

Lehrstuhl für Wirtschaftsinformatik | Business Intelligence Research

Technische Universität Dresden
Fakultät Wirtschaftswissenschaften

Ein Textbasiertes Recommender- System zur Automatisierten Selektion von Analytischen Verfahrensklassen

Richard Horn (3885589)¹

Daniel Höschele (3879394)²

Masterarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

Prüfer: **Prof. Dr. Susanne Strahringer**

Betreuer: **Patrick Zschech**

Eingereicht am: **31.03.2019**

Bearbeitungszeitraum: **01.12.2018-31.03.2019**

Quantitative Zusammenfassung



| Eigenschaft | Wert |
|-------------------------------|------|
| Quellenanzahl gesamt | 153 |
| Internetquellen | 1 |
| Zeitlicher Rahmen | |
| Älteste Quelle | 1987 |
| Neueste Quelle | 2019 |
| Anteile | |
| Bis 1999: | 22 |
| 2000 – 2010 | 40 |
| 2011 bis heute | 91 |
| Anzahl der Abbildungen | |
| 51 | |
| Anzahl der Tabellen | |
| 23 | |
| Anzahl der Formeln | |
| 37 | |

Kurzfassung

Das Areal der Data Science ist ein stark interdisziplinäres Forschungsfeld, um Erkenntnisse und Wissen aus Daten zu extrahieren. Dabei werden komplexe Methoden der Statistik, Mathematik Informatik sowie des Machine Learning kombiniert. Daneben erfordert der Umgang mit themenspezifischen Daten umfassendes Domänenwissen, dessen Aneignung als sehr zeitaufwendig sowie komplex anzusehen ist. In Anbetracht dessen versucht der vorliegende Beitrag die Lücke zwischen Domänenexperten und Data Scientists durch die Applikation eines Recommender-Systems zu schließen. Das zentrale Forschungsziel beschäftigt sich hierbei mit der Identifikation geeigneter Methoden sowie der Entwicklung eines initialen Prototyps. Dabei präsentiert die Arbeit ein Recommender-System, welches auf Basis natürlichsprachlicher Problemstellungen Empfehlungen bezüglich der Verfahrensklasse der Data-Science ausspricht. Additional erzeugt der Prototyp eine exemplarische Analyse der Problemstellung auf Basis extrahierter Analyseentitäten und gibt eine Empfehlung hinsichtlich der benötigten Datenstruktur. Insgesamt zeigt der Forschungsbeitrag zudem ein Verfahren wie ein solches System auf Basis weniger Daten umgesetzt sowie evaluiert werden kann.

Schlüsselwörter: Data Science
Recommender-System
Machine Learning
Natural Language Processing
Topic Modeling
Named-Entity Recognition
Keyword Extraction

Abstract

The Data Science area is a highly interdisciplinary field of research for extracting knowledge from data. The area combines complex methods of statistics, mathematics, computer science and machine learning. Moreover, the handling of domain-specific data requires comprehensive domain knowledge, the acquisition of which is to be understood as very time-consuming and complex. In view of this, this paper attempts to close the gap between domain experts and data scientists by applying a recommender system. The central research objective of this paper is the identification of suitable methods and the development of an initial prototype. The paper presents a recommender system, which makes recommendations regarding the data science method group by using natural language problem descriptions only. In addition, the prototype generates an exemplary analysis of the problem and a recommendation regarding the required data structure. Overall, the research paper also shows a procedure how such a system can be implemented and evaluated while only having a small amount of data.

Keywords: Data Science
Recommender System
Machine Learning
Natural Language Processing
Topic Modeling
Named-Entity Recognition
Keyword Extraction

Inhaltsverzeichnis

| | |
|--|------------|
| ABBILDUNGSVERZEICHNIS | IV |
| TABELLENVERZEICHNIS | VII |
| FORMELVERZEICHNIS | IX |
| ABKÜRZUNGSVERZEICHNIS | XI |
| 1 THEMENBESCHREIBUNG UND MOTIVATION | 1 |
| 1.1 EINORDNUNG, ABGRENZUNG UND HERAUSFORDERUNGEN..... | 3 |
| 1.2 FORSCHUNGSVORHABEN..... | 8 |
| 1.2.1 <i>Forschungsdesign</i> | 8 |
| 1.2.2 <i>Forschungsfragen</i> | 9 |
| 1.3 METHODISCHES VORGEHEN | 9 |
| 1.3.1 <i>Literaturrecherche mittels Rückwärtssuche</i> | 10 |
| 1.3.2 <i>Der Design-Science-Research-Prozess</i> | 11 |
| 1.4 AUFBAU UND STRUKTUR DER ARBEIT | 15 |
| 2 ÜBERBLICK DES FORSCHUNGSEMINARS | 17 |
| 3 FORSCHUNGSSTAND | 20 |
| 4 ADDITIONALE METHODEN | 26 |
| 4.1 EMBEDDINGS | 26 |
| 4.1.1 <i>FastText</i> | 27 |
| 4.1.2 <i>Deep Averaging Networks</i> | 29 |
| 4.1.3 <i>Universal Sentence Encoder</i> | 31 |
| 4.2 RNN-BASIERTE TECHNOLOGIEN | 33 |
| 4.2.1 <i>Long Short-Term Memory</i> | 35 |

| | | |
|----------|--|-----------|
| 4.2.2 | <i>Gated Recurrent Units</i> | 37 |
| 4.3 | KEYWORD-EXTRACTION-METHODEN | 38 |
| 4.3.1 | <i>YAKE</i> | 39 |
| 4.3.2 | <i>PositionRank</i> | 40 |
| 4.3.3 | <i>TopicRank</i> | 41 |
| 4.4 | ENSEMBLE LEARNING | 44 |
| 4.4.1 | <i>Nicht-generative Methoden des Ensemble Learnings</i> | 45 |
| 4.4.2 | <i>Generative Methoden des Ensemble Learnings</i> | 45 |
| 4.5 | NAMED-ENTITY RECOGNITION | 48 |
| 4.5.1 | <i>Wörterbuchbasierte und regelbasierte Vorgehen</i> | 48 |
| 4.5.2 | <i>Lernbasierte Verfahren der Named-Entity Recognition</i> | 49 |
| 5 | PROZESS DER DATENBESCHAFFUNG | 52 |
| 5.1 | CRAWLING VON DATENBANKEN | 52 |
| 5.2 | DATENEVALUIERUNG UND -SELEKTION..... | 55 |
| 5.2.1 | <i>Regelbasierte Datenbereinigung</i> | 55 |
| 5.2.2 | <i>Entfernen von Ausreißern</i> | 56 |
| 5.3 | DATA AUGMENTATION | 60 |
| 5.3.1 | <i>Methoden zur Substitution und Reihenfolgenanpassung</i> | 60 |
| 5.3.2 | <i>Generierung von Textdaten</i> | 61 |
| 5.3.3 | <i>Applizierte Methoden der Data Augmentation</i> | 63 |
| 5.4 | ERSTELLUNG VON DATENSÄTZEN | 66 |
| 5.4.1 | <i>Erstellung der Trainingsdatensätze</i> | 66 |
| 5.4.2 | <i>Erstellung der Validationsdatensätze</i> | 69 |
| 6 | KONZEPTION UND IMPLEMENTIERUNG DES PROTOTYP | 71 |
| 6.1 | ANFORDERUNGSDEFINITION | 71 |
| 6.2 | FRONTEND KONSEPTIONIERUNG..... | 74 |

| | | |
|-----------|--|------------|
| 6.3 | ENTWICKLUNG DES BACKEND..... | 76 |
| 6.3.1 | <i>Vorbereitung und Vorverarbeitung der Daten</i> | 76 |
| 6.3.2 | <i>Aufbau der Knowledge Base</i> | 78 |
| 6.3.3 | <i>Nutzung der Knowledge Base</i> | 82 |
| 6.4 | PROGRAMMSTRUKTUR | 87 |
| 6.5 | ANFORDERUNGSGEGENÜBERSTELLUNG | 89 |
| 7 | EVALUATION DER METHODEN SOWIE DES GESAMTSYSTEMS | 94 |
| 7.1 | EVALUATION VON EMBEDDINGS..... | 94 |
| 7.1.1 | <i>Extrinsische Evaluation</i> | 95 |
| 7.1.2 | <i>Intrinsische Evaluation</i> | 99 |
| 7.1.3 | <i>Ergebnisse der Evaluation</i> | 102 |
| 7.2 | EVALUATION VON TOPIC-MODELLEN | 105 |
| 7.2.1 | <i>Überblick der Evaluationsmethoden</i> | 105 |
| 7.2.2 | <i>Ergebnisse der Evaluation von Topic-Modellen</i> | 107 |
| 7.3 | EVALUIERUNG DER KLASSENIKATIONSMODELLE | 113 |
| 7.3.1 | <i>Überblick multinomialer Evaluationsmetriken</i> | 113 |
| 7.3.2 | <i>Ergebnisse der Evaluation von Klassifikationsmethoden</i> | 116 |
| 8 | ERGEBNISSE | 122 |
| 9 | KRITISCHE WÜRDIGUNG | 128 |
| 10 | FAZIT UND AUSBLICK | 131 |
| | LITERATURVERZEICHNIS..... | XII |
| | ANHANG | XXVI |
| | EIDESSTATTLICHE ERKLÄRUNG | XXXII |

Abbildungsverzeichnis

| | |
|--|----|
| Abbildung 1: Illustration der allgemeinen Zielstellung..... | 2 |
| Abbildung 2: Vorgehen zur Expansion der Literaturbasis..... | 10 |
| Abbildung 3: Der Design-Science-Research-Prozess..... | 12 |
| Abbildung 4: Vorgehensmodell des Prototyping..... | 13 |
| Abbildung 5: Struktureller Aufbau der Arbeit..... | 16 |
| Abbildung 6: Übersicht der methodischen Kategorien | 17 |
| Abbildung 7: Architektur des Word2Vec-Modells..... | 28 |
| Abbildung 8: Funktionsweise eines DANs | 29 |
| Abbildung 9: Gegenüberstellung DAN und Mittelwertsverfahren..... | 31 |
| Abbildung 10: Berechnungsgraph eines Recurrent Neural Networks | 33 |
| Abbildung 11: Illustration der Funktionsweise einer LSTM-Zelle | 35 |
| Abbildung 12: Illustration der Funktionsweise einer GRU-Zelle | 37 |
| Abbildung 13: Verarbeitungsschritte des TopicRank-Algorithmus | 42 |
| Abbildung 14: Illustration der Funktionsweise des Bagging-Ansatzes | 46 |
| Abbildung 15: Darstellung des Vorgehens bei Boosting..... | 47 |
| Abbildung 16: Prozess lernbasierter Verfahren der Named-Entity Recognition..... | 49 |
| Abbildung 17: Prozess der Datenbeschaffung und -selektion..... | 52 |
| Abbildung 18: Crawling-Ergebnisse in Elsevier..... | 54 |
| Abbildung 19: Density Graph des Datenpools | 56 |
| Abbildung 20: Streudiagramme des Datenpools | 57 |
| Abbildung 21: Durch DBSCAN identifizierte Ausreißer..... | 59 |
| Abbildung 22: Dichtediagramm augmentierter Daten des gefilterten Datensatzes | 66 |

| | |
|--|-----|
| Abbildung 23: Prozess zur Generierung des gefilterten Datensatzes | 67 |
| Abbildung 24: Illustration der Kosinusähnlichkeit zur Datenauswahl | 68 |
| Abbildung 25: Klassenverteilung der finalen Datensätze | 69 |
| Abbildung 26: Übersicht der Datenmenge in verschiedenen Verarbeitungsschritten .. | 70 |
| Abbildung 27: Wireframe der initialen Nutzeroberfläche | 74 |
| Abbildung 28: Wireframe der Ergebnisseite des Prototyps | 75 |
| Abbildung 29: Illustration der identifizierten Entwicklungsebenen..... | 76 |
| Abbildung 30: Klassenübersicht zur Datenvorverarbeitung..... | 78 |
| Abbildung 31: Konzeptioneller Entwurf der Knowledge Base..... | 79 |
| Abbildung 32: Illustration von potentiellen Vorgehen in der Klassifikation | 82 |
| Abbildung 33: Das Weighted-Averaging-Prinzip an einem fiktiven Beispiel | 84 |
| Abbildung 34: Illustration des Vorgehens zur Extraktion der Analyseentitäten | 85 |
| Abbildung 35: Klassenstruktur der benötigten Klassen zur Verfahrenszuweisung | 87 |
| Abbildung 36: Model-View-Controller-Architektur des Prototyps..... | 88 |
| Abbildung 37: Struktur des implementierten Python-Pakets auf Model-Ebene | 89 |
| Abbildung 38: Illustration der Ergebnisseite des Prototyps | 91 |
| Abbildung 39: Ausgewählte Korrelationen von Modellparametern (FastText) | 96 |
| Abbildung 40: Streudiagramme von FastText-Embeddings der gefilterten Daten | 97 |
| Abbildung 41: Ausgewählte Korrelationen von Modellparametern (DAN)..... | 98 |
| Abbildung 42: Deep-Averaging-Netzwerke mit multipler Layer-Anzahl..... | 99 |
| Abbildung 43: Ähnlichkeitsmatrix zur Evaluierung von Embedding-Modellen..... | 101 |
| Abbildung 44: Illustration der Evaluationsergebnisse der Keyword Extraction..... | 109 |
| Abbildung 45: Parameterkorrelationen der LDA-Modelle mit n>3 | 110 |
| Abbildung 46: Parameterkorrelationen der LDA-Modelle mit n=3 | 111 |
| Abbildung 47: Visualisierung eines LDA-Modells mittels pyLDAvis..... | 112 |

| | |
|--|------|
| Abbildung 48: Übersicht von Schlüsselwörtern der trainierten LDA-Modelle | 112 |
| Abbildung 49: Darstellung der Startseite des entwickelten Prototyps..... | 123 |
| Abbildung A 1: Evaluationswerte von Keyword-Modellen auf augmentierten Daten | XXX |
| Abbildung A 2: Klassendiagramm des Prototyps..... | XXXI |

