOROW & MILLER (1998) die Verwendung der WordNet-Ontologie zur Berechnung des kürzesten Pfads zwischen zwei Wörtern $sp(w_1, w_2)$.

$$S(w_1, w_2) = -\log \left(\frac{sp(w_1, w_2)}{2 * D} \right) \quad (22)$$

Diese wird anschließend in Relation mit der maximalen Tiefe von WordNet D gesetzt und logarithmiert (**XX**).

WU & PALMER (1994) hingegen beschreiben eine Kennzahl, welche die Tiefe der zu vergleichenden Wörter mit der Tiefe ihres sog. *Least common subsumer* in Relation setzt. Dieser beschreibt dabei den tiefsten gemeinsamen Knoten in der Ontologie WordNet. Demzufolge wird folgende Formel zur Berechnung der Ähnlichkeit herangezogen:

$$S(w_1, w_2) = \frac{2 * \text{depth}(\text{LCS}_{w_1, w_2})}{\text{depth}(w_1) + \text{depth}(w_2) + 2 * \text{depth}(\text{LCS}_{w_1, w_2})} \quad (23)$$

Weiterhin zeigt Schütze (1998) ein Verfahren Vektoren aus dem Kontext der Wörter im Korpus zu Generieren und diese zur Bewertung der Ähnlichkeit heranzuziehen. Hierbei fokussiert der Autor Co-Occurrence-Matrizen zur Erstellung der Vektoren. Auf Basis der bereits aufgezeigten Methoden besteht allerdings die Option zuvor trainierte Embeddings und das Maß der Kosinusähnlichkeit zwischen den Wörtern zu verwenden und diese über das gesamte Topic zu mitteln.

Darüber hinaus definieren NEWMAN ET AL. (2010) weitere Methoden um die Ähnlichkeit mithilfe von Wikipedia-Einträgen und der enthaltenen gegenseitigen Verlinkung von Artikeln der Wörter zu berechnen. Exemplarisch ist hier der sog. *Related Article Concept Overlap* $raco_{w_i, w_{i+1}}$ (RACO) anzusprechen, der die Überschneidung der Menge an Konzepten berechnet, auf welche die jeweiligen Wörter in Wikipedia verlinken. Hierbei beschreibt $ol(w_i)$ die ausgehenden Links eines Artikels und $cat(l)$ die Kategorien, welche dem verlinkten Artikel zugeordnet sind.

$$raco_{w_i, w_{i+1}} = \frac{|(\cup_{l \in ol(w_i)} cat(l)) \cap (\cup_{l \in ol(w_{i+1})} cat(l))|}{|(\cup_{l \in ol(w_i)} cat(l))| + |(\cup_{l \in ol(w_{i+1})} cat(l))|} \quad (24)$$

Um differenten Artikellängen und Mengen an Verlinkungen entgegenzuwirken inkludieren die Autoren den Jaccard-Koeffizienten (**XX**) (vgl. David Newman et al., 2010, S. 104). Diese Art der externen Evaluation der Topics über Wikipedia bietet den Vorteil, dass Wikipedia eine Vielzahl an explizit, manuell definierten semantischen Beziehungen beinhaltet und somit gut das allgemeine Verständnis von differenten semantischen Konstrukten wiederspiegelt. Die in Wikipedia erstellen Verlinkungen weisen somit eine hohe Eignung zur Evaluation von Topics auf (vgl. Milne & Witten, 2008, S. 26).

Additional existieren weitere Kennzahlen, wie bspw. der WordNet-basierte *Resnik Information Content* nach RESNIK (1995) oder die suchmaschinenbasierten Methoden nach NEWMAN ET AL. (2009), welche allerdings aufgrund des Umfangs hier nicht näher beleuchtet werden. Zudem zeigen BOUGOUIN ET AL. (2013), CAMPOS ET AL. (2018) und FLORESCU ET AL. (2017) die Evaluierung einzelner Algorithmen der Keyword Extraction anhand der Qualitätskennzahlen Precision, Recall sowie F1-Score um zu evaluieren, wie viele der im Korpus enthaltenen Schlüsselwörter durch den jeweiligen Algorithmus ex-

trahiert werden. Aufgrund der Notwendigkeit annotierter Daten für eine derartige Evaluierung und deren Unverfügbarkeit im Rahmen der aktuellen Untersuchung, wird sich jedoch lediglich auf die Kohärenz der extrahierten Topics fokussiert.

Neben dieser Bewertung, welche aus Methoden der Keyword Extraction bzw. des Topic Modeling resultieren, besteht, im Rahmen von Modellen des Topic Modeling, die Möglichkeit die Passgenauigkeit der Modelle hinsichtlich unbekannter Daten und somit deren Generalisierungsstärke zu bewerten. Hierbei werden in aktueller Literatur häufig die Kennzahlen der Perplexität (engl. Perplexity) sowie der Log-Likelihood angewendet. Da der Wert der Perplexity bei steigender Log-Likelihood des Testdatensatz monoton absinkt, wird im Folgenden lediglich jenes Maß deskribiert. Perplexity beschreibt das Inverse der wortspezifischen, logarithmierten Wahrscheinlichkeit eines ungewöhnlichen Testdatensatzes, welche mit der Anzahl der Wörter N im Testdatensatz normalisiert ist.

$$\text{Perplexity} = \exp \left\{ -\frac{\sum_{i=1}^M \log p(w_i)}{\sum_{i=1}^M N_i} \right\} \quad (25)$$

Resultierend ergibt sich, dass ein geringerer Wert dieser Kennzahl eine höhere Generalisierungskraft des Modells indiziert (Blei, Ng, & Jordan, 2003, S. 1008). Jedoch ist anzumerken, dass dieses Maß, aufgrund fehlender Skalierung sowie der Missachtung von potentiellen semantischen Relationen in den Topics, lediglich zum Vergleich zwischen differenten Topic Modellen verwendet wird.

7.2.2 Ergebnisse der Evaluation von Topic-Modellen

Das folgende Kapitel fokussiert die Ergebnisdarstellung der erstellten Topic-Modelle und unterteilt sich dahingehend in die Evaluation von Keyword-Extraction-basierten Topics sowie in die der trainierten Modelle des Topic-Modeling.

Generell werden im Rahmen der aktuellen Untersuchung die sechs vorgestellten und deskribierten Keyword-Extraction-Algorithmen TFIDF, TFIGM, TextRank, YAKE, PositionRank sowie TopicRank angewendet. Hierbei werden jegliche Kombinationen der Algorithmen angewendet, woraus insgesamt 61 verschiedene Topics pro zugrundeliegender Zielkategorie (Clustering, Prediction sowie Pattern Mining) resultieren, die gegenübergestellt und evaluiert werden müssen, um auf die final zu verwendende Kombination zu

schließen. Die folgende Tabelle gibt eine Übersicht der durchgeföhrten Evaluationen anhand der besten Kombinationen, wobei die Summe der differenten Kohärenzkennzahlen sowie die durchschnittliche Kohärenz pro Topic zur Auswahl der adäquaten Kombination verwendet wird.

Tabelle 13: Evaluationsergebnisse von Topic-Modellen

Kombination	Topic	FT	LC	WP	Σ
<i>tfigm+positionrank+textrank</i>	clustering	0,951	1,262	0,292	2,504
<i>tfigm+positionrank+textrank</i>	pattern_mining	0,939	1,271	0,285	2,494
<i>tfigm+positionrank+textrank</i>	prediction	0,938	1,267	0,288	2,493
<i>tfigm+textrank</i>	pattern_mining	0,939	1,263	0,284	2,485
<i>tfigm+textrank</i>	clustering	0,951	1,240	0,289	2,479
<i>tfigm+textrank</i>	prediction	0,938	1,257	0,281	2,476
<i>tfigm+positionrank</i>	clustering	0,951	1,233	0,295	2,479

Kennzahlen: FT: FastText-Kohärenz; LC: Leacock-Kohärenz; WP: Wu & Palmer-Kohärenz

Zur vereinfachten Darstellung der Werte aller Kombinationen sowie der extrahierten Topics dient folgende Abbildung, welche die summierten Kohärenzwerte pro Modell aggregiert aufzeigt.

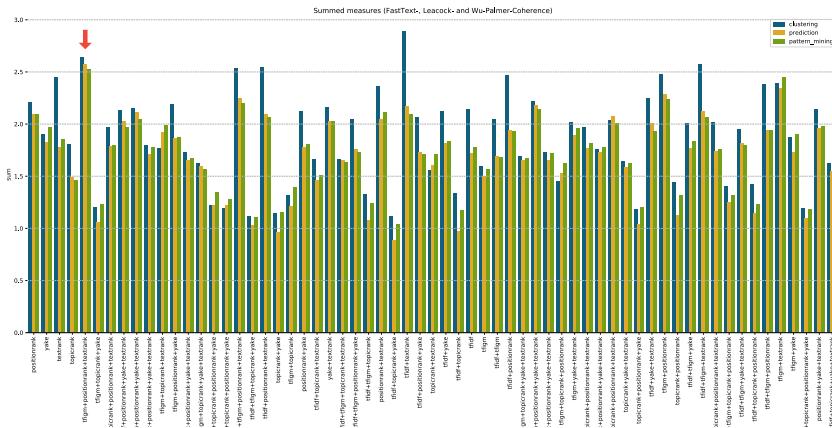


Abbildung 43: Illustration der Evaluationsergebnisse der Keyword Extraction (Eigene Darstellung)

Als Resultat dieser Analyse ist festzustellen, dass die Kombination der Algorithmen TFIGM, PositionRank und TextRank mit der Summe von ca. 2,50 die höchste Kohärenz bzgl. aller drei generierten Topics aufweist (Roter Pfeil in **Abbildung XX**) und somit zum Aufbau der Wissensbasis Verwendung findet.

Hinsichtlich der Evaluierung von LDA-Modellen wird sich der Methodik des GridSearch nach GOODFELLOW ET AL. (2016) bedient, welche auf Basis gegebener Kombinationsmöglichkeiten an Modellparametern differente Varianten des Modells trainiert und diese mittels Cross-Validation und anhand einer Bewertungsfunktion evaluiert, um die optimale Verteilung an Parametern zu extrahieren (vgl. Goodfellow et al., 2016, S. 432f.). Hierbei wird die mögliche Anzahl an Topics mit den Werten 3 bis 9, der Parameter Alpha mit den Werten 0.05, 0.1, 0.2, 0.3 und 0.4 sowie Beta mit 0.1, 0.2, 0.3 und 0.4 gewählt. Zur Bewertung im Rahmen der Cross-Validation wird eine Teilung des Datensets in fünf Teile fokussiert um die jeweilige Trainingsdatenmenge nicht zu stark zu reduzieren. Diese Angaben resultieren der Annahme, dass eine hohe Wahrscheinlichkeit vorliegt, dass ein Dokument lediglich über ein Topic verfügt. Resultierend werden insgesamt 600 LDA-Modelle trainiert, aus welchen das mit dem besten Testergebnissen ausgewählt wird. Weiterhin wird der Abstract-Datensatz verwendet um die Wahrscheinlichkeit mehrerer Topics, in Referenz zum definitionsähnlichen, gefilterten Datensatz, zu erhöhen. Aufbauend auf dieser Evaluation werden, analog zur Evaluation der Embedding-Modelle, Korrelationen der Modellparameter untersucht, welche die eben getroffenen Annahmen

unterstützen. **Abbildung XX** illustriert diese Korrelationen, wobei sich eine negative Korrelation der Parameter Alpha und Beta zur resultierenden, gemittelten Testbewertung herauskristallisiert. Diese Korrelation unterstützt dabei die oben angeführte Annahme zur Topicverteilung in Dokumenten.