Tabellenverzeichnis

| | |
|--|------|
| Tabelle 1: Gegenüberstellung von technischen Artikeln und Problemstellungen..... | 7 |
| Tabelle 2: Übersicht der Parametersuche für Noise-Clustering | 58 |
| Tabelle 3: Kriterien zur Augmentierung von Daten mittels Substitution..... | 63 |
| Tabelle 4: Exemplarische Substituierung von Synonymen in Sätzen | 64 |
| Tabelle 5: Beispiel für generierte Sätze mittels eines Markov-Modells | 65 |
| Tabelle 6: Übersicht über die erstellten Datensätze | 69 |
| Tabelle 7: Prototypbezogene Anforderungsdefinition..... | 72 |
| Tabelle 8: Zuordnung von Vorverarbeitungsmethoden zu Methoden des KB-Aufbaus | 77 |
| Tabelle 9: Zuordnung der Methoden zu Entitäten der aufgebauten Knowledge Base . | 80 |
| Tabelle 10: Bewertung der definierten Anforderungen | 90 |
| Tabelle 11: Übersicht der Parameterkombinationen für Embedding-Modelle | 102 |
| Tabelle 12: Ergebnisübersicht der Ähnlichkeitsevaluation von Embedding-Modellen | 103 |
| Tabelle 13: Übersicht der Evaluationsergebnisse von DAN-Modellen | 104 |
| Tabelle 14: Evaluationsergebnisse von Topic-Modellen | 108 |
| Tabelle 15: Überblick multinomialer Evaluationsmetriken | 114 |
| Tabelle 16:Evaluationsergebnisse auf gefilterten Daten..... | 117 |
| Tabelle 17: Evaluationsergebnisse auf augmentierten Daten | 119 |
| Tabelle 18: Ergebnisse der Klassifikationen mit LDA-Modellen..... | 120 |
| Tabelle 19: Ergebnisse des Ensemble Learning | 121 |
| Tabelle 20: Übersicht der identifizierten Design-Prinzipien | 124 |
| Tabelle 21: Minima und Maxima der Bewertung von Embedding-Modellen | 125 |
| Tabelle A 1: Datenbanksuchanfragen beim Crawling | XXVI |

Tabelle A 2: Kohärenzwerte der Keyword-Modelle auf augmentierten Daten XXIX

Formelverzeichnis

| | |
|--|----|
| (1) Aktivitätsinput eines RNN-Zeitstempels..... | 34 |
| (2) Aktivierung eines RNN-Zeitstempels..... | 34 |
| (3) Forget Gate eines LSTMs | 35 |
| (4) Input Gate eines LSTMs | 36 |
| (5) Potentielle Kandidaten eines LSTM Zellstatus | 36 |
| (6) Zellstatus eines LSTM | 36 |
| (7) Auswahl weiterzugebender Informationen in einer LSTM-Zelle | 36 |
| (8) Output einer LSTM-Zelle..... | 36 |
| (9) Update Gate eines GRU | 38 |
| (10) Reset Gate eines GRU | 38 |
| (11) Aktivierung eines Inputs in einer GRU-Zelle | 38 |
| (12) Output einer GRU-Zelle..... | 38 |
| (13) Gewichtung großgeschriebener Wörter | 39 |
| (14) Gewichtung der Wortposition nach YAKE | 39 |
| (15) Bewertung von Stopword-Charakteristika | 40 |
| (16) Bewertung eines Wortes nach YAKE | 40 |
| (17) Bewertung eines Keyphrases nach YAKE | 40 |
| (18) Bewertung der Wortposition nach PositionRank | 41 |
| (19) Berechnung der Distanz zwischen Schlüsselwörtern..... | 43 |
| (20) Gewichtung von Schlüsselwortbeziehungen..... | 43 |
| (21) Bewertung von Schlüsselwörtern nach PositionRank | 43 |
| (22) Regulärer Ausdruck zum Entfernen von Formeln | 55 |

| | |
|--|-----|
| (23) Regulärer Ausdruck zum Entfernen von Variablen..... | 55 |
| (24) Davies-Bouldin-Index | 58 |
| (25) Gemittelte euklidische Distanz..... | 58 |
| (26) Konditionale Wahrscheinlichkeit bei gegebenen Bigramm | 65 |
| (27) Bewertung von Analyseentitäten | 86 |
| (28) Der Parameter Gamma einer Support Vector Machine | 96 |
| (29) Purity | 100 |
| (30) Mean-Error..... | 102 |
| (31) Spearman-Rangkorrelationskoeffizient | 102 |
| (32) Bewertung der Wortähnlichkeit 1..... | 105 |
| (33) Bewertung der Wortähnlichkeit 2..... | 106 |
| (34) Related Article Concept Overlap | 106 |
| (35) Perplexity..... | 107 |
| (36) F1-Score..... | 115 |
| (37) Cohen's Kappa | 115 |

Abkürzungsverzeichnis

| | |
|--------|---|
| AJAX | Asynchronous Java Script |
| CNN | Convolutional Neural Network |
| CS | Cell State |
| CSS | Cascading Style Sheets |
| DAN | Deep-Averaging-Networks |
| DS | Data Science |
| DSP | Design-Science-Research-Prozess |
| EL | Ensemble Learning |
| ELMO | Embedding from Language Models |
| GLoVe | Global Vectors |
| GRU | Gated Recurrent Unit |
| HTML | Hypertext Markup Language |
| KB | Knowledge Base |
| KDD | Knowledge Discovery in Databases |
| LDA | Latent Dirichlet Allocation |
| LSTM | Long Short-Term Memory |
| ML | Machine Learning |
| NC | Noise-Clustering |
| NER | Named-Entity Recognition |
| NLP | Natural Language Processing |
| NMT | Neural Machine Translation |
| PCA | Principle Component Analysis |
| POS | Part of Speech |
| RNN | Recurrent Neural Network |
| SVM | Support Vector Machine |
| TF-IDF | Term Frequency - Inverse Document Frequency |
| TF-IGM | Term Frequency - Inverse Gravity Moment |
| USE | Universal Sentence Encoder |

1 Themenbeschreibung und Motivation²

Das Areal der Data Science (DS) ist ein an Popularität und Relevanz gewinnendes Feld, sowohl seitens der Wissenschaft als auch seitens von Unternehmen. Dabei ist die DS als ein stark interdisziplinäres Feld zu charakterisieren, welches die Kombination von mathematischem, informatischem sowie domänen spezifischem 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 und Mikalef, Giannakos, Pappas & Krogstie, 2018, S. 503f.). Als Unterstützung der Kommunikation zwischen den Personengruppen ordnet sich 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 geeigneten 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, inklusive einer exemplarischen Analyse, als Empfehlung an den Nutzer zurückgegeben, um somit Unterstützung für Personen ohne Fachkenntnisse

der Datenanalyse zu bieten. So wird im Anschluss an die vorangegangene Forschungsarbeit in der vorliegenden Arbeit der zweite Teil des Forschungsauftrags fokussiert.

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.

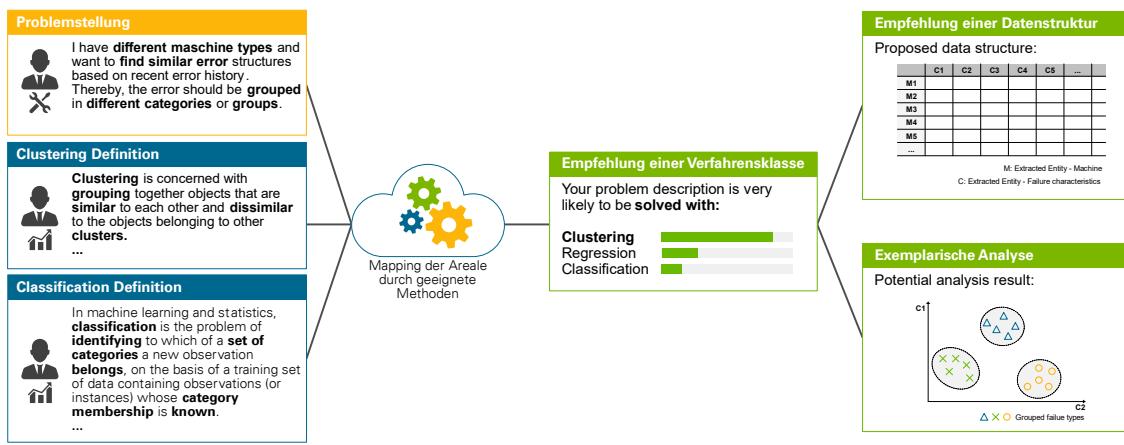


Abbildung 1: Illustration der allgemeinen Zielstellung (Eigene Darstellung)

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 Strukturbeschreibung der benötigten Daten sowie eine exemplarische Analyse anhand identifizierter Entitäten der Problemstel-

lung. Diese Entitäten gilt es hierfür im eingegebenen Text mittels ausgewählter 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 Herausforderungen¹

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** deskribiert. 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 Charakteristika 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 (vgl. Serban et al., 2013, S. 31:8ff.).

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 aus einer Domäne 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 Anwender eine exemplarische Analyse der eruierten Verfahrensklasse, unter Einbezug der zu untersuchenden Entitäten der Problemstellung, ausgegeben. Diese ermöglicht es ihm 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 **inhaltsbasierte Recommender-Systeme**, welche alleinig auf Basis eines Nutzerprofils und inhaltlichen Informationen der zu empfehlenden Daten arbeiten und Empfehlungen aussprechen. Exemplarisch bedeutet diese Fokussierung 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 technische bzw. wissenschaftliche Publikationstexte und eine sehr geringe Anzahl an Problembeschreibungen als initiale Datenbasis zur Verfügung stehen. Dazu sei auf Kapitel 5 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 neuen 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 domänen spezifischen Terminologie eine geeignete Verfahrensklasse.

Eine der großen Herausforderungen besteht in der Verarbeitung von Informationen 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. domänen spezifischen Terminologien, ist davon auszugehen, dass Problemstellungen different von beiden Personengruppen definiert werden.

Wie aus den obenstehenden Restriktionen hervorgeht, bilden wissenschaftliche sowie technische Artikel die einzige zur Verfügung stehende Datengrundlage, um den Prototyp des Recommender-Systems zu implementieren. Im Vergleich zum Eingabetext der fachfremden Person besitzen derartige Artikel andere linguistische Eigenschaften. Technische bzw. wissenschaftliche Artikel setzen häufig domänen spezifisches Wissen und die

Kenntnis der jeweiligen Terminologie voraus und bedienen sich komplexen wissenschaftlichen Konstrukten. Dabei bestehen die kommunikative Funktion und die Intention von technischen Artikeln im Übermitteln von Wissen in Form von abstrakten Konstrukten, Konzepten und Prinzipien, um Lösungen für konkrete Probleme zu erreichen. Hinzu kommt, dass technische Artikel wenig Spielraum für semantische Redundanzen lassen und Informationen strukturiert im Text wiederzufinden sind (vgl. Rinaldi, Hess, Dowdall, Mollá & Schwitter, 2004, S. 135f.). Konträr besteht die wesentliche Intention von Problembeschreibungen darin, 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 & Rodríguez, 2012, S. 9f.). Tabelle 1 gibt einen Überblick über die Unterschiede der zugrundeliegenden Daten des zu entwickelten Prototyps.

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 Problemen | 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 Forschungsvorhaben²

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 Forschungsdesign²

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 Realitätsempfinden stützt und damit gleichzeitig impliziert, dass die daraus resultierende Erkenntnis ebenfalls nur subjektiver Natur sein kann (vgl. Becker, Holten, Knackstedt & Niehaves, 2003, S. 18). Dabei wird bei der Forschungsmethode hauptsächlich ein deduktives Vorgehen gewählt.

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 Vorgehen²

Das folgende Kapitel beschäftigt sich mit dem methodischen Vorgehen, das der Durchführung des Forschungsvorhabens zugrunde liegt. Hierbei wird zunächst auf die Methodik zur 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 Literaturrecherche mittels Rückwärtssuche²

Als Grundlage für das aktuelle Forschungsvorhaben dient die in vorheriger Arbeit durchgeführte systematische Literaturrecherche nach FETTKE (2006), die „[...] aus der Perspektive einer bestimmten Fragestellung die zu einem Themengebiet relevanten Arbeiten und vorliegenden Erkenntnisse“ (Fettke, 2006, S. 258) untersucht. Diese hat zur Identifikation von potentiellen Methodengruppen zur Erreichung der definierten Zielstellung fungiert und deckt somit eine große Spannbreite an Methodenfeldern ab. Im Rahmen dieser Recherche sind die wissenschaftlichen Datenbanken *Science Direct*, *IEEE Explore*, *Google Scholar* sowie *arXiv* durchsucht worden. Hierbei sind Term-Sets definiert worden, die unterschiedliche Granularität aufweisen und von allgemeinen Kategorien, über bestimmte Methodengruppen bis zu konkreten Algorithmen reichen. Aufgrund der Einschränkung dieser Ergebnisse auf die Top-Suchergebnisse pro Anfrage, werden die hierbei erlangten Erkenntnisse als Grundlage für die vorliegende Arbeit herangezogen und mit der Methode der Rückwärtssuche weiterhin expandiert. Die Rückwärtssuche beschreibt das Vorgehen, eine Erweiterung der existenten Quellenbasis mithilfe der Verkettung von Referenzen, ausgehend von aktuellen Publikationen, zu erreichen (vgl. Webster & Watson, 2002, S. 15f.).

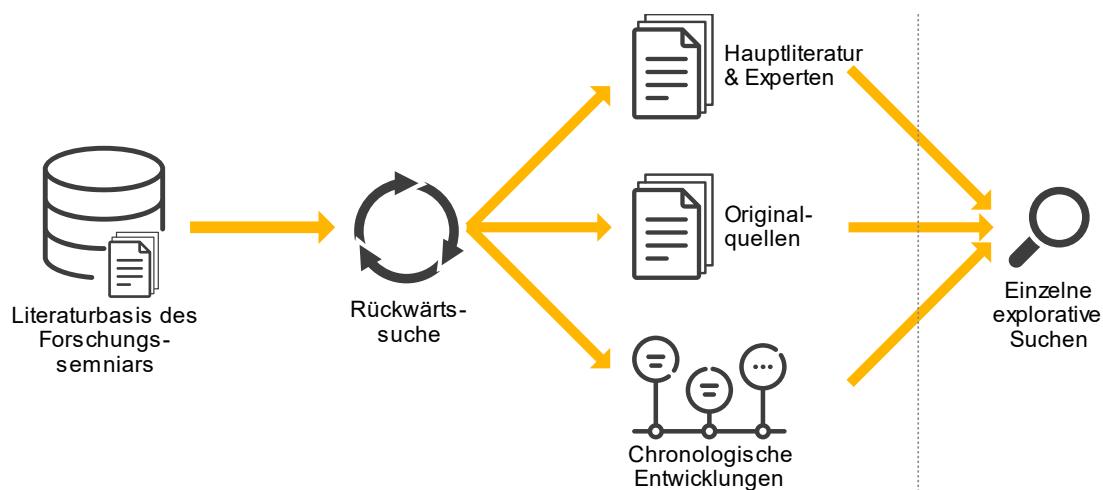


Abbildung 2: Vorgehen zur Expansion der Literaturbasis (Eigene Darstellung)

In Hinblick auf das aktuelle Vorhaben erlaubt diese Methode Originalquellen hinsichtlich differenter Methoden, sowie deren potentielle Erweiterungen in einem zeitlichen Rahmen zu identifizieren. Weiterhin bietet der gefundene zeitliche Ablauf den Vorteil einen Überblick der technischen und wissensbasierten Entwicklung zu erlangen. Neben der zeitlichen Komponente werden so relevante Hauptliteratur sowie Experten in differenten Arealen identifiziert. Aufbauend auf den Ergebnissen dieser Suche werden additional einzelne, explorative Suchen nach identifizierten Konstrukten durchgeführt. Dies dient dazu spezielle Anwendungsfälle von potentiellen Analysemethoden und State-of-the-Art Herangehensweisen zu finden und so die Literaturbasis zu erweitern.

1.3.2 Der Design-Science-Research-Prozess¹

Im Rahmen des aktuellen Forschungsvorhabens sowie in Hinsicht auf die Entwicklung eines Prototyps wird sich an dem Design-Science-Research-Prozess (DSP) nach PEFFERS, 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 Instanziierungen 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, wo hingegen 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 neuartige Design Prinzipien hinsichtlich verschiedener Ebenen der Entwicklung zu identifizieren (vgl. Gregor & Hevner, 2013, S. 341ff. sowie Hevner et al., 2004, S. 78ff.)

PEFFERS ET AL. (2006) definieren in ihrer Forschungsarbeit sechs differente Phasen, die in Abbildung 3 schematisch illustriert werden. Dieser Prozess bildet dabei einen generellen Rahmen für die Forschung im Bereich von Informationssystemen.

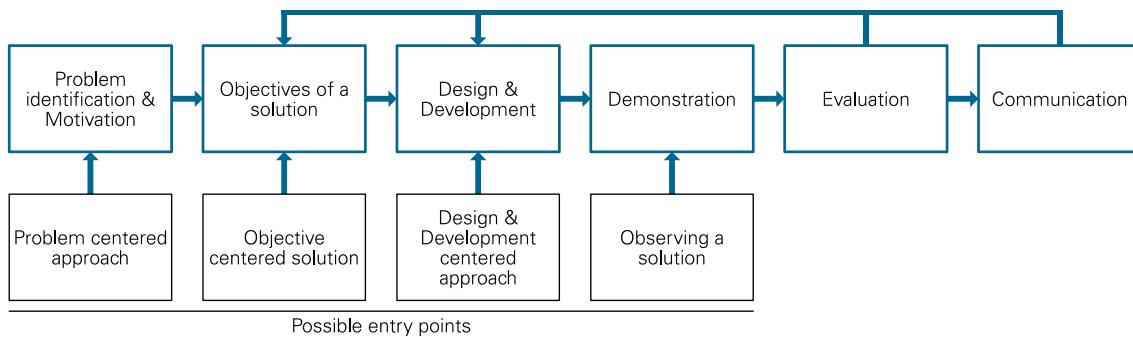


Abbildung 3: Der Design-Science-Research-Prozess nach PEFFERS ET AL. (2006)

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. 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 definiert 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 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 Prototyp eines Recommender-Systems, welcher mithilfe einer adaptierten Variante des Prototyping-Vorgehensmodells nach PRESSMAN & MAXIM (2015) entwickelt wird. Somit findet eine Integration dieses Vorgehensmodell in die aktuelle Phase des DSP statt.

PRESSMAN & MAXIM (2015) definiert in seinem Modell fünf differente Phasen eines Zyklus, welche in ihrer Gesamtheit den Prozess der Softwareentwicklung widerspiegeln

(vgl. Abbildung 4). 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 prototyp-bezogenen Anforderungen erfolgt eine Unterteilung in funktionale sowie nicht-funktionale Anforderungen. Erstere beschreiben im Allgemeinen alle Aktionen, die das System 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 die Akzeptanz des Systems gekoppelt, woraus sich eine essentielle Relevanz dieser ergibt (vgl. Robertson & Robertson, 2012, S. 9f.). 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).

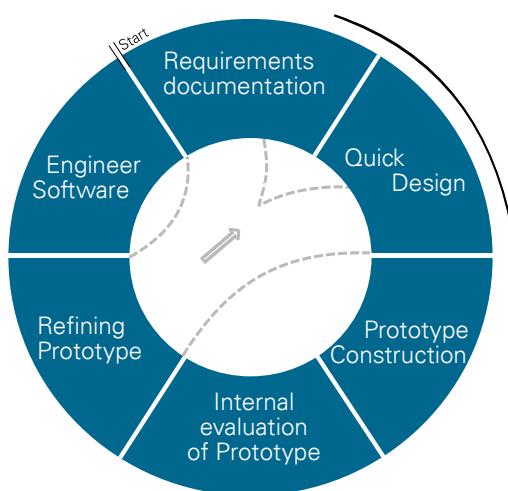


Abbildung 4: Vorgehensmodell des Prototyping in Anlehnung an PRESSMAN & MAXIM (2015)

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 kritiklosem Review die Phase der eigentlichen Systementwicklung, welche das fertige Softwareprodukt in seiner ersten Version als Resultat hervorbringt (vgl. Pressman & Maxim, 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, 2019, S. 9). Die Anwendung dieses Prinzips resultiert vor allem aus den nicht allumfänglich im Vorfeld definierbaren Anforderungen und der dadurch entstehenden Erweiterung bzw. Anpassung von Anforderungen über den Softwareentwicklungsprozess hinweg, durch welche strukturelle Änderungen des Programms auftreten können.

Nach erfolgreicher Implementierung des Artefakts bzw. des Prototyps wird in der nächsten Phase des DSP eine **Demonstration** dessen 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 oben 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 3 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 Arbeit²

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.

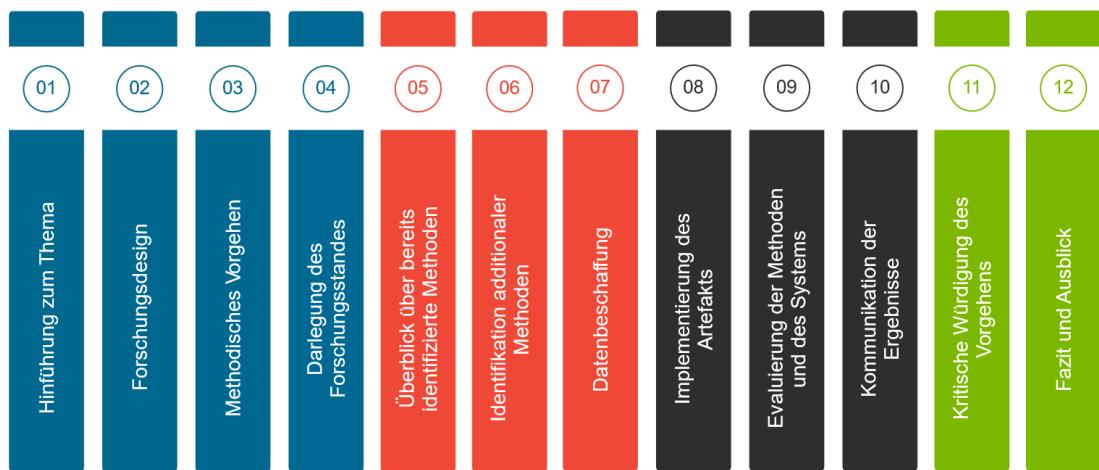


Abbildung 5: Struktureller Aufbau der Arbeit (Eigene Darstellung)

Anschließend werden unter Kapitel 4 weiterführende Methoden aufgezeigt, welche eine Relevanz für die Untersuchung aufweisen. Aufbauend auf diesen Erkenntnissen werden in Kapitel 5, zur Beantwortung der ersten forschungsleitenden Frage, der Prozess zur Datenbeschaffung deskribiert sowie die hierfür verwendeten Methoden eruiert. 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 6 den Fokus auf die Entwicklung des Prototyps, um diesen sowie dessen Methoden darauffolgend in Kapitel 7 zu evaluieren und zu präsentieren. Abschließend wird in den Kapiteln 8, 9 und 10 das ausgewählte Vorgehen kritisch beleuchtet und ein allgemeines Fazit sowie ein Ausblick gegeben.

2 Überblick des Forschungsseminars²

In Einklang mit, der in Kapitel 1 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.