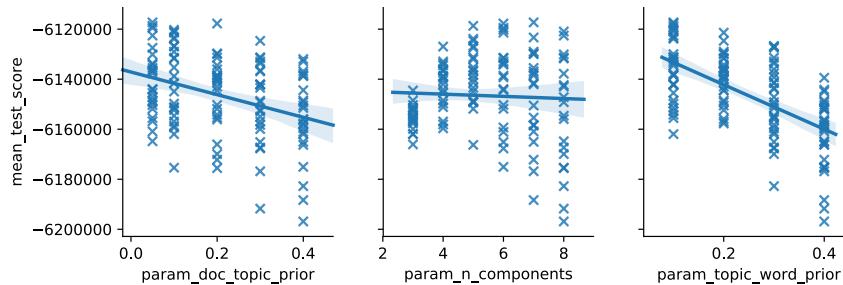


Abbildung 44: Parameterkorrelationen der LDA-Modelle mit $n > 3$ (Eigene Darstellung)

Weiterhin ist in obenstehender Abbildung zu erkennen, dass die beste Kombination an Parametern sieben Topics inkludiert. Zudem werden die Parameter Alpha und Beta mit 0.05 und 0.1 gewählt. Die resultierende Anzahl an Topics erlaubt es additional weitere Klassifikationsalgorithmen auf Basis der final identifizierten Verteilung an Topics in einem Dokument zu trainieren um eine Zuweisung des Dokumentes bzw. der Problemstellung zu einer Verfahrensklasse zu ermöglichen. An dieser Stelle sei auf das folgende Kapitel verwiesen, welches zur Deskription der Evaluation von Klassifikationsmodellen dient.

Aufgrund der ebenfalls möglichen Verwendung von LDA zur direkten Zuweisung von Verfahrensklassen, wird aufbauend auf diesem Ergebnis erneut die Methode des GridSearch benutzt um ein Modell zu trainieren, welches über drei Topics verfügt. Oppositionell zu vorheriger Operation, wird hier das gefilterte Datenset verwendet, da das Ziel darin besteht Topics zu erhalten, welche möglichst optimal das jeweilige semantische Konstrukt der Verfahrensklassen abbildet. Aufgrund des definitionsähnlichen Inhaltes dieses Datensatzes und der damit einhergehenden geringeren Menge an Rauschen

in den Daten werden bessere Ergebnisse erwartet¹². Im Training des Modells wird der Parameter Alpha mit den optionalen Werten 0.05, 0.1, 0.2, 0.3 und 0.4 sowie Beta mit 0.1, 0.2, 0.3 und 0.4 belegt, was Final in 100 differenten LDA-Modellen resultiert.

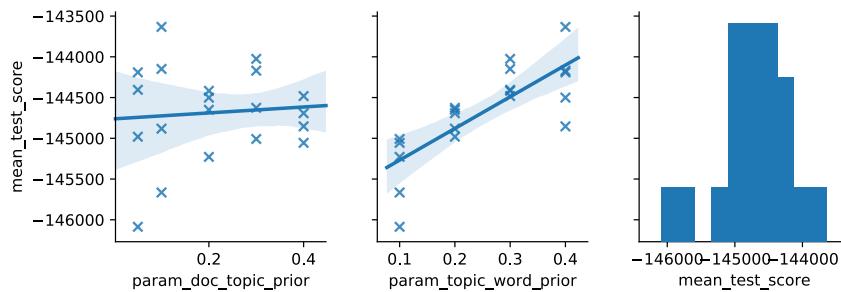


Abbildung 45: Parameterkorrelationen der LDA-Modelle mit n=3 (Eigene Darstellung)

Im Gegensatz zu den zuvor trainierten LDA-Modellen ist in den oben illustrierten Korrelationen zu erkennen, dass eine höhere Wahrscheinlichkeit der Zugehörigkeit eines Wortes zu mehreren Topics einen positiven Einfluss auf die Güte des Modelles hat. Final werden die Parameter Alpha und Beta mit den Werten 0.1 und 0.4 gewählt.

Neben der Auswahl der passenden Modelle werden diese hinsichtlich der Unterscheidbarkeit der Topics untersucht. Hierfür wird das Python-Paket *pyLDAvis* herangezogen, welches eine Visualisierung der Topics eines derartigen Modells ermöglicht.

Kommentiert [m46]: Add from mac

¹² Additional wird ebenfalls zur Unterstützung dieser Aussage ein LDA-Modell auf dem Abstract-Datensatz mit drei Topics trainiert. Dieses Modell erreicht allerdings im Rahmen einer indirekten Klassifikation lediglich eine Modellgüte von 35.55%.

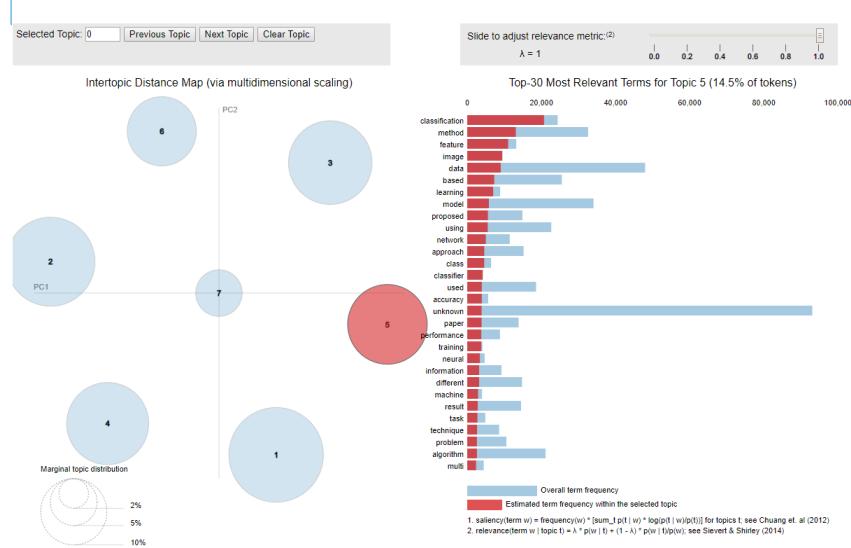


Abbildung 46: Visualisierung eines LDA-Modells (Eigene Darstellung)

Generell zeigt der linke Teil der Abbildung die Distanz zwischen den einzelnen Topics und dient somit zur Bewertung deren Unterscheidbarkeit, wohingegen die rechte Seite die Zugehörigkeit von Wörtern zu den Topics illustriert¹³. Additional gibt folgende Abbildung eine Übersicht über die Topics der beiden erstellten LDA-Modelle und zeigt hierfür die zehn wahrscheinlichsten Wörter pro extrahiertem Thema.

¹³ Da **Abbildung XX** eine Momentaufnahme einer interaktiven, HTML-basierten Darstellung ist, kann diese Zuordnung hier nur bedingt gezeigt werden. Die vollständige Visualisierung findet sich jedoch im elektronischen Anhang der Arbeit wieder.

LDA 3 Topics			LDA 7 Topics						
Topic 0 Clustering	Topic 1 Prediction	Topic 2 Pattern Mining	Topic 0 Classification	Topic 1 Medical	Topic 2 Geographical	Topic 3 Clustering	Topic 4 Medical Pattern	Topic 5 Prediction	Topic 6 Undefined
'clustering'	'classification'	'mining'	'classification'	'unknown'	'unknown'	'data'	'unknown'	'model'	'specie'
'data'	'regression'	'pattern'	'method'	'patient'	'pattern'	'clustering'	'sequence'	'prediction'	'analysis'
'cluster'	'prediction'	'rule'	'feature'	'study'	'soil'	'algorithm'	'gene'	'regression'	'sequence'
'analysis'	'model'	'association'	'image'	'risk'	'study'	'cluster'	'analysis'	'data'	'data'
'group'	'variable'	'sequential'	'data'	'clinical'	'concentration'	'method'	'protein'	'unknown'	'group'
'object'	'method'	'frequent'	'based'	'disease'	'sequential'	'based'	'cell'	'using'	'study'
'introduction'	'data'	'algorithm'	'learning'	'pattern'	'water'	'paper'	'sequencing'	'method'	'phylogenetic'
'technique'	'class'	'data'	'model'	'cancer'	'activity'	'proposed'	'expression'	'used'	'relationship'
'learning'	'feature'	'support'	'proposed'	'treatment'	'using'	'problem'	'genome'	'based'	'word'
'used'	'analysis'	'sequence'	'using'	'analysis'	'time'	'approach'	'region'	'study'	'classification'

Abbildung 47: Übersicht über die Schlüsselwörter pro Topic (Eigene Darstellung)

Wie hieran zu erkennen, inkludiert das zuletzt beschriebene Modell die relevanten Begriffe der einzelnen Verfahrensklassen, wohingegen diese im Modell mit sieben Topics streuen. Ebenfalls fällt auf, dass das umfassendere Modell themenspezifische Topics extrahiert und so beispielsweise geographische oder medizinische Begriffe in ein Topic fasst.

7.3 Evaluierung der Klassifikationsmodelle

Das folgende Kapitel dient der Evaluation der deskribierten Klassifikationsmodelle und der finalen Modellselektion, welche sich in der Implementation des Prototyps wiederfindet. Zunächst wird ein Überblick über multinominale Evaluationsmetriken gegeben, bevor auf die detaillierten Evaluationsergebnisse der einzelnen Modelle eingegangen wird.

7.3.1 Überblick multinomialer Evaluationsmetriken

Wie in **Kapitel XYZ** beschrieben bildet der Kern des Systems eine Textklassifikationsaufgabe, bei welcher differente Textdaten x_1, x_2, \dots, x_i zuvor definierten Klassen C_1, \dots, C_n zugeordnet werden. Abhängig von der zugrundeliegenden Datenbasis unterteilt sich die Evaluation dieser in binäre, multinomiale und hierarchische Metriken, um die Qualität des Modells zu bewerten. Im Zuge der binären Klassifikation werden die Eingabedaten in ausschließlich zwei sich nicht ineinander überlappende Klassen ($C_n, \neg C_n$) unterteilt,

wobei $n = 2$ gilt. Die resultierende Performance eines Klassifikators wird häufig in der Form einer binären Konfusionsmatrix dargestellt, aus welcher differente Evaluationsmetriken abgeleitet werden können. Im Falle der multinomiale Klassifikation wird genau ein Eingabedatenpunkt x_i einer Klasse C_n zugeordnet, wobei x_i jetzt mehr als zwei Klassen zugeordnet werden kann. Hierbei fundieren die Evaluationsmetriken auf einer multidimensionalen Konfusionsmatrix, welche bestimmte Evaluationsmetriken der binären Klassifikationsevaluation ausschließen (vgl. Sokolova & Lapalme, 2009, S. 427–429). Basierend auf einem One-vs-all-Verfahren können multinomiale auf binäre Klassifikationsprobleme heruntergebrochen werden. Dabei wird der Trainingsdatensatz in n verschiedene Teile geteilt und für jeden dieser ein eigener Klassifikator trainiert. Jeder der Klassifikatoren wird darauf trainiert eine Klasse von $n - 1$ Klassen zu unterscheiden (vgl. Mehra & Gupta, 2013, S. 573).

Im Rahmen des aktuellen Forschungsvorhabens würde dies allerdings den Trainingsaufwand verdreifachen, weshalb aus Komplexitätsgründen nur auf ausgewählte Evaluationsmetriken der multinomialen Klassifikation eingegangen wird. Hinzukommt, dass Evaluationsmetriken der hierarchischen Klassifikation ebenfalls vernachlässigt werden, da diese keinerlei Relevanz für die vorliegende Arbeit aufweisen. Die nachfolgende Tabelle illustriert die von SOKOLOVA & LAPALME (2009) zusammengetragenen Evaluationsmetriken der multinomialen Klassifikation.