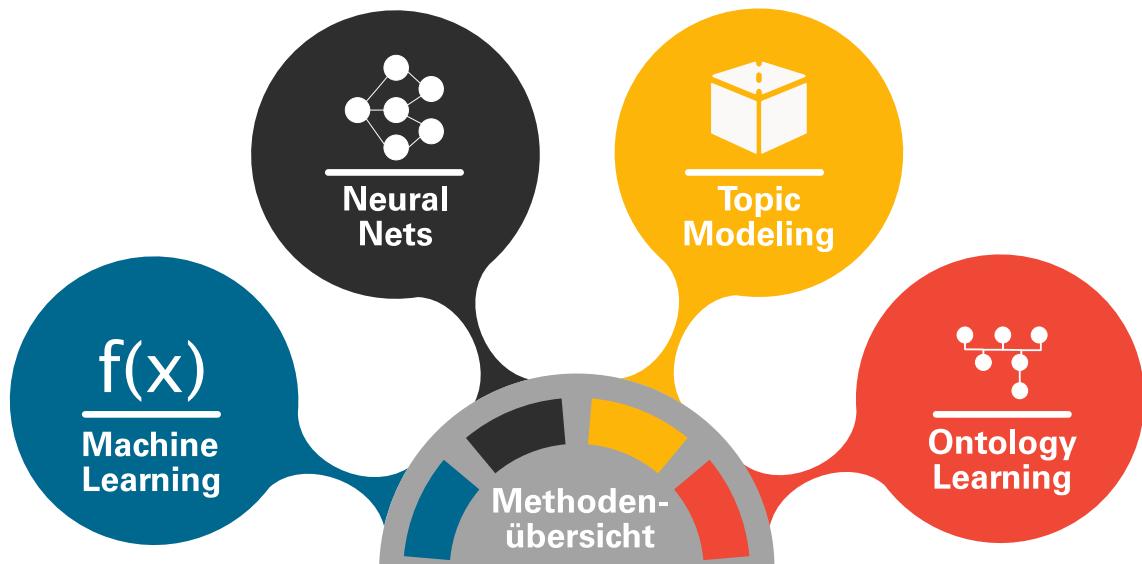


Abbildung 6: Übersicht der methodischen Kategorien (Eigene Darstellung)

Machine Learning

Aufgrund dessen, dass sich die Hauptaufgabe der Zielstellung auf ein Klassifikationsproblem 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, 2018, S. 1). Dabei überführen diese Methoden Textdaten in dichtbesetzte Vektoren und verinnerlichen, beruhend auf der distributionellen Hypothese, die Bedeutung eines Wortes auf Basis des zugrundeliegenden Kontexts. 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 et al., 2018, 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 dichtbesetzte Vektorrepräsentationen einzelner Paragraphen bzw. Dokumente errechnet (vgl. Le & Mikolov, 2014, S. 1188ff.). Jedoch müssen ungesehene Dokumente bzw. Paragraphen neu angelernt 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., 2018, S. 9). Diese Modelle zeigen besonders hinsichtlich angeschlossener 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 widerspiegeln, indirekt als Klassifikationsalgorithmus angewandt werden, da die Topic-Verteilung eines ungesesehenen Dokuments extrahiert und somit auf dessen Klasse geschlossen werden kann.

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 Forschungsstand¹

Wie unter Kapitel 1 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. 31ff. und 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 BRAZDIL & HENERY (1994) zu verweisen. Die Autoren untersuchen hierbei, nach dem unter Kapitel 1.1 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. Brazdil & Henery, 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 Clusteringbäumen nach TODOROVSKI, BLOCKEL & DZEROSKI (2002). Diese werden allerdings aufgrund des Rahmens dieser Arbeit nicht weiter behandelt.

Final wird der Prototyp in Kapitel 1.1 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 an Methoden zur automatisierten Suche nach Schlüsselwörtern existiert und die Qualität der gefunden 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) und YAKE nach CAMPOS ET AL. (2018) anzuführen. In der Subkategorie der

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

graphbasierten Verfahren hingegen ist auf die Algorithmen TextRank nach MIHALCEA & TARAU (2004), 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 ungewöhnliche 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 Topic-Verteilungen erfolgt. Gerade in Hinsicht dieser Klassifikation definiert BISHOP (2006) multiple Methoden des **Machine Learning**, von denen sich einige ebenfalls in modernen Recommender-Systemen

men 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 Programmdaten 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 ebenfalls für diesen Zweck angewendet werden können.

Wie bereits angedeutet, bedienen sich moderne Systeme, welche Klassifikationen jeglicher Art vornehmen, zumeist an 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 zuordnenbaren, 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 sowie Perone et al., 2018, S. 2). Basierend auf der Pionierarbeit von MIKOLOV ET AL. (2013) berechnen einfache neuronale Netze distribuierte Repräsentationen, welche die Semantik über 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., 2018, 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 Ergebnisse als die klassischen Word2Vec-Modelle führen (vgl. Pennington, Socher & Manning, 2014, S. 1532). Additional zeigen BOJANOWSKI ET AL. (2017) 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, 2017, S. 137f.). Zeitgleich wird auf diese Weise die Wahrscheinlichkeit für ungewohnte 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., 2018, S. 1). Als Grundverfahren zur Errechnung von distribuierten Paragraphenreprä-

sentation ist das Bilden des Mittelwerts über verschiedene Wort-Embeddings zu nennen. Jedoch weisen derartige Verfahren Unzulänglichkeiten bzgl. der Inklusion der exakten Semantik längerer Paragraphen auf (vgl. Perone et al., 2018, 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*² 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., 2018, S. 169).

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

4 Additionale Methoden¹

In Anlehnung an die vorangegangene Forschungsarbeit dient das folgende Kapitel zur Identifikation und Deskription additionaler Methoden, welche 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 Embeddings¹

Wie in Abschnitt 3 bereits dargelegt, existiert eine Vielzahl an Methoden, um Wörter distribuiert zu repräsentieren. Die Verwendung von Embeddings hat in vielen nachgelagerten NLP-Anwendungen Verbesserungen erzielt, wie beispielsweise in den Forschungsarealen der Textklassifikation, der Machine Translation oder bei Frage-Antwort-Systemen (vgl. Camacho-Collados & Pilehvar, 2018a, S. 743). Zum einen ist dies darauf zurückzuführen, dass mit dichtbesetzten Vektoren gelernt wird, welche die 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 eine derartige Repräsentation von Paragraphen vergleichsweise unerforscht (vgl. Camacho-Collados & Pilehvar, 2018a, S. 743f.).

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 mit ein (vgl. Iyyer, Manjunatha, Boyd-Graber & Daumé,

2015, 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., 2018, S. 9). Nichtsdestotrotz benötigen diese Methoden mehr Trainingszeit und -daten, um signifikante Ergebnisse zu erzielen (vgl. Iyyer et al., 2015, S. 1681).

Im Folgenden werden ausgewählte Methoden für die Erstellung von distribuierten Repräsentationen deskribiert, welche eine Eignung für das vorliegende Forschungsvorhaben aufweisen und daher Anwendung finden.

4.1.1 FastText¹

Wie zuvor erwähnt, existieren im Bereich der Wort- bzw. Paragraph-Embeddings zahlreiche Ansätze, Methoden und Architekturen. Eine dieser Architekturen wird von BOJANOWSKI ET AL. (2017) publiziert. Die Autoren beschreiben das Modell FastText, welches als Weiterentwicklung aus dem bereits deskribierten Modell Word2Vec von MIKOLOV ET AL. (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 7 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 et al., 2013, S. 3ff.).

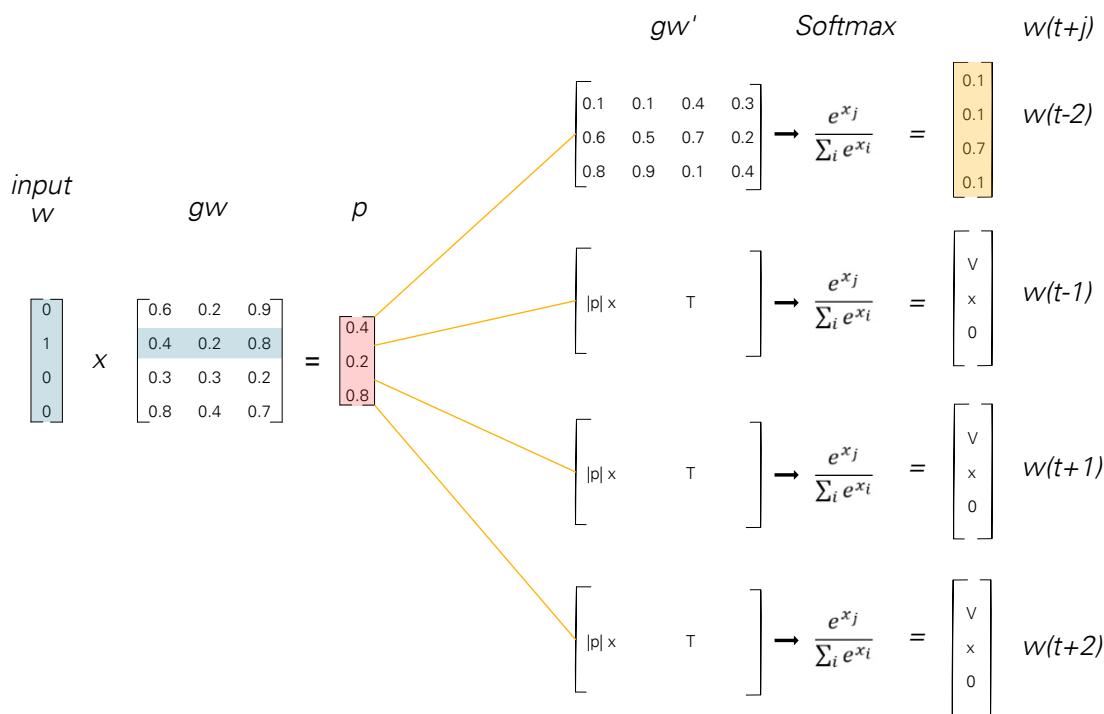


Abbildung 7: Architektur des Word2Vec-Modells nach MIKOLOV ET AL. (2013)

BOJANOWSKI ET AL. (2017) transferieren diese Methodik auf Buchstaben-N-Gramme, trainieren 100-dimensionale Vektoren für sämtliche dieser im Korpus enthaltenen N-Gramme und beachten hierdurch die interne Struktur von Wörtern. Ein Wortvektor wird daraufhin mithilfe einer ungeordneten Kompositionsfunktion 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 Trainingsausrichtung 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 Networks²

Deep Averaging Networks verwenden eine ungeordnete Kompositionsfunktion, welche auf Basis bestehender Wort-Embeddings gleichdimensionale Paragraphvektoren berechnet. Im Zusammenspiel mit einem bereits trainierten Word-Embedding-Modell erreicht die Architektur in nachgelagerten NLP-Anwendungen State-of-the-Art-Performance (vgl. Iyyer et al., 2015, S. 1681ff.). Das grundlegende Prinzip von DANs ist es eine arbiträre Anzahl von Wort-Embeddings zu mitteln und daraufhin mit einem neuronalen Netz die Paragraph-Embeddings zu berechnen (vgl. Iyyer et al., 2015, S. 1685).

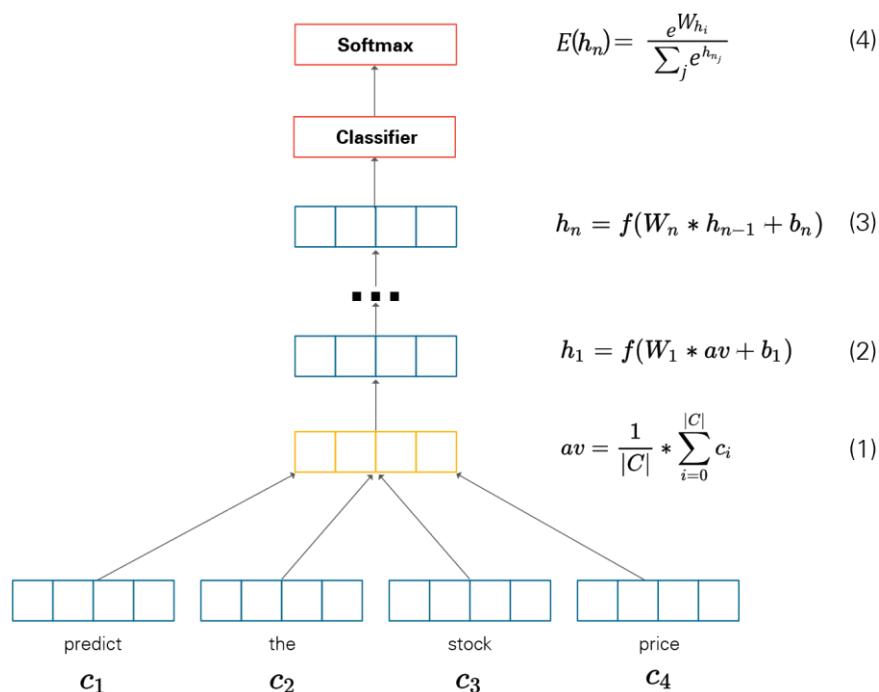


Abbildung 8: Funktionsweise eines DANs in Anlehnung an IYYER ET AL. (2018)

Wie in Abbildung 8 veranschaulicht, funktioniert das Modell in drei grundlegenden Schritten. Zunächst wird der Mittelwert der entsprechenden, distribuierten Wortrepräsentationen gebildet, um multiple Wort-Embeddings in einen einzigen Vektor zu überführen (1). C beschreibt dabei die Menge der Wörter des Paragraphen und c_i die jeweiligen Elemente der Menge. Das Resultat ist ein Vektor derselben Dimensionalität, wie die der jeweilig eingehenden Wortvektoren. Daraufhin wird der gemittelte Vektor durch n verschiedene Hidden-Layer geführt (2)(3). Dabei stellt W_n die Gewichte des entsprechenden Hidden-Layer h_n sowie b_n den entsprechenden Bias dar. Die Anzahl der Hidden-Layer sowie die zugrundeliegende Aktivierungsfunktion $f(x)$ sind dabei dem Modell lieber überlassen. Der Output des letzten Hidden-Layers bildet final den distribuierten Paragraphvektor. Allerdings setzt dies erst ein Training des Netzes voraus. Dafür schließt sich den grundlegenden Hidden-Layern ein Classifier (z.Dt.: Klassifikator) 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 bedient, welche eine Wahrscheinlichkeitsverteilung über die Menge der Zielklassen wiedergibt (4).