Measure	Formula	Evaluation focus
Average Accuracy	$\frac{\sum_{i=1}^n \frac{tp_i + tn_i}{2n}}{n}$	The average per-class effectiveness of a classifier
Error Rate	$\frac{\sum_{i=1}^n \frac{fp_i + fn_i}{2n}}{n}$	The average per-class classification error
Precision _μ	$\frac{\sum_{i=1}^n \frac{tp_i}{tp_i + fp_i}}{n}$	Agreement of the data class labels with those of a classifiers if calculated from sums of per-text decisions
Recall _μ	$\frac{\sum_{i=1}^n \frac{tp_i}{tp_i + fn_i}}{n}$	Effectiveness of a classifier to identify class labels if calculated from sums of per-text decisions
Fscore _μ	$\frac{(F^2 + 1)Precision_{\mu}Recall_{\mu}}{F^2Precision_{\mu} + Recall_{\mu}}$	Relations between data's positive labels and those given by a classifier based on sums of per-text decisions
Precision _M	$\frac{\sum_{i=1}^n \frac{tp_i}{2n}}{n}$	An average per-class agreement of the data class labels with those of a classifiers
Recall _M	$\frac{\sum_{i=1}^n \frac{tp_i}{2n}}{n}$	An average per-class effectiveness of a classifier to identify class labels
Fscore _M	$\frac{(F^2 + 1)Precision_{\mu}Recall_{\mu}}{F^2Precision_{\mu} + Recall_{\mu}}$	Relations between data's positive labels and those given by a classifier based on a per-class average

Kommentiert [DH47]: Tabelle mit Formeln

Basierend auf einer nicht binären Konfusionsmatrix lassen sich Accuracy, Precision, Recall sowie der F-Score als Kombination beider Metriken aufzählen. Im Vergleich zu binären müssen jedoch multinomiale Klassifikatoren eine aggregierte Metrik über alle Klassen berechnen, um eine einheitliche Kennzahl zu erlangen. Grundsätzlich kann hierbei

zwischen Macro- und Microaveraging unterschieden werden. Erst genanntes berechnet einen einfach durchschnitt über alle Klassen. Das zweitgenannte Verfahren berechnet erst einen Durchschnittswert pro zugrundeliegende Klassen der Testdaten und aggregiert diese dann zur entsprechenden Metrik (vgl. Manning, Raghavan, & Schütze, 2008, S. 280). Die Macroaveraging-Verfahren beziehen dabei keine Klassenimbalance ein, weshalb im Rahmen der vorliegenden Arbeit alle multinomialen Evaluationsmetriken auf Basis des Microaveraging-Verfahrens kalkuliert werden. SOKOLOVA & LAPALME (2009) betonen, dass Precision und Recall die besten geeigneten Kennzahlen sind, um textbasierte Klassifikatoren zu evaluieren.

Unter dem Deckmantel des aktuellen Forschungsvorhabens ist Precision als die Kennzahl zu interpretieren, welche Ausschluss darüber gibt, wie viele der einer spezifischen Verfahrensklasse zugeordneten Dokumente, tatsächlich jener zugehörig sind. Wohingegen Recall den Anteil der tatsächlich richtig klassifizierten Dokumente, an der Menge aller Dokumente bemisst. Nachfolgend sei die Berechnung beider Verfahren auf Basis einer multinomialen Klassifikation dargestellt, wobei C die Menge aller zugrundeliegenden Klassen darstellt.

$$P = \frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FP_c} \quad (26)$$

$$R = \frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FN_c}$$

Als eine Kombination beider Metriken wird der F1-Score in der vorliegenden Untersuchung verwendet, welcher das harmonische Mittel beider Kennzahlen ausdrückt, um eine unikale Kennzahl des resultierenden Recommender-Systems zu erlangen.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Nichtsdestotrotz wird in der folgenden Ergebnisauswertung Precision und Recall vollständigkeitshalber mitaufgelistet. Hinzukommt, dass F1-Score zwar eine unikale Kennzahl zur Evaluierung des Systems darstellt, diese durch die Verschmelzung zweier Kennzahlen schwer zu interpretieren ist.

Neben dem F1-Score bietet sich noch die Cohens Kappa Statistik an, um multinomiale Klassifikationsprobleme zu bewerten. Dabei gilt die Metrik als ein Maß für die Intrarater-

Reliabilität, welche im Rahmen des Machine-Learning angibt, um wieviel besser ein multinomialer Klassifikator gegenüber puren Raten performt (vgl. Vieira, Kaymak, & Sousa, 2010, S. 924ff.). Berechnet wird die Metrik dabei wie folgt:

$$k = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)}$$

$\Pr(a)$ gibt dabei die Übereinstimmung des Klassifikators des Raters an und $\Pr(e)$ die zufällig erwartete Übereinstimmung des Raters. Dabei liegt die Evaluationsmetrik immer zwischen Null und Eins, wobei ein höherer Wert eine bessere Performance indiziert. Dabei sind Werte unter 0.4 als nicht annehmbar und somit als purer Zufall zu charakterisieren (vgl. Greve & Greve, 1997, S. 111).

Kommentiert [DH48]: Genauer

7.3.2 Ergebnisse der Evaluation von Klassifikationsmethoden (D)

Kommentiert [DH49]: Hier muss noch Klassifikationsergebnisse von:
1.LDA
2.Ensemble

Der nachfolgende Abschnitt gibt einen Überblick der Evaluationsergebnisse der trainierten Klassifikatoren. Dabei wurden alle Klassifikationsmodelle jeweils auf den in **Kapitel XYZ** beschriebenen Modellen der distribuierten Semantik trainiert sowie evaluiert. In Bezug auf den USE wird das Modell auf dem Abstract-Datensatz neu angeleert und somit der Aspekt des Transfer-Learnings mit in die Evaluation aufgenommen. Alle nachfolgenden Evaluationsmetriken basieren dabei auf den in **Kapitel XYZ** charakterisierten, manuell erstellten Validierungsdaten.

Hinzukommt, dass die folgende Evaluation zweigeteilt ist. Zum einen werden die Modelle auf den selektierten Daten und zum anderen auf den Augmentierten Daten trainiert. Die in **Kapitel XYZ** tangierte Klassenimbalance wird umgangen, in dem jeweils gleich viele Instanzen aus den entsprechenden Datensets entnommen werden. Hierbei werden die Instanzen zufällig aus dem zugrundeliegenden Datenset gewählt. Letztendlich reduzieren sich die in **Kapitel XYZ** deskribierten Datentöpfe somit auf 7.330 und im Falle der augmentierten Daten auf 19.383 Instanzen.

Hinzukommt, dass es für jede Modellklasse verschiedene Parameterkombinationen mithilfe des Grid-Search-Ansatzes getestet werden. Aus Übersichtlichkeitsgründen sind diese lediglich im Anhang dargestellt und werden an dieser Stelle nicht weiter beschrieben. Insgesamt wurden hierbei **10.950** differente Modellkombinationen trainiert, welche

Kommentiert [DH50]: Damit der sieht, dass wir ausgerastet sind. Alleine SVM/KNN sind auf den Datensätzen **10.950** Modelltrainings..

sich auf die verschiedene Modellklassen verteilen. An dieser Stelle sei jedoch zu betonen, dass weitere Parameterkombinationen durchaus in Betracht gezogen werden können, dieses jedoch die für die Arbeit zur Verfügung stehenden zeitlichen Rahmenbedingungen überschreiten würden.

Die nachfolgende **Tabelle XYZ** zeigt die verschiedenen Evaluationsmetriken der Modelle auf, welche auf dem selektierten Datensatz durchgeführt wurden. Klar zu erkennen ist dabei, dass neben der SVM eine Ensemble-Learning basierte Methodik die besten Evaluationsmetriken hinsichtlich des F1-Scores und der Cohen's-Kappa-Metrik liefern.¹⁴

Die schlechtesten Modelle bilden dabei, Model X und Model X, welche mit einem F1-Score von Zahl X 20% unter dem besten Modell liegen.

Tabelle 14:Evaluationsergebnisse auf nicht-augmentierten Datensatz

Model	FastText		DAN		USE	
	F ₁	k	F ₁	k	F ₁	k
SVM	0.733	0.60	0.85	0.775	0.667	0.50
KNN	0.783	0.675	0.817	0.725	0.60	0.40
LSTM (N-1)	0.767	0.650	0.73	0.60	-	-
GRU (N-1)	0.833	0.750	0.73	0.60	-	-
LSTM (1-1)	0.716	0.575	0.80	0.70	0.733	0.600
GRU (1-1)	0.783	0.675	0.783	0.675	-	-

F₁: F1-Score; k: Cohen's-Kappa-Score

Hinzukommt, dass sich DANs als beste Methode zu Errechnung distribuierter Repräsentationen herausstellen. Neben diesen empfehlen sich jedoch auch klassische FastText-Modelle, mit einem klassischen Mittelwertsverfahren als Kompositionsfunktion, als starkes Grundvorgehen. Wohingegen sich neu angelernte USE-Modelle als eher schlechte Modelle im Rahmen der Arbeit herauskristallisieren. Zwar erzielen diese Modelle State-of-the-Art-Ergebnisse in vielen NLP-Benchmarks, allerdings sind diese nicht auf themen-

¹⁴ Im elektronischen Anhang der Arbeit finden sich noch die Micro- sowie Macro-gemittelten Kennzahlen zu den Metriken: Recall, Precision und F1-Score wieder.

spezifischen Daten trainiert was einen Vergleich als diffizil gestaltet. **XXX** berichten ähnliche Ergebnisse auf domänen spezifisches Vokabular und unterstreichen somit das Ergebnis der vorangegangenen Untersuchung. Allerdings bleibt anzumerken, dass die in Kapitel XYZ deskribierten DAN-Modelle für weitere Forschungsvorhaben, als Transfer-Learning-Modell verwendet bzw. evaluiert werden können, falls es keinen zu großen themenspezifische Diskrepanzen gibt¹⁵.

Kommentiert [DH51]: Quelle raussuchen

Auffallend ist allerdings die schlechte Performance der DAN-Modelle bei Many-to-One-Architekturen. Dies kann dadurch begründet werden, dass die der Forschungsarbeit zugrundeliegenden DAN-Modelle lediglich auf Paragraphen trainiert sind und nicht auf einzelnen Wörtern oder N-Grammen dieser. Da Many-to-One-Architekturen im Rahmen des NLP grundsätzlich eine Sequenz von Wörtern verarbeiten, gestaltet sich eine Verwendung von DANs als problematisch. Allerdings empfiehlt sich ähnlich wie **XXX** diese Modelle neben reinen Paragraphen zusätzlich mit Wörtern sowie deren N-Grammen zu trainieren.

Kommentiert [DH52]: Quelle

In Anbetracht der durchschnittlichen Kennzahlen ist ersichtlich, dass Klassifikatoren welche auf DAN-Modellen basieren im Schnitt rund XX% (F1) und XX% (Cohen's Kappa) besser sind als FastText-Modelle und sogar XX% (F1) und XX% (Cohen's Kappa) besser als die von Google vortrainierten USE-Modelle. Ein Ähnliches Ergebnis zeichnet bei der Verwendung von SVMs und **XXX** ab, welche durchschnittlich die beste Performance auf Grundlage der selektierten Daten abliefern.

Die **Tabelle XYZ** illustriert, analog zu den nicht-augmentierten Daten, die Ergebnisse auf Basis der augmentierten Daten. Grundsätzlich sei auch hier zu betonen, dass auf DAN basierende Klassifikatoren bessere Ergebnisse erzielen als andere. Die besten Modelle bilden dabei **XXX** und **XXX** mit einer Cohen's-Kappa-Metrik von XX%. Das schlechteste Modell ist **XXX** mit XXX%. Ähnlich wie bei den nicht-augmentierten Daten platzieren sich FastText-Modelle vor dem USE-Modell, was ebenfalls auf die nicht-themenspezifischen Trainingsdaten des Modells zurückgeführt werden kann.

¹⁵ Die nötigen Tensorflow-Checkpoint-Files finden sich ebenfalls im elektronischen Anhang der Arbeit wieder.

Allerdings ist zu konstatieren, dass Modelle mit den augmentierten Daten leicht schlechtere Ergebnisse liefern als die Modelle mit nicht-augmentierten Daten. Dies unterstreichen ebenfalls ein Vergleich der besten Modelle beider Untersuchungen. So erreicht die **XXX mit XXX%** (F1) und **XXX%** (K) eine bessere Performance als das **XXX**-Modell der augmentierten Daten mit **XXX%**(F1) und **XXX%** (K). Zurückführen lässt sich dies, zum einen auf eine schlechtere Datenqualität der augmentierten Daten sowie zum anderen auf die vergleichbar einfachen applizierten Methoden.

Tabelle 15: Evaluationsergebnisse auf augmentierten Daten

Model	FastText		DAN		USE	
	F ₁	k	F ₁	k	F ₁	k
SVM	0.750	0.625	0.833	0.750	-	-
KNN	0.783	0.675	0.800	0.700	0.567	0.35
LSTM (N-1)	0.850	0.775	0.833	0.750	-	-
GRU (N-1)	0.833	0.750	0.800	0.700	-	-
LSTM (1-1)	0.750	0.625	0.800	0.700	0.767	0.650
GRU (1-1)	0.833	0.750	0.783	0.675	-	-

F₁: F1-Score; k: Cohen's-Kappa-Score

Zusammenfassend ist zu konstatieren, dass sich somit eine Nutzung einer Ensemble-Learning basierten Methodik empfiehlt, um gute Ergebnisse in der Zuordnung der Verfahrensklasse zu ermöglichen. Hinzukommt, dass eine Verwendung von DAN-Embedding bessere Ergebnisse erzielt als andere Modelle der distribuierten Semantik. Allerdings empfiehlt sich nicht die Nutzung der augmentierten Daten, da diese zu keinen besseren Evaluierungsergebnissen führen.

Kommentiert [DH53]: Ist noch gar nicht in der Tabelle – Wo ist beschrieben wie wir das Ensemble zusammensetzen?

Kommentiert [DH54R53]: In Nutzung KB wird die Zusammensetzung besprochen

8 Ergebnisse

Wie in **Kapitel XX** beschrieben besteht das Resultat eines Design-Science-Projektes aus einem nutzbaren Artefakt sowie gefundenen Design-Prinzipien, welche in weiterer Forschung und Entwicklung Beachtung finden können. Demnach dient das folgende Kapitel zur summierenden Darstellung der Ergebnisse sowie der eruierten Design-Prinzipien.

Zunächst ist hierbei auf das entwickelte Artefakt in Form des Prototyps eines textbasierten Recommender-Systems zur Unterstützung von fachfremden Personen durch die automatisierte Zuordnung von Problemstellungen zu spezifischen datenanalytischen Verfahrensklassen zu verweisen. Generell wird in der vorliegenden Untersuchung ein erster Prototyp eines solchen Systems entwickelt, welcher durch die Anwendung von Algorithmen aus den Bereichen des Machine Learning, Deep Learning, Topic Modeling sowie der Keyword Extraction eine Evaluation der übergebenen Problemstellung vornimmt. Hierbei erfolgt zunächst eine Vorverarbeitung der Trainingsdaten um darauf aufbauend eine Wissensbasis zu kreieren. Diese Wissensbasis enthält final differente Arten an Informationen bzgl. des Areals der Data Science. So umfasst diese Basis häufigkeitsbasierte Schlüsselwort-Informationen, strukturelle Informationen sowie eine Repräsentation des Areals im Vektorraum. Fortführend wird dieses Wissen herangezogen um unter der Vereinigung multipler Algorithmen eine finale Einordnung vorzunehmen. Insgesamt wird dieser Prozess mithilfe der Programmiersprache Python in ein webbasiertes Programm überführt. **Abbildung XX** zeigt die finale Startseits des entwickelten Systems.

Abbildung 48: Darstellung der Startseite des entwickelten Prototyps (Eigene Darstellung)

Im Rahmen dieser Startseite wird dem Nutzer ermöglicht eine Problemstellung einzutragen und diese dem System zu übergeben. Nach erfolgreicher Berechnung einer Verfahrensklasse wird dem Nutzer diese, eine personalisierte, exemplarische Analyse sowie die vorgeschlagene Datenstruktur ausgegeben (vgl. Abbildung XX). Weiterhin zeigen die Ergebnisse der Modellevaluierung, dass das Ziel eine bessere Einschätzung als das manuelle Raten hervorzubringen, mit einer Modellgüte des Ensemble-Modells von **XX** erreicht werden kann. Gleichwohl erhält der Nutzer eine kurze Beschreibung der Verfahrensklasse, damit dieser ein Verständnis für das Ergebnis aufbauen kann. Letztendlich kann demnach eine Unterstützung für fachfremde Personen erreicht werden, wobei zeitgleich die Kommunikation zwischen dieser und Datenanalysten simplifiziert werden kann, da die Information der Verfahrensklasse eine direkte, gedankliche Einordnung des Problems für den Datenanalysten ermöglicht.

Als Resultat der Konstruktion und Implementierung dieses Systems sind insgesamt **elf Design Prinzipien** abzuleiten. Die folgende Tabelle fasst diese zusammen.

Tabelle 16: Tabellarische Übersicht der identifizierten Design-Prinzipien

ID	Prinzip
P1	Empfehlung themenspezifische Embeddings zu trainieren und diese sowohl ex- als auch intrinsisch zu evaluieren.
P2	Erstellung von themenspezifischen Wort-Embeddings mithilfe der Architektur FastText.
P3	Hyperparameter beim Training von themenspezifischen Embeddings relevant.
P4	Entfernen von Punktierungen und Zahlen beim Training von Embeddings empfohlen.

Kommentiert [m55]: Bekommen wir hier vielleicht irgendwie den Footer dran oder so?
Oder ist das ok so? Ist halt unten abgeschnitten

Kommentiert [m56]: Ganz oben den Screen vom Programm. Wird auch in Anforderungsgegenüberstellung verwiesen. Ist das komisch soweit zurück zu verweisen?

Kommentiert [m57]: Do not forget!

- P5** Untersuchung der Verfahrensklassen im Vektorraum kann helfen die richtige Granularität der abgezielten Verfahrensklassen zu finden.
- P6** Überführung von Textdaten in Vektorraum erlaubt das automatisierte entfernen von **semantischen** Ausreißern.
- P7** Kombination von verschiedenen Keyword Extraktoren hilfreich.
- P8** Kosinusähnlichkeit zur Berechnung von Ähnlichkeiten im Vektorraum empfohlen.
- P9** Es ist möglich Topic-Modelle zur Klassifikation zu verwenden. Trainingsdaten sollten sehr hohe Qualität haben und möglichst nur die Zielklassen repräsentieren.
- P10** Klassifikation auf kurzen Paragraphen verspricht bessere Ergebnisse als auf längeren.

Wie dieser Tabelle zu entnehmen kann auf Basis dieser Untersuchung die Empfehlung ausgesprochen werden, themenspezifische Embeddings zu trainieren und nicht bereits bestehende, global trainierte Vektormodelle zu verwenden (P1). Wie in **Kapitel XX** dargestellt erreichen potentielle Klassifikationsalgorithmen durch Datenanalyse-Embeddings höhere Gütwerte und stärkere Generalisierungskraft im Rahmen der abgezielten Nutzung. Weiterhin ist zu konstatieren, dass bei geringen Datenmengen, aufgrund der unter **Kapitel XX** aufgezeigten Vorteile, auf die Architektur FastText zum Training von Wort-Embeddings verwiesen wird (P2). Zeitgleich zeigt die adäquate Wahl von Hyperparametern einen starken Einfluss auf die Qualität der resultierenden Embeddings und ist somit essentiell für die Passgenauigkeit der Repräsentation des Themenfeldes im Vektorraum (P3).

Tabelle 17: Minima und Maxima der Bewertung von Embedding-Modellen

Modell (Basis)	Valid linear		P		ME		r_{SP}	
	Min	Max	Min	Max	Min	Max	Min	Max
<i>FastText</i>	0.51	0.7	0.65	0.76	0.20	0.27	0.24	0.35
<i>DAN (fasttext_12)</i>	0.58	0.83	0.95	0.98	-	-	-	-
<i>DAN (fasttext_31)</i>	0.51	0.8	0.94	0.98	-	-	-	-
<i>DAN (fasttext_1)</i>	0.46	0.7	0.94	0.99	-	-	-	-

P: Purity; ME: Gemittelter Fehler der Ähnlichkeitsbewertung; r_{SP} : Rangkorrelation der Ähnlichkeitsbewertung

Kommentiert [m58]: Noch eintragen -> Sonntag alle DAN Modelle tauschen

Die trainierten FastText-Modelle zeigen beispielsweise Differenzen von ca. 20 Prozent in anschließenden Klassifikationsaufgaben und jeweils ca. sieben bis elf Prozent hinsichtlich differenter, intrinsischer Evaluationsmetriken. Ebenfalls zeigen sich in obiger Tabelle starke Unterschiede zwischen 24 und 29 Prozent hinsichtlich der Klassifikationen mithilfe unterschiedlicher DAN-Modelle, welche dieses Prinzip unterstützen. Additional zeigt sich die Relevanz der detaillierten Untersuchung des Vektorraums darin, dass dieser die Basis für viele weitere Aufgaben in der Durchführung der abgezielten Zuweisung darstellt.

Neben diesem Einfluss ist ebenfalls zu vermerken, dass die Entfernung von Punktierungen und Zahlen aus den Ausgangsdaten Vorteile hinsichtlich verschiedener Aufgaben mit sich bringt (P4). Dies ist vor allem bei denjenigen Aufgaben, die die Suche nach ähnlichen Wörtern im Vektorraum inkludieren von Relevanz, da durch diese Reinigung identische Wörter, die sich lediglich durch ein Satzzeichen unterscheiden entfernt werden (Beispielsweise die Wörter „clustering“ und „-clustering“). Gleichzeitig erhöht dies die Menge differenter Kontexte eines Wortes, was aufgrund der kontext-bezogenen Funktionsweise von Embedding Modellen ebenfalls Vorteile bringt.

In Anschluss an die Kreierung von adäquaten Embeddings ist ein weiteres Prinzip zu konstatieren. In der vorliegenden Untersuchung zeigt sich, dass die Exploration der Trainingsdaten im Vektorraum, durch Reduktion der Vektoren auf visualisierbare Dimensionen, eventuell notwendige Anpassungen in der Granularität der Zielklassen aufdecken kann (P5). So kann hierbei festgestellt werden, dass eine Zielklasse zwei stark konzentrierte Areale im Vektorraum aufweist und jene demnach weiter unterteilt werden sollte. Zeitgleich erlaubt es diese Untersuchung Ausreißer in den Ausgangsdaten zu entfernen. Dieses Vorgehen wird in vorliegender Arbeit empfohlen, da die Bereinigung von Ausreißern im Vektorraum ermöglicht diese aus einer semantischen Perspektive zu bewerten und als semantische Ausreißer zu identifizieren (P6).

Im weiteren Aufbau einer Wissensbasis ist weiterhin zu konstatieren, dass die Kombination differenter Keyword-Extraction-Methoden empfehlenswert ist (P7). Wie in **Kapitel XX** illustriert, können durch die Applikation mehrerer Algorithmen und deren Vereinigung bessere Ergebnisse hinsichtlich der Kohärenz der resultierenden Topics erreicht werden. Neben der besseren Evaluationsergebnisse stellt dies ebenfalls eine höhere wortbezogene Abdeckung der Topics sicher, da die Algorithmen durch unterschiedliche Bewertungsfunktionen unterschiedliche Wörter extrahieren.

Hinsichtlich der finalen Zuweisung einer Problemstellung im Rahmen des aktuellen Forschungsvorhabens ist festzustellen, dass das Maß der Kosinusähnlichkeit, aufgrund dessen Bewertung der Richtung von Vektoren und nicht deren Länge, eine gute Eignung zur Berechnung von Ähnlichkeiten zwischen Wörtern oder Konstrukten im Vektorraum aufweist. Generell wird demnach empfohlen dieses Maß bei der Verwendung von Embeddings zu benutzen (P8). Weiterhin zeigt sich bei Verwendung des LDA-Algorithmus, wie bereits unter **Kapitel XX** dargelegt, dass Modelle des Topic-Modeling ebenfalls zur indirekten Klassifikation von Problemstellungen herangezogen werden können (P9). Hierbei muss allerdings sichergestellt werden, dass die Ausgangsdaten, auf welchen das Modell trainiert wird eine hohe Qualität hinsichtlich der Menge an Rauschen sowie der Repräsentanz semantischer Konstrukte aufweisen müssen. So sollten die Daten für diese Art der Anwendung lediglich die verfolgten Zielklassen repräsentieren.

Abschließend ist festzuhalten, dass die Klassifikation einer Problemstellung im Allgemeinen auf kürzeren Texten bessere Ergebnisse verspricht. Dies resultiert aus differenten Charakteristiken der angewendeten Methoden.