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 zur Anpassung der Gewichte des DAN dient (vgl. Iyyer et al., 2015, S. 1681). Dabei kann das Modell 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., 2015, S. 1685f.).

Die Intention eines DAN ist es, dass jede der verschiedenen Netzsichten eine immer abstraktere Repräsentation des Inputs berechnet. Dadurch werden kleine, aber ausschlaggebende, semantische Unterschiede stärker hervorgehoben und in den resultierenden Vektor inkludiert. In der untenstehenden Abbildung 9 werden vier verschiedene, dimensionsreduzierte Paragraphvektoren illustriert. Das rechte Schaubild zeigt Paragraphvektoren, die auf Basis des arithmetischen Mittels einzelner Wortvektoren erzeugt werden. Das linke Bild hingegen veranschaulicht Paragraphvektoren, die mittels der eben deskribierten Architektur generiert sind. Als Eingabevektoren werden hierbei Fast-Text-Embeddings verwendet.

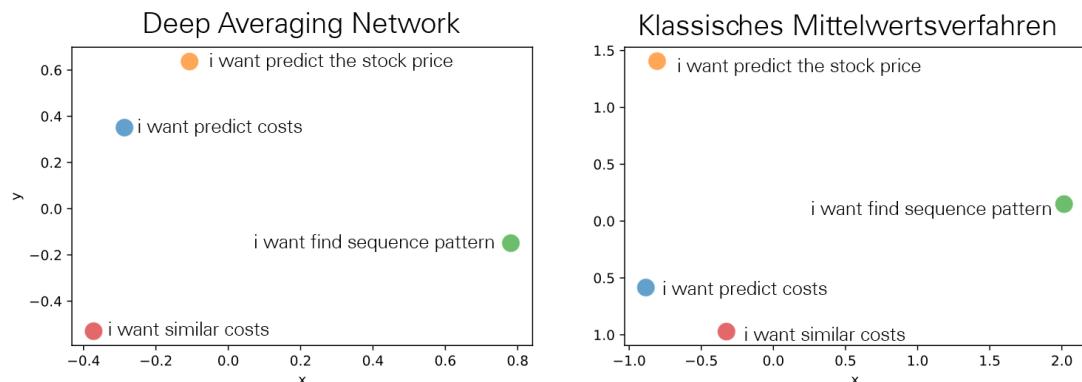


Abbildung 9: Gegenüberstellung DAN und Mittelwertsverfahren (Eigene Darstellung)

In Anlehnung an die in Kapitel 1 definierte Zielstellung sind vier verschiedene Sätze der 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 diese Paragraphen auf zwei unterschiedliche Klassen hindeuten, sind beide bei dem klassischen Mittelwertverfahren sehr dicht beieinander, wodurch eine große Ähnlichkeit impliziert wird. Das Abstrahieren dieses Mittelwerts durch multiple Transformationen mit aufeinander folgenden Hidden-Layern grenzt die semantischen Unterschiede besser voneinander ab. Hierbei liegen Paragraphen einer Klasse näher aneinander.

IVYER ET AL. (2018) heben zusätzlich hervor, dass DANs besonders bei „unscharfen“ Daten funktionieren und domänenübergreifend State-of-the-Art-Ergebnisse erzielen. So eignen sich diese Modelle für 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 vorliegenden Forschungsarbeit.

4.1.3 Universal Sentence Encoder²

Die Universal Sentence Encoder (USE) sind zwei von Google propagierte Modelle der distributionellen Semantik, welche sich insbesondere auf das Gebiet des Transfer Learning spezialisieren. In Anbetracht der Effizienz und Performance in nachgelagerten NLP-Anwendungen erzielen beide Modelle State-of-the-Art-Ergebnisse. 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 diese Modelle auf Basis geringer Datenmengen vergleichbare Ergebnisse erzielen wie die bereits im Forschungsstand erwähnten Modelle der distributionellen Semantik. Zu betonen ist, dass die USE-Modelle besonders effektiv semantische Nuancen der Paragraphen inkludieren und somit State-of-the-Art-Ergebnisse im Erkennen tex-tueller 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. 174f.). Die beiden von Google bereitgestellten Modelle kreieren dafür Vektoren mit jeweils 512 Dimensionen.

Die beiden von CER ET AL. (2018) etablierten Modelle bieten darüber hinaus die Möglichkeit zwischen Effizienz und Modellgenauigkeit abzuwägen. Zum einen basiert das leichtgewichtige Modell auf der von IYYER ET AL. (2015) beschriebenen DAN-Architektur und verspricht ein performantes Training sowie schnelle Inferenz. Das trainierte DAN-Modell ist dabei auf Wörtern sowie N-Grammen 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 spezielle Attention-Funktionen nutzt (vgl. Vaswani et al., 2017, S. 5999ff.). Für die final publizierten USE-Modelle verwenden CER ET AL. (2018) ein trainiertes Word2Vec-Modell. Im Rahmen des aktuellen Forschungsvorhabens wird das auf der DAN-Architektur³ fundierende Modell aufgegriffen und mit den anderen vorgestellten Methoden verglichen. Im Folgenden wird hierbei mit der Begrifflichkeit USE auf das DAN basierte Modell referenziert, da dieses eine schnellere Inferenz von Vektoren verspricht und nach IYYER ET AL. (2015) syntaktischen Divergenzen entgegenwirkt. Hierbei wird auf eine umfassende Anpassung der Parameter sowie auf eine Modellevaluation verzichtet. Für eine ausführliche Evaluation sei auf PERONE ET AL. (2018) verwiesen, die eine intrinsische sowie extrinsische Evaluation der Modelle durchführen.

³ Eine Beschreibung zum originalen Modell findet sich unter folgenden Link:
<https://tfhub.dev/google/universal-sentence-encoder-lite/2> [Abgerufen am: 23.03.2019]

4.2 RNN-basierte Technologien¹

Weiterhin existieren im Kontext des Deep Learning Architekturen, welche aufgrund ihrer Fähigkeit zur Verarbeitung von sequenziellen Daten im Bereich des NLP starke Anwendung finden. Generell weist ein Wort hinsichtlich seiner Bedeutung eine Abhängigkeit zu vorangegangenen Wörtern auf. Exemplarisch sind hier die englischen Wörter „dog“ sowie „hot dog“ anzuführen, in welchen es eine starke Abweichung des Wortsinnes „dog“ durch das vorausgehende Wort „hot“ gibt. Mit rekurrenten Netzwerken kann die sequenzielle Struktur von Sprache und somit kontextuelle Abhängigkeiten zwischen Wörtern einbezogen werden, indem Buchstaben, Wörter oder Sätze nacheinander in das Netzwerk eingelesen werden. Der zunehmende Fokus des NLP-Bereichs auf RNN-basierte Technologien spiegeln diese Vorteile wider (vgl. Young, Hazarika, Poria & Cambria, 2018, S. 65f.).

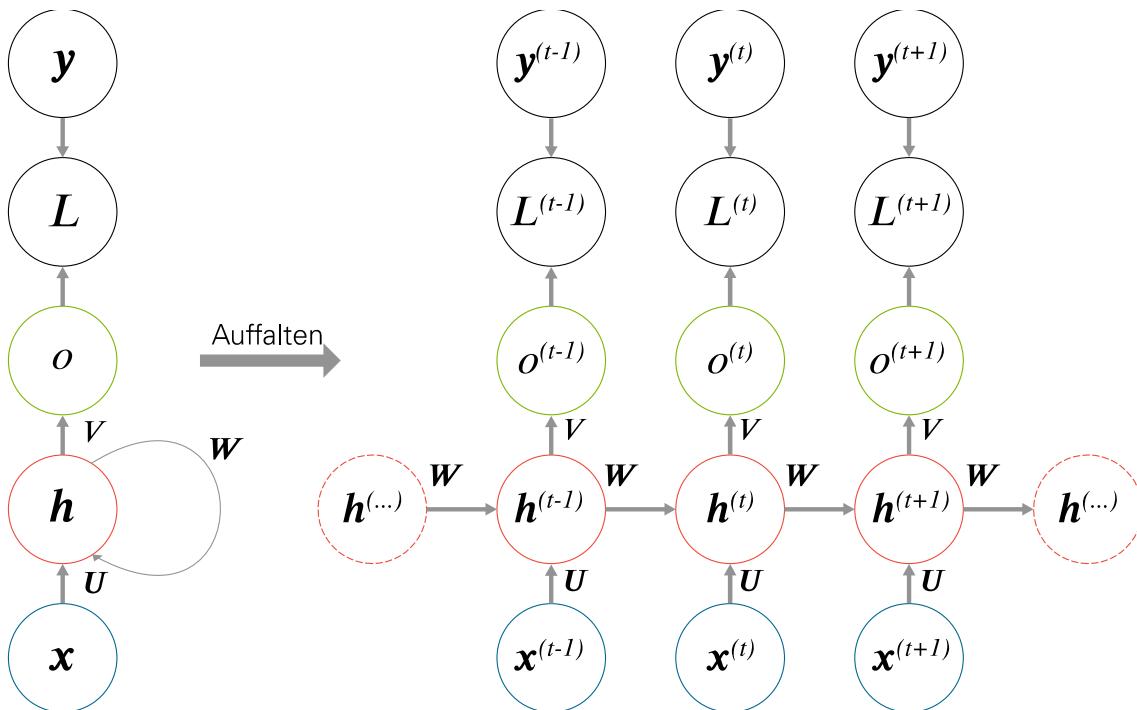


Abbildung 10: Berechnungsgraph eines Recurrent Neural Networks nach GOODFELLOW ET AL. (2016)

In Referenz zur vorherigen Forschungsarbeit wird in Abbildung 10 diese Architektur veranschaulicht, wobei x die Inputsequenz der Länge τ darstellt, die 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 $h^{(t)}$ eingeht. So entsteht der Input pro Zeitstempel $a^{(t)}$ sowie dessen Verarbeitung 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 der 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. Analog zur rechten Hälfte der Abbildung 10 wird hierfür das Netzwerk 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. der damit verbundenen, tieferen Architektur des Netzwerkes, da der berechnete Wert zur Anpassung der Gewichte, in Relation zu jener Tiefe exponentiell steigt 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 impliziert ein instabiles 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. Gleichzeitig geht damit eine Unsicherheit, in welche Richtung die Gewichtsanpassung zur Optimierung der Kostenfunktion erfolgen muss, einher (vgl. Goodfellow et al., 2016, S. 289f. sowie Sussillo & Abbott, 2014, S. 1f.).

4.2.1 Long Short-Term Memory¹

Zur Adressierung der eben erläuterten Problematiken publizieren HOCHREITER & SCHMIDHUBER (1997) die Architektur des LSTM in seiner ersten Version. Diese findet in den Folgejahren hohen Anklang und wird sukzessiv durch andere Publikationen, wie beispielsweise von GERS, SCHMIDHUBER & CUMMINS (2000) oder BAYER, WIERSTRA, TOGElius & SCHMIDHUBER (2009), angepasst und optimiert, woraus letztendlich die heutige Version des LSTM resultiert. Generell ersetzen die Autoren in jeder Zelle die singuläre Tangens-Funktion eines klassischen RNNs (vgl. Formel 2) durch mehrere, komplexere Operationen und wirken so einer potentiellen Instabilität bzw. Stagnation des Trainings bei RNNs entgegen. Abbildung 11 gibt hierfür eine ablauforientierte Illustration einer LSTM-Zelle.

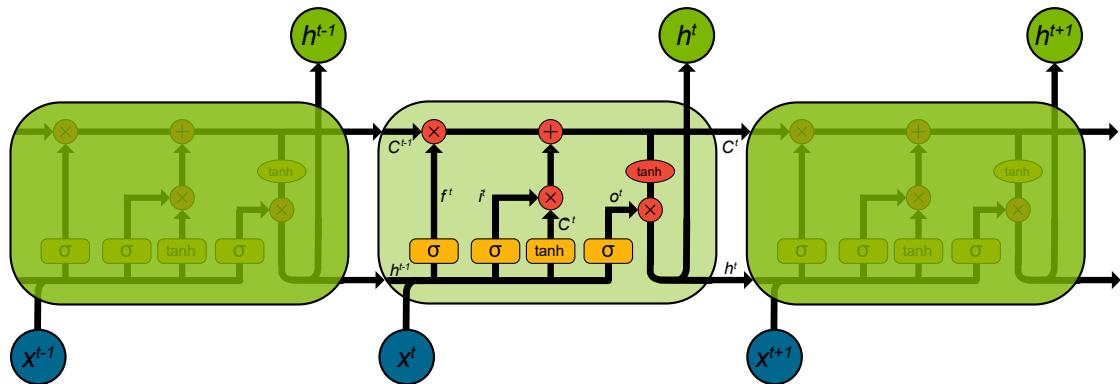


Abbildung 11: Illustration der Funktionsweise einer LSTM-Zelle (Eigene Darstellung nach COLAH (2015))

Auffällig in dieser Architektur ist primär der sog. *Cell State* (CS) (z.Dt. 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 das sog. Forget Gate anzusprechen, das durch GERS ET AL. (2000) zur Basisversion des LSTM hinzugefügt wird. Dieses Gate, charakterisiert durch die erste Sigmoid-Funktion sowie die darauffolgende Multiplikation mit dem CS, legt fest, welche bisher bereits gespeicherten Informationen im CS gelöscht und somit „vergessen“ werden (vgl. Abbildung 11).