So besteht beispielsweise ein Problem in der Verarbeitung langer Sequenzen im Rahmen von RNN-basierten Netzwerken (vgl. **Kapitel XX**)

Vielleicht dass Ensemble benutzt werden soll? -> Mal sehen ob es uns viel bringt... Hoffen wir es

Fallen uns noch Prinzipien zur Programmierung ein?



9 Kritische Würdigung (D)

Das grundlegende Ziel der Arbeit ist es ein Recommender-System zur Selektion von Machine-Learning bzw. Data-Mining basierte Verfahrensklassen zu entwickeln, um so mit eine Unterstützung für fachfremde Personen der Data Science zu ermöglichen. Dabei beschäftigt sich der erste Teil der Arbeit mit einer initialen Datenbeschaffung, -Selektion und -Augmentation der Daten, um anschließend geeignete Methoden zur Erfüllung der Zielstellung aufzuzeigen sowie zu evaluieren. Abschließend ist ein Vorgehen zur Implementierung des Prototypens aufgezeigt, sowie eine umfassende Softwarearchitektur, welche auf der Implementation dessen beruht dargelegt. Im folgenden Kapitel erfolgt eine kritische Würdigung, des in der Arbeit dargelegten Vorhabens, um anschließend, im Zuge des Fazits, die in **Kapitel XYZ** deskribierten Forschungsfragen zu beantworten.

Zunächst kann herausgestellt werden, dass durch den in **Kapitel XYZ** beschriebenen Datenbeschaffungsprozess, man eine hohe Anzahl an Trainingsdaten beschaffen kann, jedoch der Aufbau einer Validierungsbasis damit nicht gewährleistet werden kann. Zwar beinhalten die Trainingsdaten semantisch relevante Konstrukte, der zu klassifizierenden Verfahrensklassen, allerdings liegen diese nicht in der Form von Problembeschreibungen vor, sondern in Form von wissenschaftlichen Artikeln. Die damit einhergehende Problematik, diese Divergenzen zu umgehen, wurde ausführlich **Kapitel XYZ** dargelegt. Nichtsdestotrotz bleibt zu konstatieren, dass die Diskrepanz an der Trainingsdaten- und Validierungsdatenmenge als sehr hoch festzulegen ist. Somit gilt es als diffizil, allgemeingültige Aussagen über die Generalisierungsfähigkeiten der Modelle zu treffen. Hinzukommt, dass mit zunehmender Menge an potentiellen Validierungsdaten, die dargelegten Evaluationsergebnisse stark von denen in der Arbeit dargelegten differieren können.

Im Zuge der distribuierten Repräsentationen ist anzumerken, dass die im Rahmen der Arbeit trainierten Modelle jeweils Vektoren mit 100 Dimensionen errechnen¹⁶. Im Zuge weiterer Forschungsvorhaben empfiehlt es sich Modelle der distribuierten Semantik mit mehr als den gegebenen Dimensionen zu trainieren. Die zeitlichen sowie rechentechnischen Restriktion des aktuellen Forschungsvorhabens begründen den Verzicht, zusätzliche Dimensionalitätsaspekte mit in das Training einzubeziehen.

Nichtsdestotrotz haben sich distribuierte Paragraphrepräsentation als die beste Operationalisierung, in Bezug auf Textdaten, herausgestellt. Im Verlauf der Arbeit finden sich häufig, mit PCA reduzierte Diagramme wieder. Allerdings wurde das PCA-Verfahren im Zuge der Arbeit hauptsächlich wegen seines deterministischen Charakters gewählt sowie der geringen Rechenintensität im Vergleich zu anderen Verfahren ([Quelle](#)). An dieser Stelle sein Verfahren wie T-SNE oder **XXX** genannt, welche ebenfalls für die Dimensionalitätsreduktionen verwendet werden können.

Kommentiert [DH59]: References

Hinzukommt, dass bei der Evaluation der Topic-Modelle, externe Kennzahlen deskribiert sind, welche allerdings aus rechenintensiven Zeitgründen nicht angewandt wurden. So bietet sich die *Related-Article-Concept-Overlap-Kennzahl* sehr gut für die Evaluations dieser an, da diese Wikipedia als Quelle semantische Konstrukte nutzt, was die wahrscheinlich größte Quelle menschlich generierter semantischer Konstrukte darstellt. Allerdings ist die Abfrage dessen mit einem exorbitanten Zeitaufwand verbunden, weshalb auch hier auf weiterführende Forschungsvorhaben verwiesen sei.

In Anbetracht der eigentlichen Klassifikation haben sich, im Rahmen des aktuellen Forschungsvorhabens, Ensemble-Learning basierte Klassifikatoren als die beste Methodik für die Implementierung eines Recommender-Systems herausgestellt. Allerdings ist anzumerken, dass das Gebiet des Ensemble-Learning weitaus größer ist als in der Arbeit deskribiert. Hinzukommt, dass das im Rahmen der Arbeit verwendete Weighted-Averaging-Verfahren vergleichsweise simple, zu erweiterten Boosting- oder Bagging-Strategien ist, welche i.d.R. State-of-the-Art-Performance in Hinblick auf Klassifikationen versprechen (vgl. Sagi & Rokach, 2018, S. 1249).

¹⁶ Die einzige Ausnahme bildet hierbei das vortrainierte Universal-Sentence-Encoder-Modell von Google. Allerdings hat dies, im Rahmen des Forschungsvorhabens, vergleichsweise schlechte Ergebnisse erzielt, weswegen es an dieser Stelle vernachlässigt wird.

Neben der Zuordnung einer Verfahrensklasse zu einer gegebenen natürlichsprachlichen Problemstellung, illustriert die vorliegende Arbeit einen regelbasierten Ansatz der Named-Entity-Recognition, um gegebene Datenentitäten aus dem Text zu extrahieren. Ähnlich wie bei dem eben beschriebenen Sachverhalt, ist das Gebiet der NER als viel komplexer und umfassender anzusehen, als in der Arbeit dargelegt. Beispielsweise listen GOYAL ET AL. (2018) insgesamt 48 verschiedene lernbasierte, regelbasierte sowie hybride NER-Ansätze auf, was die Komplexität dieses Forschungsfeldes unterstreicht.

Aufbauend auf dem aktuellen Forschungsvorhaben sind an dieser Stelle hybride Systeme zu propagieren, welche den deskribierten regelbasierten Ansatz (vgl. **Kapitel XYZ**) in ihr Vorgehen inkludieren können. Zu betonen ist jedoch, dass im Falle lernbasierter NER-Strategien, zusätzliche annotierte Daten beschaffen werden müssen, welche positive so wie negative Entitäten aus dem Text hervorheben.

10 Fazit und Ausblick (D, R)

Im nachfolgenden Abschnitt werden die zentralen Erkenntnisse unter der Berücksichtigung der deskribierten Forschungsfragen zusammengefasst. Neben der Beantwortung der Forschungsfragen schließt außerdem ein Ausblick für weitere Forschungsarbeiten an.

Forschungsfrage I: (D)

Wie lässt sich eine natürlichsprachliche Datenbasis erstellen und expandieren, um die Implementierung des fokussierten Prototyps zu ermöglichen?

Im Rahmen der vorliegenden Forschungsarbeit haben sich wissenschaftlichen Datenbanken zur **Extraktion** von Trainingsdaten etabliert. Hierbei sind der Science-Direct-Datenbankverbund sowie die Arxiv-Datenbank mit klar definierten und voneinander abgegrenzten Suchanfragen (vgl. **Anhang XYZ**) durchsucht worden. Daraus ergeben sich insgesamt 90.000 initiale Textdokumente, welche nachfolgend von Rauschen bereinigt werden. Zunächst empfiehlt sich eine rein regelbasierte Reinigung der Textdaten mithilfe regulärer Ausdrücke, um syntaktisch irrelevante Textbausteine sowie Dokumente, welche ein gewisses Level an Rauschen überschreiten zu entfernen. Basierend auf der distributionellen Hypothese, sind anschließend semantische Ausreißer mithilfe eins dichten basierten Noise-Clustering-Verfahrens entfernt worden, um semantisch unrelevante Textdaten zu entfernen. Schlussfolgernd stehen somit rund 11.000 bereinigte Dokumente zu Verfügung, welche für die Expansion des Datensets genutzt werden.

Die **Expansion** des extrahierten Datensatzes strebt eine höhere Variabilität der zugrundeliegenden Daten an sowie eine Erweiterung der Trainingsdaten. Die Ambiguität der natürlichen Sprache begründet dabei verschiedene Methoden, welche für die Augmentation angewandt werden können. Hierbei bieten sich die Möglichkeiten der Substitution, der Reihenfolgenanpassung, dem automatischen Generieren neuer Textdaten oder auch Verfahren der Machine-Translation an.

Im Rahmen des aktuellen Forschungsvorhabens sind jedoch einfache Methoden zur Expansion des Datenset appliziert worden. Allerdings hat sich bei der Evaluation der Klassifikationsmodelle gezeigt, dass die augmentierten Daten zu leicht schlechteren Ergebnissen wie die bereinigten Daten. Schlussfolgernd sind in sich anschließenden Forschungsvorhaben komplexere Methoden, wie beispielsweise das Verwenden von Encoder-Decoder-Modellen (vgl. **Kapitel XYZ**) zu propagieren. Allerdings konnten diese Verfahren wegen ihrer Komplexität, Rechenintensität sowie zu wenigen Daten keine Anwendung im aktuellen Forschungsvorhaben finden.

Forschungsfrage II: (D)

Welche Anforderungen existieren an ein textbasiertes Recommender-System und wie lässt sich ein Prototyp eines solchen Systems entwickeln?

In Anbetracht der aktuellen Zielstellung sowie den gegebenen Restriktionen der Arbeit, sind in **Kapitel XYZ** die Anforderungen an den zu erstellenden Prototypen aufgelistet. Dabei sind diese in funktionale und nicht-funktionale unterteilt und in essentielle, konditionale und optionale Anforderungen priorisiert. Vor dem Hintergrund des deskribierten Gestaltungsziel der Forschungsarbeit, bildet die wichtigste Anforderung die Zuordnung der eingegebenen Problembeschreibung zu einer Verfahrensklasse, da diese die Kernfunktionalität des Systems beschreibt. Eine weitere daraus resultierende Anforderung ist, dass das System in seiner Zuordnung zuverlässiger als reines Raten funktionieren soll, um eine adäquate Anwendung der Software zu garantieren. Dazu wird die Cohens-Kappa-Metrik verwendet, um diese Anforderung mit einer unikalen Kennzahl zu evaluieren.

Hinzukommt, dass dem Nutzer neben der Empfehlung der Data-Science-Verfahrensklasse, zusätzlich eine personalisierte exemplarische Analyse gegeben wird, um den Anforderungen an ein **Recommender-System** gerecht zu werden.

Neben den funktionalen Anforderungen existiert die konditionale Anforderung, der Extraktion von Analyseentitäten. Zwar würde ein Fehlen dieser Anforderung dem zugrundeliegenden Gestaltungsziel nicht entgegenwirken, jedoch die Kommunikation zwischen Data-Science-Experten und fachfremder Person erleichtern. Das Ziel dieser Anforderung ist es, dem Data-Science-Experten einen ersten Überblick, über die in der Problembeschreibung deskribierten Datenattribute bereitzustellen.

Ebenfalls ist zu konstatieren, dass die in **Kapitel XYZ** beschriebene linguistische Herausforderung, die Zuordnung zu einer Verfahrensklasse erschwert. Aus diesem Grund soll der Prototyp in der Lage sein, syntaktische Divergenzen zu umgehen. Zum einen kann dies erreicht werden durch die in den Prototypen verwendeten Methoden. Zum anderen kann neben einer ausführlichen Problembeschreibung, eine Kurzbeschreibung ebenfalls verarbeitet werden, welche die Kernproblematik stärker hervorhebt und weniger linguistische Diskrepanzen beinhaltet.

Bezüglich der zweiten Forschungsfrage empfiehlt sich ein agiles Vorgehen, um eine anforderungsgerechte Prototypenentwicklung zu ermöglichen. Auf dem zugrundeliegenden

Kommentiert [DH60]: Irgendwie BlaBla. Warum genau brauchen wir das?

Design-Science-Prozess empfiehlt sich die Verwendung des Prototyping-Modells nach PRESSMAN (2015). Hierbei können nicht-funktionale Anforderungen, wie die Einfachheit oder die Usability des Systems, intern durch die Phase des Quick-Design evaluiert werden und iterativ novelliert werden.

Additional basiert das durch PRESSMAN (2015) vorgeschlagene Modell auf einer ständigen Rückkopplung, was schnell sichtbare Ergebnisse in der Prototypentwicklung ermöglicht sowie eine interne Evaluation und Anpassung gegebener Anforderungen ermöglicht.

Neben dem Vorgehen nach Pressmann empfiehlt sich des Weiteren einen starken konzeptionellen Entwurf zu gestalten, welcher sich an den Prinzipien der OOP anlehnt, um Erweiterungen oder Änderungen schnell in die Software zu implementieren.

Forschungsfrage III:

Wie lassen sich die im Rahmen der Arbeit identifizierten Methoden sowie das zu implementierende Gesamtsystem vor dem Hintergrund der aktuellen Zielstellung evaluieren?

Im Rahmen der vorliegenden Arbeit werden unterschiedliche Methodengruppen sowie Algorithmen identifiziert und appliziert. Die Evaluation dieser sowie des Gesamtsystems teilt sich daher in die verschiedenen Kategorien und orientiert sich an Methoden aus der jeweiligen Fachliteratur. Hinsichtlich der distribuierten Repräsentationen wird zwischen einer extrinsischen und intrinsischen Bewertung unterschieden. Im Zuge der ersten Evaluation werden die insgesamt 516 trainierten Modelle in klassischen Textklassifikationsaufgaben als Inputdaten verwendet um die Güte dieser Klassifikation als Entscheidungskriterium heranzuziehen. Hierbei wird sich einer SVM als Classifier bedient, welche im Sinne der Vergleichbarkeit, in ihren Hyperparametern konstant gehalten wird. Die intrinsische Evaluation hingegen wird mittels ähnlichkeitsbasierter Analysen und der Reinheit der im Vektorraum geclusterten Klassen durchgeführt. Hinsichtlich der Evaluation von Topic Modellen ist festzustellen, dass die Bewertung der Passgenauigkeit eines Topics auf ein semantisches Konstrukt im Allgemeinen diffizil ist. So werden für Methoden der Keyword Extraction unterschiedliche Kohärenzkennzahlen verwendet, welche die Zusammengehörigkeit der Wörter in einem Topic bemisst. Weiterhin wird die Methode des GridSearch angewendet um das beste LDA-Modell anhand der Kennzahl der Log-likelihood zu identifizieren. Ferner erfolgt die Evaluation von Klassifikationsmodellen

sowie des Gesamtsystems, das aufgrund der Anwendung des Ensemble Learnings ebenfalls durch Klassifikationsmetriken evaluiert wird. Aufgrund des Tatbestandes einer multinomialen Klassifikation wird hier das Verfahren des Micro-Averaging hinsichtlich der Kennzahlen Precision, Recall sowie F1-Score verwendet. Additional wird die Kennzahl Cohens-Kappa hinzugezogen, um zu bewerten, wie viel besser die Modelle gegenüber dem puren Raten agieren. Darüber hinaus werden für alle Methodengruppen Korrelationen zwischen den Modellparametern und den Testergebnissen analysiert, um die Sinnhaftigkeit potentieller Anpassungen zu bewerten.

Ausblick

- ➔ Arbeit gibt Analupunkt für weitere Forschung bezüglich:
 - Embeddings
 - Topic-Modelling
 - Keyword-Extraction
 - Implementierung eines neuen Klassifikators
- ➔ Dabei dient die entwickelte Systemarchitektur als Schnittstelle und ermöglicht eine leichte Anbindung weiterer Funktionalitäten

Ausblick

- # Interessanter Ansatz Übersetzungsmodelle zu schaffen
- # Multilingualer Korpus notwendig -> Sehr schwer zu bekommen

Kombination von diesem System mit IDAs bieten gute Vorteile

- # Zunächst auswahl von Verfahrensklasse und dann auswahl der Methode in der Verfahrensklasse

DAN auf ngrams

- # Jeden Satz einzeln klassifizieren und irgendwie zusammenführen

Literaturverzeichnis

- Aggarwal, C. C. (2014). An Introduction to Frequent Pattern Mining. In C. C. Aggarwal & J. Han (Hrsg.), *Frequent Pattern Mining*. Springer International Publishing.
- Amato, F., Moscato, V., Picariello, A., & Piccialli, F. (2019). SOS: A multimedia recommender System for Online Social networks. *Future Generation Computer Systems*, 93, 914–923. <https://doi.org/10.1016/j.future.2017.04.028>
- Bag, S., Ghadge, A., & Tiwari, M. K. (2019). An integrated recommender system for improved accuracy and aggregate diversity. *Computers & Industrial Engineering*, 130, 187–197. <https://doi.org/10.1016/j.cie.2019.02.028>
- Bagali, S., & Waggenspack, W. N., Jr. (1995). A Shortest Path Approach to Wireframe to Solid Model Conversion. In *Proceedings of the Third ACM Symposium on Solid Modeling and Applications* (S. 339–350). New York, NY, USA: ACM. <https://doi.org/10.1145/218013.218083>
- Bakarov, A. (2018). A Survey of Word Embeddings Evaluation Methods. *CoRR, abs/1801.09536*. Abgerufen von <http://arxiv.org/abs/1801.09536>
- Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, Collaborative Recommendation. *Communications of the ACM*, 40(3), 66–72. <https://doi.org/10.1145/245108.245124>
- Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1, 238–247. <https://doi.org/10.3115/v1/P14-1023>
- Bayer, J., Wierstra, D., Togelius, J., & Schmidhuber, J. (2009). Evolving Memory Cell Structures for Sequence Learning. In C. Alippi, M. Polycarpou, C. Panayiotou, & G. El-llinas (Hrsg.), *Artificial Neural Networks – ICANN 2009* (S. 755–764). Springer Berlin Heidelberg.

Becker, J., Holten, R., Knackstedt, R., & Niehaves, B. (2003). *Forschungsmethodische Positionierung in der Wirtschaftsinformatik: epistemologische, ontologische und linguistische Leitfragen*. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Westfälische Wilhelms-Universität Münster. Abgerufen von <http://www.econstor.eu/handle/10419/59562>

Becker, J., Niehaves, B., & Knackstedt, R. (2004). Bezugsrahmen zur epistemologischen Positionierung der Referenzmodellierung. In J. Becker & P. Delfmann (Hrsg.), *Referenzmodellierung* (S. 1–17). Heidelberg: Physica-Verlag HD. Abgerufen von http://www.springerlink.com/index/10.1007/978-3-7908-2698-2_1

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-parameter Optimization. *J. Mach. Learn. Res.*, 13, 281–305.

Bhukya, D. P., & Ramachandram, S. (o. J.). Performance Evaluation of Partition Based Clustering Algorithms in Grid Environment Using Design of Expirements. *International Journal of Reviews in Computing*, 4(4), 46–53.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Biswas, S. K., Bordoloi, M., & Shreya, J. (2018). A graph based keyword extraction model using collective node weight. *Expert Systems with Applications*, 97, 51–59. <https://doi.org/10.1016/j.eswa.2017.12.025>

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3, 993–1022.

B.Meshram, S., & M. Shinde, S. (2015). A Survey on Ensemble Methods for High Dimensional Data Classification in Biomedicine Field. *International Journal of Computer Applications*, 111(11), 5–7. <https://doi.org/10.5120/19580-1162>

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *CoRR*, *abs/1607.04606*. Abgerufen von <http://arxiv.org/abs/1607.04606>

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5, 135–146.

- Bougouin, A., Boudin, F., & Daille, B. (2013). TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing* (S. 543–551). Nagoya, Japan: Asian Federation of Natural Language Processing. Abgerufen von <http://www.aclweb.org/anthology/I13-1062>
- Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., & Bengio, S. (2016). Generating Sentences from a Continuous Space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning* (S. 10–21). Berlin, Germany: Association for Computational Linguistics. <https://doi.org/10.18653/v1/K16-1002>
- Brazdil, P., Carrier, C. G., Soares, C., & Vilalta, R. (2009). *Metalearning: Applications to Data Mining*. Berlin Heidelberg: Springer-Verlag. Abgerufen von <https://www.springer.com/de/book/9783540732624>
- Camacho-Collados, José, & Pilehvar, M. T. (2018). From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. *Journal of Artificial Intelligence Research*, 63, 743–788.
- Camacho-Collados, Jose, & Pilehvar, M. T. (2018). On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (S. 40–46). Brussels, Belgium: Association for Computational Linguistics. Abgerufen von <http://www.aclweb.org/anthology/W18-5406>
- Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., & Jatowt, A. (2018). A Text Feature Based Automatic Keyword Extraction Method for Single Documents. In G. Pasi, B. Piwowarski, L. Azzopardi, & A. Hanbury (Hrsg.), *Advances in Information Retrieval* (S. 684–691). Springer International Publishing.
- Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., ... Kurzweil, R. (2018a). Universal Sentence Encoder. *arXiv:1803.11175 [cs]*. Abgerufen von <http://arxiv.org/abs/1803.11175>
- Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., ... Kurzweil, R. (2018b). Universal Sentence Encoder. *arXiv:1803.11175 [cs]*. Abgerufen von <http://arxiv.org/abs/1803.11175>

- Chen, K., Zhang, Z., Long, J., & Zhang, H. (2016). Turning from TF-IDF to TF-IGM for term weighting in text classification. *Expert Systems with Applications*, 66, 245–260. <https://doi.org/10.1016/j.eswa.2016.09.009>
- Cho, K., van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (S. 103–111). Doha, Qatar: Association for Computational Linguistics. Abgerufen von <http://www.aclweb.org/anthology/W14-4012>
- Chung, J., Gülcühre, Ç., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv e-prints*, *abs/1412.3555*. Abgerufen von <https://arxiv.org/abs/1412.3555>
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.*, 12, 2493–2537.
- Danubianu, M., & Socaciu, T. (2011). Efficient Selection of Data Mining Method. *ANUL XVIII*, 3, 101–108.
- Dave, R. N. (1991). Characterization and detection of noise in clustering. *Pattern Recognition Letters*, 12(11), 657–664. [https://doi.org/10.1016/0167-8655\(91\)90002-4](https://doi.org/10.1016/0167-8655(91)90002-4)
- Dhenakaran, S. S., & Thirugnana, K. S. (2011). WEB CRAWLER - AN OVERVIEW. *International Journal of Computer Science and Communication*, 2(1), 265–267.
- Domhan, T., & Hieber, F. (2017). Using Target-side Monolingual Data for Neural Machine Translation through Multi-task Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (S. 1500–1505). Copenhagen, Denmark: Association for Computational Linguistics. <https://doi.org/10.18653/v1/D17-1158>
- Eftimov, T., Seljak, B. K., & Korošec, P. (2017). A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations. *PLoS ONE*, 12(6). <https://doi.org/10.1371/journal.pone.0179488>
- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2), 179–211.
- Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial

Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (S. 226–231). Portland, Oregon: AAAI Press. Abgerufen von <http://dl.acm.org/citation.cfm?id=3001460.3001507>

Faber, P. (2012). *A Cognitive Linguistics View of Terminology and Specialized Language*. De Gruyter. Abgerufen von <https://books.google.com.au/books?id=U2Gu15sLIKwC>

Fadaee, M., Bisazza, A., & Monz, C. (2017). Data Augmentation for Low-Resource Neural Machine Translation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 567–573. <https://doi.org/10.18653/v1/P17-2090>

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3), 37–54.

Fettke, P. (2006). State-of-the-Art des State-of-the-Art - Eine Untersuchung der Forschungsmethode „Review“ innerhalb der Wirtschaftsinformatik. *WIRTSCHAFTSINFORMATIK*, 48(4), 257–266.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., & Ruppin, E. (2001). Placing Search in Context: The Concept Revisited. In *Proceedings of the 10th International Conference on World Wide Web* (S. 406–414). New York, NY, USA: ACM. <https://doi.org/10.1145/371920.372094>

Florescu, C., & Caragea, C. (2017). PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (S. 1105–1115). Vancouver, Canada: Association for Computational Linguistics. <https://doi.org/10.18653/v1/P17-1102>

Fowler, M. (2018). *Refactoring: improving the design of existing code* / (2nd edition). New Jersey: Pearson Education. Abgerufen von [http://slubdd.de/katalog?TN_li-bero_mab22\)500288897](http://slubdd.de/katalog?TN_li-bero_mab22)500288897)

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2015). *Design Patterns: Entwurfsmuster als Elemente wiederverwendbarer objektorientierter Software*. mitp/bhv.