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

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 3). 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 die Verwendung der Informationen des neuen Inputs und somit deren Speicherung im CS regelt (vgl. Formeln 4, 5, 6). Hierbei ist das Gate zweigeteilt und berechnet zunächst, welche Informationen des CS generell aktualisiert werden (i^t) und multipliziert diese Informationen mit potentiellen Kandidaten \tilde{C}^t , die mittels einer Tangens-Hyperbolikus-Funktion aus dem Input berechnet werden.

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

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

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

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 an den Formeln zu erkennen, erfolgt die Evaluation, welche Informationen ausgegeben werden, analog zum Input Gate. Die finale punktweise Tangens-Hyperbolikus-Operation findet statt, um die Werte des CS im Intervall $[-1, 1]$ zu skalieren. Schließlich ist additional auf den Vorteil der Variabilität derartiger, auf RNN basierenden, Architekturen zu verweisen. In Abbildung 10 und 11 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 Textklassifikation oder zur Vorhersage eines Wertes auf Basis einer Zeitreihe Anwendung. Analog sind derartige Modelle ebenfalls als One-

to-One- bzw. One-to-Many-Architekturen realisierbar. Im Rahmen der aktuellen Untersuchung sind diese Modelle vor allem in Form von Many-to-One- und One-to-One-Architekturen zu analysieren, 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 der zweiten Variante beispielsweise durch ein Satz- bzw. Paragraph-Embedding realisiert werden kann (vgl. Goodfellow et al., 2016, S. 409ff. sowie Hochreiter & Schmidhuber, 1997, S. 1746ff. als auch Young et al., 2018, S. 66).

4.2.2 Gated Recurrent Units²

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 konkatenieren (vgl. Chung, Gülcöhre, Cho & Bengio, 2014, S. 4). Die nachfolgende Abbildung illustriert zunächst die grundlegende Funktionsweise einer solchen Architektur.

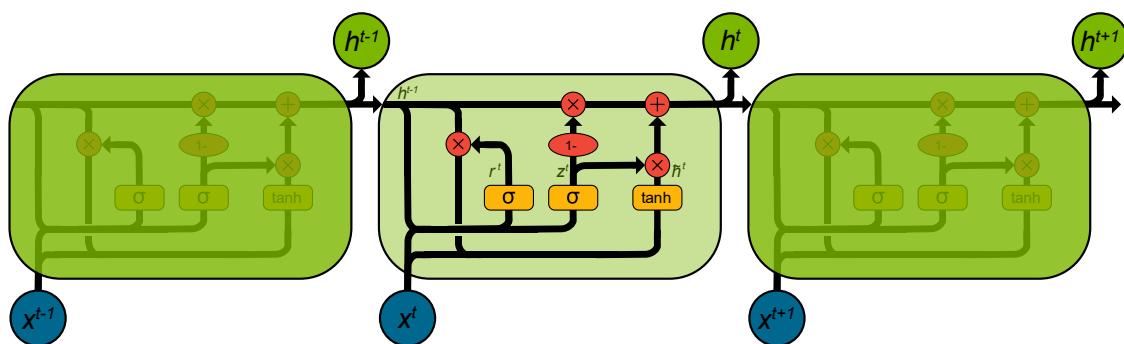


Abbildung 12: Illustration der Funktionsweise einer GRU-Zelle (Eigene Darstellung nach COLAH (2015))

Im Vergleich zu LSTM-Zellen wird bei dieser Architektur auf zwei verschiedene Zellstatus verzichtet, stattdessen wird der CS zusammen mit dem Hidden-State zu einem Zellstatus h_t kombiniert, welcher den Informationsfluss über die einzelnen Zellen hinweg abbildet. Somit besitzen diese Zellen keine Kontrolle darüber, in welchem Maße die Zellinformationen weitergeben werden (vgl. Chung et al., 2014, S. 4).

Hinzukommt, dass GRU-Zellen nicht zwei separate Input- und Forget-Gates aufweisen, sondern diese in einem sog. Update-Gate z_t vereinen. Dabei bestimmt das Update-Gate

die Menge an Informationen, die vom Hidden-State h_t aktualisiert werden. Hierbei wird der Sigmoiden-Funktion σ das Produkt der Gewichtsmatrix W_z und den konkatenierten Vektoren $[h_{t-1}, x_t]$ zugeführt. Anschließend wird durch $(1 - z_t) * h_{t-1}$ definiert, welche Informationen im Hidden-State $h_t - 1$ aktualisiert werden.

$$z_t = \sigma(W_z * [h_{t-1}, x_t]) \quad (9)$$

Neu hingegen ist die Einführung eines Reset-Gates. Dieses 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.

$$r_t = \sigma(W_r * [h_{t-1}, x_t]) \quad (10)$$

Der finale Zellstatus bzw. der neue Hidden-State h_t wird somit durch das Zwischenergebnis \tilde{h}_t und dem Hidden-State h_{t-1} mittels Addition der Vektoren bestimmt.

$$\tilde{h}_t = \tanh(W * [r_t * h_{t-1}, x_t]) \quad (11)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad (12)$$

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 und muss im Rahmen dieser untersucht werden (vgl. Chung et al., 2014, S. 6f.).

4.3 Keyword-Extraction-Methoden²

Im folgenden Kapitel werden ausgewählte Modelle der Keyword Extraction vorgestellt und erläutert. Es erfolgt demnach eine Erweiterung der bereits in der vorangegangenen Forschungsarbeit aufgezeigten Methoden *Term Frequency - Inverse Document Frequency* (TFIDF), *Term Frequency - Inverse Gravity Moment* (TFIGM) und TextRank. Aufbauend hierauf lässt sich in aktueller Literatur eine Vielzahl an Keyword-Extraction-Algorithmen identifizieren. Die Auswahl der Algorithmen orientiert sich hierbei an den in der Literatur dargelegten Ergebnissen der jeweiligen Algorithmen und konzentriert sich, in

Kombination mit den bereits aufgezeigten Keyword-Extraction-Methoden, auf jeweils drei Algorithmen aus den Subkategorien der gewichtungsbasierten sowie graphbasierten Verfahren.

4.3.1 YAKE²

CAMPOS ET AL. (2018) publizieren den Algorithmus YAKE und definieren fünf divergente Eigenschaften eines Wortes, die zu dessen Bewertung herangezogen werden. Hierbei handelt es sich um statistische Größen, welche auf Basis der übergebenen Dokumente berechnet werden können, wodurch das Verfahren in die Kategorie der unüberwachten Keyword-Extraction-Methoden einzuordnen ist (vgl. Campos et al., 2018, S. 684).

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

$$W_{case} = \frac{\max(TF(U(w)), TF(A(w)))}{\log_2(TF(w))} \quad (13)$$

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.

$$W_{pos} = \log_2(\log_2(2 + Median(Sen_w))) \quad (14)$$

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 durchschnitt-

lichen Häufigkeit von Wörtern, addiert mit der Standardabweichung dieser. Weiterführend berechnen die Autoren, inwieweit das betrachtete Wort die **Charakteristika eines Stopwords** erfüllt und demnach weniger Relevanz aufweist.

$$W_{rel} = \left(0.5 + \left(\left(WL * \frac{TF(w)}{MaxTF} \right) + PL \right) \right) + \left(0.5 + \left(\left(WR * \frac{TF(w)}{MaxTF} \right) + PR \right) \right) \quad (15)$$

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 variierendem 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 $W_{DifSentence}$. Final führt dies zur Berechnung der Kandidatenbewertung mittels der untenstehenden Formeln, wobei $S(w)$ für die Bewertung eines Wortes sowie $S(kw)$ zur Bewertung von N-Gramm-Kandidaten herangezogen wird.

$$S(w) = \frac{W_{rel} * W_{pos}}{W_{case} + \frac{W_{freq}}{W_{rel}} + \frac{W_{DifSentence}}{W_{rel}}} \quad (16)$$

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

Abschließend zeigen CAMPOS ET AL. (2018) in einem Vergleich dieser Methodik mit TFIDF und TextRank starke Verbesserungen im Rahmen der Keyword Extraction.

4.3.2 PositionRank¹

In Addition zur graphbasierten Methode des TextRank nach MIHALCEA & TARAU (2004), welches in vorangegangener Arbeit deskribiert worden ist, wird der Algorithmus PositionRank nach FLORESCU & CARAGEA (2017) erläutert. Analog zu TextRank basiert der Algorithmus auf dem PageRank-Modell von PAGE, BRIN, MOTWANI & WINOGRAD (1999). Dabei liegt der allgemeine Vorteil von graphbasierten Verfahren bei der namensgebenden

Verwendung von Graphen zur Repräsentation der Dokumentstruktur, da die Gewichtung der verschiedenen Knoten eines Graphs dessen Relevanz im gesamten Netzwerk widerspiegelt (vgl. Mihalcea & Tarau, 2004, 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 beurteilt, wobei die Summe dieser Bewertungen 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 Inklusion 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 besitzen. Zur Realisierung dieser Intention beziehen die Autoren alle inversen Positionen eines Wortes im Dokument mit in die Rangbewertung nach PageRank bzw. TextRank⁴ ein. Hierbei berechnet sich z.B. der Wert der Positionsbewertung p eines Wortes an der zweiten, fünften und zehnten Position im Dokument wie folgt.

$$p = \frac{1}{2} + \frac{1}{5} + \frac{1}{10} = 0.8 \quad (18)$$

Das Aufsummieren der einzelnen Werte dient additional zur 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 TopicRank²

Als ein weiterer Vertreter graphbasierter Verfahren wird TopicRank herangezogen, welches den Graphen anhand von übergeordneten Themen (Topics) aufbaut anstelle von

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

einzelnen Wörtern. Ähnlich zum TextRank-Algorithmus besteht die wesentliche Intention darin, einen Graphen aus den zugrundeliegenden Dokumenten zu erstellen, welcher die semantisch wichtigsten Textbausteine widerspiegelt. Im Kontrast zu klassischen Verfahren der Keyword Extraction baut TopicRank den Graphen an zuvor erarbeiteten Topics auf, die durch ein spezifisches Clusteringverfahren erzeugt werden. Hierbei sind die Topics nicht durch einzelne Wörter des Korpus definiert, sondern durch die im Text enthaltenen, potentiellen Schlüsselwortphrasen (vgl. Bougouin et al., 2013, S. 543ff.). Das Verfahren ist grundsätzlich als eine Erweiterung des TextRank-Algorithmus anzusehen, das signifikante Verbesserungen im Bereich der Keyword Extraction aufzeigt (vgl. Bougouin et al., 2013, S. 548f.). In der folgenden Abbildung werden die grundlegenden Schritte des Algorithmus nach BOUGOUIN ET AL. (2013) erläutert, wonach auf die detaillierte Berechnung des Graphen eingegangen wird.

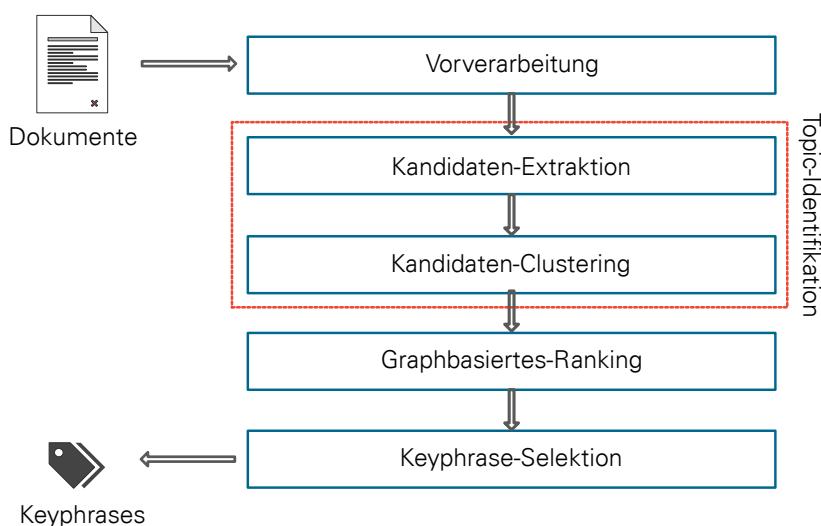


Abbildung 13: Verarbeitungsschritte des TopicRank-Algorithmus (Eigene Darstellung)

Im Vorfeld der eigentlichen Extraktion der relevanten Topics wird der Textkorpus einer Datenvorverarbeitung unterzogen (Tokenization, Part-of-Speech-Tagging (POS)). Daraufhin erfolgt die Topic-Selektion, welche die längsten Nominalphrasen identifiziert und

diese als potentielle Topic-Kandidaten behandelt⁵. Die automatisierte Gruppierung ähnlicher Schlüsselwortphrasen basiert hierbei auf einem agglomerativen Clusteringverfahren mit dem Average-Linkage-Verfahren als Fusionierungsalgorithmus.