Gers, F. A., Schmidhuber, J., & Cummins, F. A. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12, 2451–2471. <https://doi.org/10.1162/089976600300015015>

- Giraud-Carrier, C. (2005). The Data Mining Advisor: Meta-learning at the Service of Practitioners. In *Proceedings of the Fourth International Conference on Machine Learning and Applications* (S. 113–119). Washington, DC, USA: IEEE Computer Society. <https://doi.org/10.1109/ICMLA.2005.65>
- Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. *J. Artif. Int. Res.*, 57(1), 345–420.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. The MIT Press.
- Goyal, A., Gupta, V., & Kumar, M. (2018). Recent Named Entity Recognition and Classification techniques: A systematic review. *Computer Science Review*, 29, 21–43. <https://doi.org/10.1016/j.cosrev.2018.06.001>
- Graner, N., Sharma, S., Sleeman, D., Rissakis, M., Craw, S., & Moore, C. (1993). THE MACHINE LEARNING TOOLBOX CONSULTANT. *International Journal on Artificial Intelligence Tools*, 02(03), 307–328. <https://doi.org/10.1142/S0218213093000163>
- Gregor, S., & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. *MIS Q.*, 37(2), 337–356. <https://doi.org/10.25300/MISQ/2013/37.2.01>
- Greve, W., & Greve, W. (1997). *Wissenschaftliche Beobachtung: eine Einführung*. Weinheim.
- Gridach, M. (2017). Character-level neural network for biomedical named entity recognition. *Journal of Biomedical Informatics*, 70, 85–91. <https://doi.org/10.1016/j.jbi.2017.05.002>
- Gruslys, A., Munos, R., Danihelka, I., Lanctot, M., & Graves, A. (2016). Memory-efficient Backpropagation Through Time. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (S. 4132–4140). USA: Curran Associates Inc. Abgerufen von <http://dl.acm.org/citation.cfm?id=3157382.3157559>
- Habibi, M., & Popescu-Belis, A. (2015). Keyword Extraction and Clustering for Document Recommendation in Conversations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(4), 746–759. <https://doi.org/10.1109/TASLP.2015.2405482>

- Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On Clustering Validation Techniques. *Journal of Intelligent Information Systems*, 17(2), 107–145. <https://doi.org/10.1023/A:1012801612483>
- Hasan, K. S., & Ng, V. (2014). Automatic Keyphrase Extraction: A Survey of the State of the Art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (S. 1262–1273). Baltimore, Maryland: Association for Computational Linguistics. Abgerufen von <http://www.aclweb.org/anthology/P14-1119>
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75–105.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Huang, F., & Yates, A. (2009). Distributional Representations for Handling Sparsity in Supervised Sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1* (S. 495–503). Stroudsburg, PA, USA: Association for Computational Linguistics. Abgerufen von <http://dl.acm.org/citation.cfm?id=1687878.1687948>
- Hulth, A. (2003). Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing* (S. 216–223). Stroudsburg, PA, USA: Association for Computational Linguistics. <https://doi.org/10.3115/1119355.1119383>
- IEEE Std 830-1998 (Hrsg.). (1998). *IEEE recommended practice for software requirements specifications*. New York, NY: IEEE.
- Iyyer, M., Manjunatha, V., Boyd-Graber, J., & Daumé III, H. (2015a). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (S. 1681–1691). Beijing, China: Association for Computational Linguistics. Abgerufen von <http://www.aclweb.org/anthology/P15-1162>

- Iyyer, M., Manjunatha, V., Boyd-Graber, J., & Daumé III, H. (2015b). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (S. 1681–1691). Beijing, China: Association for Computational Linguistics. Abgerufen von <http://www.aclweb.org/anthology/P15-1162>
- Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (S. 427–431). Valencia, Spain: Association for Computational Linguistics. Abgerufen von <http://www.aclweb.org/anthology/E17-2068>
- Jurafsky, D., & Martin, J. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (Bd. 2).
- Kalousis, A., & Hilario, M. (2001). Model Selection via Meta-Learning: A Comparative Study. *International Journal on Artificial Intelligence Tools*, 10(4), 525–554. <https://doi.org/10.1142/S0218213001000647>
- Kiperwasser, E., & Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4, 313–327.
- Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Skip-Thought Vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Hrsg.), *Advances in Neural Information Processing Systems 28* (S. 3294–3302). Curran Associates, Inc. Abgerufen von <http://papers.nips.cc/paper/5950-skip-thought-vectors.pdf>
- Kuncheva, L. I., & Rodríguez, J. J. (2014). A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems*, 38(2), 259–275. <https://doi.org/10.1007/s10115-012-0586-6>

- Le, Q., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (S. 1188–1196). Beijing, China: JMLR.org. Abgerufen von <http://dl.acm.org/citation.cfm?id=3044805.3045025>
- Leacock, C., Miller, G. A., & Chodorow, M. (1998). Using Corpus Statistics and WordNet Relations for Sense Identification. *Comput. Linguist.*, 24(1), 147–165.
- Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), 76–80. <https://doi.org/10.1109/MIC.2003.1167344>
- Liu, T., Liu, S., Chen, Z., & Ma, W.-Y. (2003). An Evaluation on Feature Selection for Text Clustering. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning* (S. 488–495). Washington, DC, USA: AAAI Press. Abgerufen von <http://dl.acm.org/citation.cfm?id=3041838.3041900>
- Logeswaran, L., & Lee, H. (2018). An efficient framework for learning sentence representations. *CoRR, abs/1803.02893*. Abgerufen von <http://arxiv.org/abs/1803.02893>
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.
- March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, 15(4), 251–266. [https://doi.org/10.1016/0167-9236\(94\)00041-2](https://doi.org/10.1016/0167-9236(94)00041-2)
- McAfee, A., & Brynjolfsson, E. (2012, Oktober 1). Big Data: The Management Revolution. *Harvard Business Review*, (October 2012). Abgerufen von <https://hbr.org/2012/10/big-data-the-management-revolution>
- McCallum, A. (2005). Information extraction: distilling structured data from unstructured text. *ACM Queue*, 3(9), 48–57. <https://doi.org/10.1145/1105664.1105679>
- Mehra, N., & Gupta, S. (2013). Survey on Multiclass Classification Methods. *International Journal of Computer Science and Information Technologies*, 4(4), 572–576.
- Michie, D., Spiegelhalter, D. J., Taylor, C. C., & Campbell, J. (Hrsg.). (1994). *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA: Ellis Horwood.

- Mihalcea, R., & Tarau, P. (2004a). TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Abgerufen von <http://aclweb.org/anthology/W04-3252>
- Mihalcea, R., & Tarau, P. (2004b, Juli). TextRank: Bringing Order into Texts [Paper]. Abgerufen 24. Mai 2018, von <https://digital.library.unt.edu/ark:/67531/metadc30962/>
- Mikalef, P., Giannakos, M. N., Pappas, I. O., & Krogstie, J. (2018). The human side of big data: Understanding the skills of the data scientist in education and industry. In *Proceedings of IEEE Global Engineering Education Conference (EDUCON)* (S. 503–512). Tenerife: IEEE. <https://doi.org/10.1109/EDUCON.2018.8363273>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*. Abgerufen von <http://arxiv.org/abs/1301.3781>
- Mikolov, T., Corrado, G. S., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*. Scottsdale, Arizona.
- Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., & Riedl, J. (2003). MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System. In *Proceedings of the 8th International Conference on Intelligent User Interfaces* (S. 263–266). New York, NY, USA: ACM. <https://doi.org/10.1145/604045.604094>
- Milne, D., & Witten, I. (2008). An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy* (S. 25–30). Chicago, USA: AAAI Press.
- Mooney, R. J., & Roy, L. (2000). Content-based Book Recommending Using Learning for Text Categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries* (S. 195–204). New York, NY, USA: ACM. <https://doi.org/10.1145/336597.336662>
- Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticæ Investigationes*, 30(1), 3–26. <https://doi.org/10.1075/li.30.1.03nad>
- Newman, D., Karimi, S., & Cavedon, L. (2009). External evaluation of topic models (S. 1–8). Gehalten auf der Australasian Document Computing Symposium (ADCS), School of Information Technologies, University of Sydney. Abgerufen von <https://researchbank.rmit.edu.au/view/rmit:11686>

- Newman, David, Lau, J. H., Grieser, K., & Baldwin, T. (2010). Automatic Evaluation of Topic Coherence. In *The 2010 Annual Conference of the North American Chapter of the ACL* (S. 100–108). Los Angeles, California: Association for Computational Linguistics.
- Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank Citation Ranking: Bringing Order to the Web*. (Technical Report No. 1999–66). Stanford InfoLab. Abgerufen von <http://ilpubs.stanford.edu:8090/422/>
- Pavithra, M. (2017). A Survey on Clustering High Dimensional Data Techniques. *International Journal of Applied Engineering Research*, 12(11), 2893–2899.
- Pawade, D., Sakhapara, A., Jain, M., Jain, N., & Gada, K. (2018). Story Scrambler - Automatic Text Generation Using Word Level RNN-LSTM. *I.J. Information Technology and Computer Science*, 6, 44–53.
- Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., & Hui, W. (2006). THE DESIGN SCIENCE RESEARCH PROCESS: A MODEL FOR PRODUCING AND PRESENTING INFORMATION SYSTEMS RESEARCH. *Journal of Management Information Systems*, 24(3), 24.
- Pennacchiotti, M., & Gurumurthy, S. (2011). Investigating Topic Models for Social Media User Recommendation. In *Proceedings of the 20th International Conference Companion on World Wide Web* (S. 101–102). New York, NY, USA: ACM. <https://doi.org/10.1145/1963192.1963244>
- Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (S. 1532–1543). Doha, Qatar: Association for Computational Linguistics. <https://doi.org/10.3115/v1/D14-1162>
- Perez, L., & Wang, J. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv:1712.04621 [cs]*. Abgerufen von <http://arxiv.org/abs/1712.04621>
- Perone, C. S., Silveira, R., & Paula, T. S. (2018a). Evaluation of sentence embeddings in downstream and linguistic probing tasks. *CoRR, abs/1806.06259*. Abgerufen von <http://arxiv.org/abs/1806.06259>

- Perone, C. S., Silveira, R., & Paula, T. S. (2018b). Evaluation of sentence embeddings in downstream and linguistic probing tasks. *CoRR, abs/1806.06259*. Abgerufen von <http://arxiv.org/abs/1806.06259>
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *CoRR, abs/1802.05365*. Abgerufen von <http://arxiv.org/abs/1802.05365>
- Polikar, R. (2012). Ensemble Learning. In C. Zhang & Y. Ma (Hrsg.), *Ensemble Machine Learning: Methods and Applications* (S. 1–34). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4419-9326-7_1
- Pressman, R. S. (2015). *Software Engineering: A Practitioner's Approach* (8. Aufl.). Palgrave Macmillan.
- Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., & Riedl, J. (2002). Getting to Know You: Learning New User Preferences in Recommender Systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces* (S. 127–134). New York, NY, USA: ACM. <https://doi.org/10.1145/502716.502737>
- Re, M., & Valentini, G. (2012). Ensemble methods: A review. In M. J. Way, J. D. Scargle, K. M. Ali, & A. N. Srivastava (Hrsg.), *Advances in Machine Learning and Data Mining for Astronomy* (1. Aufl., S. 563–594). Boca Raton: CRC Press.
- Rehm, F., Klawonn, F., & Kruse, R. (2007a). A Novel Approach to Noise Clustering for Outlier Detection. *Soft Computing, 11*(5), 489–494. <https://doi.org/10.1007/s00500-006-0112-4>
- Rehm, F., Klawonn, F., & Kruse, R. (2007b). A Novel Approach to Noise Clustering for Outlier Detection. *Soft Computing, 11*(5), 489–494. <https://doi.org/10.1007/s00500-006-0112-4>
- Rendell, L., Sheshu, R., & Tcheng, D. (1987). Layered Concept-learning and Dynamically Variable Bias Management. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 1* (S. 308–314). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Abgerufen von <http://dl.acm.org/citation.cfm?id=1625015.1625079>
- Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 1994*

ACM Conference on Computer Supported Cooperative Work (S. 175–186). New York, NY, USA: ACM. <https://doi.org/10.1145/192844.192905>

Resnik, P. (1995). Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1* (S. 448–453). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Abgerufen von <http://dl.acm.org/citation.cfm?id=1625855.1625914>

Rinaldi, F., Hess, M., Dowdall, J., Mollá, D., & Schwitter, R. (2004). Question answering in terminology-rich technical domains. In *New directions in question answering* (S. 71–82). United States: Association for the Advancement of Artificial Intelligence.