Basierend auf den 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 beschreibt. Hierbei repräsentieren Topics die einzelnen Knoten des Graphs und die Kanten zwischen Topics t_i sowie t_j symbolisieren ihren semantischen Zusammenhang. Dabei haben zwei Knoten eine starke, semantische Beziehung, wenn die jeweiligen Topics dicht beieinander im Dokument liegen. So resultiert die Distanz $dist(c_i, c_j)$ aus den Abständen der potentiellen Schlüsselwortphrasen p eines Topics c_i und c_j im zugrundeliegenden Text. Weiterhin wird die Gewichtung einer Beziehung zwischen zwei Knoten t_i und t_j mit dem Gewicht w_{ij} symbolisiert.

$$dist(c_i, c_j) = \sum_{c_i \in t_i} \sum_{c_j \in t_j} \frac{1}{|p_i - p_j|} \quad (19)$$

$$w_{i,j} = \sum_{c_i \in t_i} \sum_{c_j \in t_j} dist(c_i, c_j) \quad (20)$$

Konträr zum klassischen TextRank-Algorithmus handelt es sich folglich, um einen vollvernetzten Graphen, welcher einen besseren Gesamtüberblick über die Topic-Relationen liefert. Das Berechnen der relevantesten Schlüsselwortphrasen des Graphen erfolgt hingegen nach dem Bewertungsverfahren des TextRank-Algorithmus:

$$S(t_i) = (1 - d) + d * \sum_{t_j \in V_i} \frac{w_{j,i} * S(t_j)}{\sum_{t_k \in V_j} w_{j,k}} \quad (21)$$

wobei hier jedoch die Input- und Output-Beziehungen keine Rolle spielen, da es sich um einen ungerichteten Graphen handelt. Abschließend repräsentiert d den sog. Dumping-Faktor, der die Wahrscheinlichkeit angibt von einem Knoten zum nächsten zu wandern. Der Standardwert dieses Faktors liegt bei 0.85 (vgl. Mihalcea & Tarau, 2004, S. 404).

⁵ HULTH (2003) konstatiert, dass der Großteil an Schlüsselwortphrasen durch Nominalphrasen beschrieben werden kann. Daher werden bei TopicRank lediglich diese betrachtet, um so zu große Topics zu vermeiden.

4.4 Ensemble Learning¹

Das Gebiet des Ensemble Learning (EL) ist als ein zentrales Forschungsareal des Machine Learning zu charakterisieren und untersucht Möglichkeiten differente Algorithmen des ML zu kombinieren, um bessere Resultate zu erzielen. Hierbei liegt der zentrale Fokus 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 und generative Methoden zu unterteilen. Nicht-generative Ansätze umfassen hierbei diejenigen Methoden, die 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 Trainingsdaten, eine Erhöhung der Genauigkeit forcieren (vgl. Goodfellow et al., 2016, S. 256f. und Re & Valentini, 2012, S. 567, 572).

Unter dem Gesichtspunkt der Anzahl an unterschiedlichen Methoden, welche zur Einordnung von Problembeschreibungen im Kontext der aktuellen Untersuchung verwendet werden können, bietet das EL einen Ansatzpunkt zu deren Kombination. Gerade hinsichtlich der stark differierenden Herangehensweisen zur Zielerreichung und der geringen Datenmenge kristallisiert sich dieses Areal als vielversprechend heraus. Aufgrund dessen dient das folgende Kapitel zur Deskription von ausgewählten Methoden des EL. Hierbei wird auf nicht-generative Methoden sowie speziell auf die generativen Methoden Bagging und Boosting eingegangen. Dies stellt allerdings nicht die Gesamtheit der EL-Methoden dar. Für eine umfangreichere Auflistung und Beschreibung derartiger Methoden ist auf die Publikation von RE & VALENTINI (2012) zu verweisen, die eine generelle Übersicht bereitstellen.

4.4.1 Nicht-generative Methoden des Ensemble Learnings¹

Die nicht-generativen Methoden des EL lassen sich weiterhin in Fusions- und Selektionsmethoden untergliedern. Im Folgenden werden primär erstere fokussiert, da Selektionsmethoden 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, da sich stark unterscheidende Methoden verwendet werden.

Zunächst ist das Prinzip der **Majority Vote** anzuführen, welches auf Basis aller Vorhersagen der Classifier das am häufigsten prognostizierte Resultat 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 weitverbreiteter Ansatz ist auf das Verfahren des **Weighted majority voting** zu verweisen. Motiviert durch different ausgeprägte Gütwerte der Classifier werden deren Vorhersagen hierbei mit Gewichten belegt, um diese in die Berechnung der finalen Vorhersage einfließen zu lassen. Häufig wird hierfür 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 andere Methoden, die das Training eines additionalen Classifiers auf Basis der Ergebnisdaten fokussieren. Hierzu können multiple ML-Klassifikationsalgorithmen verwendet werden, was aufgrund der geringen Datenmenge im aktuellen Forschungsvorhaben allerdings vernachlässigt wird.

4.4.2 Generative Methoden des Ensemble Learnings²

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

Bagging (Bootstrap Aggregating)

Wie für die Kategorie der generativen Methoden charakteristisch, 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 14 dient zur Illustration des Bagging-Ansatzes.

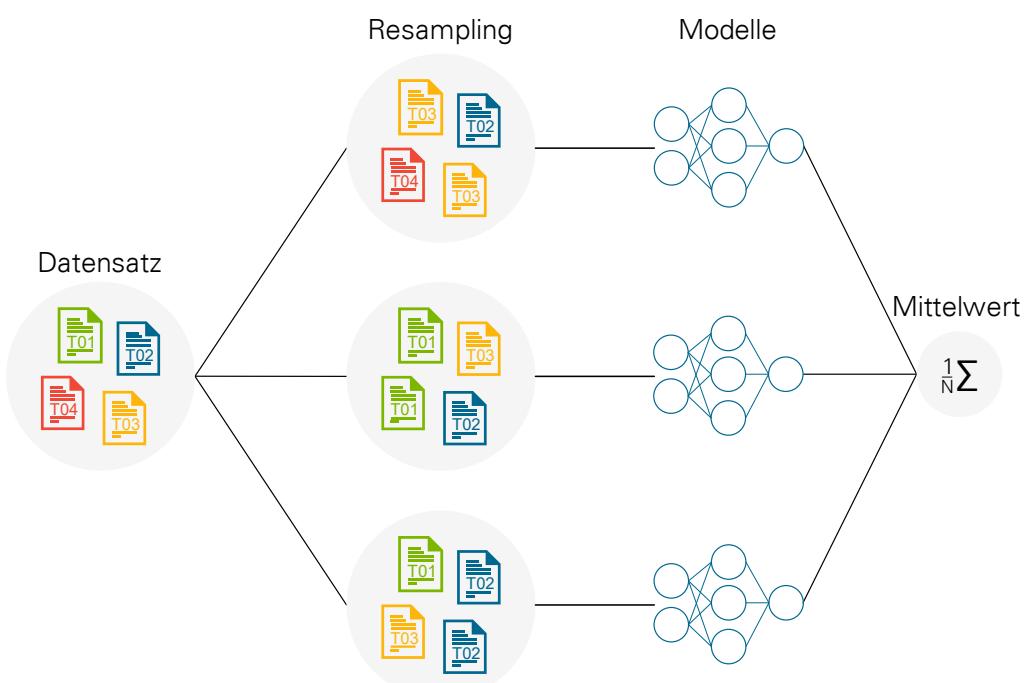


Abbildung 14: Illustration der Funktionsweise des Bagging-Ansatzes (Eigene Darstellung)

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 verschiedene Elemente mehrfach in den resultierenden Datensatz integriert. Im aktuellen Fall werden hierbei andere Elemente durch diese Kopien ersetzt, sodass die Erstellung unterschiedlicher Datensätze identischer Größe möglich ist. Diese Variation der Daten verfolgt das Ziel die oben angesprochene Unabhängigkeit der Fehler und somit unterschiedliche Modelle als Resultat der Trainingsphase zu erreichen. Im 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 und Polikar, 2012, S. 11f.).

Boosting

Im Kontrast zu Bagging-Methoden werden bei einer auf Boosting basierenden Strategie verschiedene, schwache Klassifikatoren zu einem starken Klassifikator zusammengeführt. Meist sind diese schwachen Classifier einfach aufgebaut und berücksichtigen nur ein Merkmal des zugrundeliegenden Datenpools (vgl. Polikar, 2012, S. 35). Hierbei handelt es sich in der Regel um ein sequentielles Vorgehen, um die multiplen schwachen Klassifikatoren zu kombinieren.

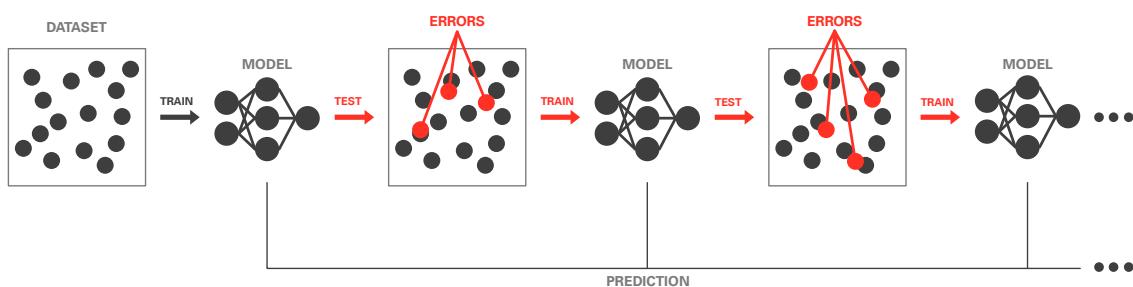


Abbildung 15: Darstellung des Vorgehens bei Boosting (Eigene Darstellung)

Wie in Abbildung 15 illustriert, wird ein initiales Model auf dem Trainingsdatensatz trainiert und getestet. Ausgehend von den falschen Vorhersagen sowie additionalen, zufälligen Datenpunkten aus dem ursprünglichen Datensatz wird das darauffolgende Model trainiert. Somit werden Modelle erstellt, welche speziell auf den Fehlern des Vorgängermodells fundieren.

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. Campos, Canuto, Salles, de Sá & Gonçalves, 2017, S. 114). Allerdings benötigen diese ähnlich viele Daten wie Bagging-Methoden, um ausreichend schwache Klassifikatoren auf den Fehlern der Vorgängermodelle zu trainieren. Hinzukommt, dass diese Methode dazu tendiert zu sehr auf die Trainingsdaten angepasst zu sein und dadurch stark auf Ausreißer zu reagieren, weshalb das Finden der adäquaten Parameterkombination hierbei essentiell ist (vgl. Meshram & Shinde, 2015, S. 7). Aufgrund der fehlenden Datenbasis und der hohen Divergenz an verwendeten Herangehensweisen im aktuellen Forschungsvorhaben werden für die Implementierung des Prototyps jedoch keine generativen Methoden verwendet.

4.5 Named-Entity Recognition²

Das Areal der Named-Entity Recognition (NER) beschreibt ein Subareal der Information Extraction und fokussiert die Entitätsextraktion aus Textdaten. Global ausgerichtete Systeme finden 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 Bereich ist zunächst die Extraktion von Textstellen, welche Entitäten repräsentieren, um diese anschließend unter Hinzunahme von zusätzlichen Attributen, wie der Schreibweise oder deren POS-Tags, zu klassifizieren (vgl. Jurafsky & Martin, 2009, S. 765ff.). GOYAL, GUPTA & KUMAR (2018) deskribieren in ihrer Forschungsarbeit differente Techniken, welche zur NER verwendet werden. So unterscheiden die Autoren zwischen regel- und lernbasierten Methoden, wobei sich letztere zudem in die Kategorien des überwachten, unüberwachten sowie semi-überwachten Lernens unterteilen. EFTIMOV, SELIAC & KOROŠEC (2017) spezifizieren darüber hinaus wörterbuchbasierte Vorgehen als weitere Kategorie von NER-Methoden.