Robertson, S., & Robertson, J. (2012). *Mastering the Requirements Process: Getting Requirements Right*. Pearson Education. Abgerufen von <https://books.google.de/books?id=yE91LgrpaHsC>

Robertson, Suzanne, & Robertson, J. (2012). *Mastering the Requirements Process: Getting Requirements Right*. Addison-Wesley.

Saha, S. K., Narayan, S., Sarkar, S., & Mitra, P. (2010). A composite kernel for named entity recognition. *Pattern Recognition Letters*, 31(12), 1591–1597. <https://doi.org/10.1016/j.patrec.2010.05.004>

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513–523. [https://doi.org/10.1016/0306-4573\(88\)90021-0](https://doi.org/10.1016/0306-4573(88)90021-0)

Schnabel, T., Labutov, I., Mimno, D., & Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (S. 298–307). Lisbon, Portugal: Association for Computational Linguistics. Abgerufen von <http://aclweb.org/anthology/D15-1036>

Schneider, J. (2017). Topic Modeling based on Keywords and Context. *arXiv:1710.02650 [cs]*. Abgerufen von <http://arxiv.org/abs/1710.02650>

Schutt, R., & O’Neil, C. (2013). *Doing Data Science: Straight Talk from the Frontline*. O'Reilly Media, Inc.

Schütze, H. (1998). Automatic Word Sense Discrimination. *Comput. Linguist.*, 24(1), 97–123.

- Sennrich, R., Haddow, B., & Birch, A. (2016). Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (S. 86–96). Berlin, Germany: Association for Computational Linguistics. <https://doi.org/10.18653/v1/P16-1009>
- Serban, F., Vanschoren, J., Kietz, J.-U., & Bernstein, A. (2013). A survey of intelligent assistants for data analysis. *ACM Computing Surveys (CSUR)*, 45(3), 31. <https://doi.org/10.1145/2480741.2480748>
- Sleeman, D., Rissakis, M., Craw, S., Graner, N., & Sharma, S. (1995). Consultant-2: pre- and post-processing of Machine Learning applications. *International Journal of Human-Computer Studies*, 43(1), 43–63. <https://doi.org/10.1006/ijhc.1995.1035>
- Sokolova, M., & Lapalme, G. (2009). A Systematic Analysis of Performance Measures for Classification Tasks. *Inf. Process. Manage.*, 45(4), 427–437. <https://doi.org/10.1016/j.ipm.2009.03.002>
- Sussillo, D., & Abbott, L. F. (2014). Random Walk Initialization for Training Very Deep Feedforward Networks. *arXiv:1412.6558 [cs, stat]*. Abgerufen von <http://arxiv.org/abs/1412.6558>
- Sutskever, I., Martens, J., & Hinton, G. (2011). Generating Text with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning* (S. 1017–1024). USA: Omnipress. Abgerufen von <http://dl.acm.org/citation.cfm?id=3104482.3104610>
- Thenmalar, Balaji, J., & Geetha, T. . (2015). Semi-Supervised Bootstrapping Approach for Named Entity Recognition. *International Journal on Natural Language Computing*, 4(5), 01–14. <https://doi.org/10.5121/ijnlc.2015.4501>
- Todor, A., Lukasiewicz, W., Athan, T., & Paschke, A. (2016). Enriching Topic Models with DBpedia. In C. Debruyne, H. Panetto, R. Meersman, T. Dillon, eva Kühn, D. O’Sullivan, & C. A. Ardagna (Hrsg.), *On the Move to Meaningful Internet Systems: OTM 2016 Conferences* (S. 735–751). Cham: Springer International Publishing.
- Todorovski, L., Blockeel, H., & Dzeroski, S. (2002). Ranking with Predictive Clustering Trees. In *Proceedings of the 13th European Conference on Machine Learning* (S. 444–455). London, UK, UK: Springer-Verlag. Abgerufen von <http://dl.acm.org/citation.cfm?id=645329.650061>

- Tsai, C., Lai, C., Chiang, M., & Yang, L. T. (2014). Data Mining for Internet of Things: A Survey. *IEEE Communications Surveys Tutorials*, 16(1), 77–97. <https://doi.org/10.1109/SURV.2013.103013.00206>
- Tsvetkov, Y., Faruqui, M., Ling, W., Lample, G., & Dyer, C. (2015). Evaluation of Word Vector Representations by Subspace Alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (S. 2049–2054). Lisbon, Portugal: Association for Computational Linguistics. Abgerufen von <http://aclweb.org/anthology/D15-1243>
- Unterstein, G., Michael Matthiessen. (2013). *Anwendungsentwicklung mit Datenbanken*. Berlin, Heidelberg: Springer.
- Vanschoren, J. (2010). *Understanding Machine Learning Performance with Experiment Databases* (PhD). KU Leuven, Arenberg Doctoral School of Science, Engineering & Technology - Faculty of Engineering - Department of Computer Science, Leuven. Abgerufen von https://limo.libis.be/primo-explore/fulldisplay?docid=LIRIAS1652449&context=L&vid=Lirias&search_scope=Lirias&tab=default_tab&lang=en_US&fromSitemap=1
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Hrsg.), *Advances in Neural Information Processing Systems 30* (S. 5998–6008). Curran Associates, Inc. Abgerufen von <http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>
- Vieira, S. M., Kaymak, U., & Sousa, J. M. C. (2010). Cohen's kappa coefficient as a performance measure for feature selection. In *International Conference on Fuzzy Systems* (S. 924–931). <https://doi.org/10.1109/FUZZY.2010.5584447>
- Vijaymeena, M. K., & Kavitha, K. (2016). A SURVEY ON SIMILARITY MEASURES IN TEXT MINING. *Machine Learning and Applications: An International Journal (MLAI)*, 3(1), 19–28.
- Wang, C., & Blei, D. M. (2011). Collaborative Topic Modeling for Recommending Scientific Articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (S. 448–456). New York, NY, USA: ACM. <https://doi.org/10.1145/2020408.2020480>

- Wang, P., Xu, B., Xu, J., Tian, G., Liu, C.-L., & Hao, H. (2016). Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, 174, 806–814. <https://doi.org/10.1016/j.neucom.2015.09.096>
- Wang, W. Y., & Yang, D. (2015). That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (S. 2557–2563). Lisbon, Portugal: Association for Computational Linguistics. Abgerufen von <http://aclweb.org/anthology/D15-1306>
- Wang, Z., Hahn, K., Kim, Y., Song, S., & Seo, J.-M. (2018). A news-topic recommender system based on keywords extraction. *Multimedia Tools and Applications*, 77(4), 4339–4353. <https://doi.org/10.1007/s11042-017-5513-0>
- Wong, W., Liu, W., & Bennamoun, M. (2012). Ontology Learning from Text: A Look Back and into the Future. *ACM Comput. Surv.*, 44(4), 20:1–20:36. <https://doi.org/10.1145/2333112.2333115>
- Wu, Z., & Palmer, M. (1994). Verbs Semantics and Lexical Selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics* (S. 133–138). Stroudsburg, PA, USA: Association for Computational Linguistics. <https://doi.org/10.3115/981732.981751>
- Xu, J. A., & Araki, K. (2006). A SVM-based personal recommendation system for TV programs. In *2006 12th International Multi-Media Modelling Conference* (S. 4 pp.-). <https://doi.org/10.1109/MMMC.2006.1651358>
- Yadav, V., & Bethard, S. (2018). A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In *Proceedings of the 27th International Conference on Computational Linguistics* (S. 2145–2158). Santa Fe, New Mexico, USA: Association for Computational Linguistics. Abgerufen von <http://www.aclweb.org/anthology/C18-1182>
- Yan, T. W., & Garcia-Molina, H. (1995). SIFT: A Tool for Wide-area Information Dissemination. In *Proceedings of the USENIX 1995 Technical Conference Proceedings* (S. 15–

15). Berkeley, CA, USA: USENIX Association. Abgerufen von <http://dl.acm.org/citation.cfm?id=1267411.1267426>

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Computational Intelligence Magazine*, 13(3), 55–75. <https://doi.org/10.1109/MCI.2018.2840738>

Zhang, X., & LeCun, Y. (2015). Text Understanding from Scratch. *arXiv:1502.01710 [cs]*. Abgerufen von <http://arxiv.org/abs/1502.01710>

Zhang, Y., Rahman, M. M., Braylan, A., Dang, B., Chang, H.-L., Kim, H., ... Lease, M. (2016). Neural Information Retrieval: A Literature Review. *arXiv:1611.06792 [cs]*. Abgerufen von <http://arxiv.org/abs/1611.06792>

Anhang

Die folgende Tabelle gibt eine Übersicht der verwendeten Suchanfragen beim Crawling der Datenbank *ArXiv* dem Datenbankverbund *Elsevier* sowie dem Online-Journal *Medium*. Wie bereits unter **Kapitel XX** beschrieben, wird versucht die Anfragen unter den jeweils gegebenen technischen Rahmenbedingungen zwischen den Datenbanken zu standardisieren. Beim Online-Journal Medium ist allerdings eine Ausnahme zu verzeichnen, da hier direkt nach den Schlagwörtern gesucht wird. Dieses Vorgehen resultiert aus der verfügbaren, geringeren Datenmenge auf Medium sowie der nicht vorhandenen Metadatensuche auf dieser Seite.

Tabelle A 1: Datenbanksuchanfragen beim Crawling

Datenbank	Anfrage
<i>Elsevier</i>	Title-Abstr-Key(Clustering AND data) AND Title(Clustering)
<i>Elsevier</i>	Title-Abstr-Key(„Clustering is“) AND Title(Clustering)
<i>Elsevier</i>	Title-Abstr-Key(„Classification is“) AND Title(Classification OR Prediction)
<i>Elsevier</i>	Title-Abstr-Key(Classification AND Prediction) AND Title(Classification OR Prediction)
<i>Elsevier</i>	Title-Abstr-Key(„Regression is“) AND Title(Prediction)
<i>Elsevier</i>	Title-Abstr-Key(Regression AND data) AND Title(Prediction)
<i>Elsevier</i>	Title-Abstr-Key(„Sequence Analysis is“) AND Title(Sequence)
<i>Elsevier</i>	Title-Abstr-Key(„Sequence Analysis“ AND data) AND Title(Sequence)
<i>Elsevier</i>	Title-Abstr-Key(„Sequence Analytics“ AND data) AND Title(Sequence)
<i>Elsevier</i>	Title-Abstr-Key(„Association rule“ AND data) AND Title(Association)
<i>Elsevier</i>	Title-Abstr-Key(„Association rule mining is“) AND Title(Association)
<i>Elsevier</i>	Title-Abstr-Key(„Sequential Pattern“)
<i>Elsevier</i>	Title-Abstr-Key(„Frequent Pattern“)
<i>ArXiv</i>	

ArXiv

ArXiv

ArXiv

ArXiv

ArXiv

Medium Clustering

Medium Classification

Medium Regression

Medium Frequent Pattern

Medium Association rules

Medium Market-Basket Analysis

Medium Sequential Pattern

Medium Sequence Mining

Kommentiert [m61]:

Eidesstattliche Erklärung

Autoren: Richard Horn, Daniel Höschele

Matrikelnummer: 3885589, 3879394

Eidesstattliche Erklärung

Die Autoren erklären hiermit an Eides statt, dass die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Richard Horn

Daniel Höschele

Dresden, 29.09.2017