4.5.1 Wörterbuchbasierte und regelbasierte Vorgehen²

Wörterbuchbasierte Methoden definieren im Allgemeinen die Applikation 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, herangezogen. Aufgrund der Domänenunabhängigkeit des hier zu entwickelnden Artefakts und der resultierenden nicht gegebenen Anwendbarkeit von wörterbuchbasierten Methoden werden derartige Heuristiken allerdings nicht weiter vertieft. Grundsätzlich erweisen sich wörterbuchbasierte Methoden als inflexibel und besitzen den Nachteil, dass lediglich im Vorfeld spezifizierte Entitäten aus neuen Texten extrahiert werden können (vgl. Eftimov et al., 2017, S. 5).

Konträr spezialisieren regelbasierte Verfahren auf die Nutzung von regulären Ausdrücken, mittels denen terminologische Informationen und Charakteristika der Entitäten inkludiert werden können. So werden Ausdrücke definiert, welche konkrete Muster widerspiegeln, die charakteristisch für die zu extrahierende Entitäten sind, wodurch domänen spezifisches Wissen einbezogen werden kann. Gleichzeitig stellt dies den Nachteil dieser Methodik dar, da diese Regeln nur schwer generalisierbar sind und somit lediglich für spezielle Anwendungen Funktionalität sichern können. Weiterhin setzt die Erstellung derartiger regulärer Ausdrücke die Kenntnis der linguistischen Struktur der verwendeten Sprache voraus, weshalb die Methode zudem auf eine Sprache restringiert ist. Summierend bietet diese Methodik, durch den Einbezug von domänen spezifischen Informationen, zwar eine Möglichkeit zur Extraktion von Entitäten, ist allerdings auf eine Domäne bzw. Anwendung und Sprache beschränkt. Resultierend aus diesem Sachverhalt existieren in aktueller Literatur viele Ansätze im Bereich der lernbasierten Methoden (vgl. Goyal et al., 2018, S. 25 und Nadeau & Sekine, 2007, S. 6).

4.5.2 Lernbasierte Verfahren der Named-Entity Recognition²

Im Vergleich zu den oben genannten Methoden basieren lernbasierte Verfahren auf Machine-Learning-Algorithmen, um automatisch Textentitäten zu extrahieren. Dabei erbringen derartige Verfahren i.d.R. eine bessere 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 Kennungen versehen, welche Aufschluss darüber geben, ob Wörter oder Wortgruppen einzelne Entitäten darstellen (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 16 illustriert ist.



Abbildung 16: Prozess lernbasierter Verfahren der Named-Entity Recognition (Eigene Darstellung)

Allgemein ist der Schritt der Feature Extraction als essentiell bei der Extraktion von Textentitäten anzusehen, um die multidimensionalen Aspekte derselben getreu zu erfassen. Dabei unterscheiden sich die zu extrahierenden Eigenschaften in drei verschiedene Gruppen. Wörterbuchbasierende Eigenschaften dienen als Look-Up-Tabelle, um Wortzugehörigkeiten zu erfassen. Hinzukommen Dokumenteneigenschaften, welche strukturelle und syntaktische Merkmale aufgreifen. Die wortbasierten Eigenschaften hingegen konzentrieren sich darauf, semantische Aspekte einzelner Wörter zu inkludieren. Somit unterscheiden sich lernbasierte NER-Verfahren von anderen NLP-Methoden, wie beispielsweise Embeddings, da diese ausschließlich 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, komplexere Modelle entlang des illustrierten Prozesses. Beispielsweise können bidirektionale LSTMs dazu verwendet werden, Dependenzbeziehungen zu isolieren und diese Dokumenteneigenschaften wiederum für das spätere Training eines Classifiers verwendet werden (vgl. Kiperwasser & Goldberg, 2016, S. 323).

Neben den überwachten Verfahren etablieren sich auch **unüberwachte Verfahren** der NER, welche sich hauptsächlich auf Clustering- und Assoziationsanalysen stützen. Zum einen werden Clusteringverfahren zur Extraktion kontextueller Ähnlichkeiten einzelner Entitäten appliziert. Zum anderen werden Assoziationsanalysen dafür genutzt, um Beziehungen zwischen potentiellen Entitätskandidaten zu entdecken und diese anhand verschiedener Algorithmen zu bewerten. Eine Kombination beider eben genannter lernbasierter Verfahren bilden **semi-überwachte Ansätze**. Hierbei wird aus einer Mischung von unüberwachten und überwachten Verfahren ein NER implementiert. Die Intention ist es, ein überwachtes NER-Modell einzusetzen, welches einen kleinen Teil annotierter Daten verwendet, um die restlichen nicht-annotierten Daten zu kennzeichnen. Als letzte Kategorie sind **hybride Verfahren** des NER zu nennen, die die Vorteile der beiden Hauptkategorien vereinen. So beschreiben diese Ansätze die Verkettung von wörterbuch- oder regelbasierten mit lernbasierten Methoden, wobei die regelbasierten Methoden zur Vorauswahl und Bereitstellung von Lerninformationen dienen. Die Systeme werden zwar als mehr akkurat empfunden, gehen jedoch mit einem größeren Aufwand in ihrer Erstellung einher (vgl. Goyal et al., 2018, S. 26 sowie Nadeau & Sekine, 2007, S. 6f.).

Allem in allem ist das Training eines NER-Modelles mit vielen Herausforderungen verbunden. So geht das Labeln der Daten mit einem großen Aufwand einher. Zwar gibt es bereits verfügbare, umfangreiche Datensätze annotierter Daten, allerdings fehlen in diesen domänen- sowie sprachspezifische Konstrukte. Hinzukommen weitere 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 wird sich in der vorliegenden Forschungsarbeit auf einen regelbasierten Ansatz zur Extraktion von potenziellen Analyseentitäten aus den vorliegenden Problemstellungen gestützt. Das detaillierte Vorgehen wird im folgenden Kapitel 6.3.3 beschrieben.

5 Prozess der Datenbeschaffung¹

Das folgende Kapitel gibt eine Übersicht zu den durchgeführten Schritten der Beschaffung von adäquaten Daten bzw. zu deren Selektion und zielt demnach auf die Beantwortung der Forschungsfrage I ab. Im Wesentlichen wird in der vorliegenden Arbeit der in Abbildung 17 dargestellte Prozess verfolgt, welcher weiterführend deskribiert wird.



Abbildung 17: Prozess der Datenbeschaffung und -selektion (Eigene Darstellung)

Wie zu erkennen erfolgt zunächst das Crawling von wissenschaftlichen Datenbanken zur Bildung 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 verwendet 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 Datenbanken¹

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 somit ermöglichen die wachsende Menge an Inhalten im Internet leichter zu assimilieren. Dies wird durch das Auslesen von Website-Quelltexten

sowie dem Folgen von Hyperlinks zwischen Seiten erreicht (vgl. Dhenakaran & Thirugnanan, 2011, S. 265). Im aktuellen Kontext werden die Datenbanken arXiv, der Verbund an Datenbanken von Elsevier sowie das Online-Journal Medium⁶ durchsucht. Hierbei ist anzumerken, dass für erstere Datenbanken Python-Programmierschnittstellen der jeweiligen Organisationen ihre Anwendung finden, wohingegen Medium mittels eines klassischen Crawlers durchsucht wird.

Zu Beginn werden hierfür verschiedene Suchbegriffe definiert, welche im Rahmen der technischen Möglichkeiten zwischen den differenten Datenbanken standardisiert werden. Aufgrund der Fokussierung auf Textdaten, welche Informationen über die unterschiedlichen 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, die diese Begriffe in anderem Kontext verwenden, auszuschließen. Weiterhin wird spezifiziert, dass der Titel der Veröffentlichung ebenfalls die jeweilige Algorithmusklasse beinhalten muss. Aufgrund der Zusammenführung 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. Weiterhin werden aufgrund der geringen Trefferanzahl für die Klasse *Frequent Pattern Mining* ebenfalls die Subareale *Association Rule Mining* und *Sequential Pattern Mining* in die Suchanfragen integriert. 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 explizit 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 die Häufigkeiten von Wörtern eine starke

⁶ <https://medium.com> [Aufgerufen am: 22.09.2018]

Abhängigkeit zur Datenqualität bzw. dem enthaltenen Rauschen aufweisen. Hierfür werden die Suchanfragen um den Term „*is*“ im ersten Teil der Anfrage erweitert. Exemplarisch ergeben sich somit weiterhin die folgenden Anfragen⁷:

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 18 zeigt einen Ausschnitt dieser Dokumentation und stellt die Quellenanzahl pro Publikationsjahr für das Crawling der Elsevier-Datenbanken dar.

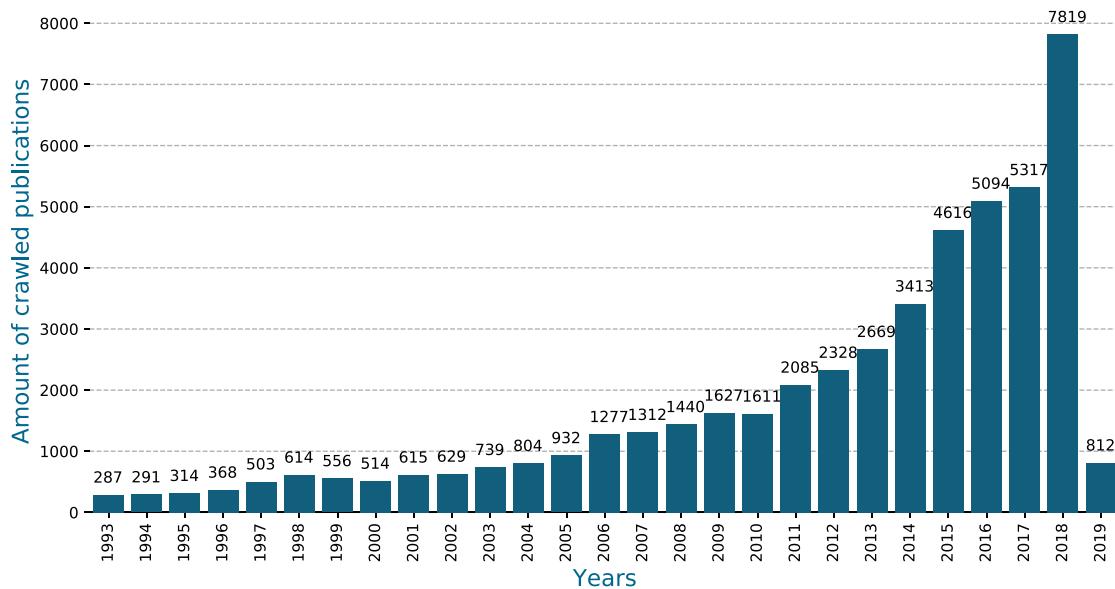


Abbildung 18: Crawling-Ergebnisse in Elsevier (Eigene Darstellung)

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.

⁷ Eine komplette Auflistung der verwendeten Suchanfragen findet sich in Tabelle A1.

5.2 Datenevaluierung und -selektion¹

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

5.2.1 Regelbasierte Datenbereinigung¹

Zur allgemeinen Vorverarbeitung der Daten und speziell zum Bereinigen der Texte von unnötigem Rauschen empfiehlt sich ein rein logikbasierter Ansatz. Im Rahmen des NLP werden hierbei häufig Technologien wie Stemming, Lemmatization, das Entfernen von Stopwords oder reguläre Ausdrücke eingesetzt, um semantisch irrelevante lexikalische Bausteine aus dem Text zu entfernen (vgl. Jurafsky & Martin, 2009, S. 806). Allerdings bietet sich bei der Verwendung von distribuierten Repräsentation nur eine sehr verhaltene Datenvorverarbeitung an, da diese keinen signifikant positiven Einfluss auf die resultierende Qualität der Embeddings hat (vgl. Camacho-Collados & Pilehvar, 2018b, S. 44).

Basierend auf den extrahierten Daten in Kapitel 5.1 wird im Rahmen der Arbeit, Textrauschen im Folgenden als alle Duplikate, Zahlen, Abkürzungen, einzelne Buchstaben, Sonderzeichen und Leerzeichen definiert. Des Weiteren werden Formeln sowie Variablen (vgl. Formeln 22 und 23) mithilfe von regulären Ausdrücken aus dem Text gefiltert.

$$r"(\backslash s +\backslash S +\backslash d +\backslash S *)|(\backslash s +\backslash S *\backslash d +\backslash S +)" \quad (22)$$

$$r"(\backslash w *\backslash W +\backslash d+)" \quad (23)$$

Hinzukommt, dass Dokumente mit einer Länge von weniger als sieben Satzzeichen automatisch aussortiert und Sätze relativens Anteil an Rauschen von mehr als 45% entfernt werden.

5.2.2 Entfernen von Ausreißern²

Wie anfänglich bereits erwähnt, ist anzunehmen, dass aufgrund der Datenvielfalt semantische Divergenzen zwischen den Dokumenten einer Zielklasse auftreten können. Wie in der untenstehenden Abbildung 19 deutlich zu erkennen, reicht das alleinige Entfernen mittels regulärer Ausdrücke nicht aus, um irrelevante Dokumente aus dem Datenpool zu entfernen. Mithilfe der in Kapitel 4.1 angesprochenen Modelle lassen sich die einzelnen Dokumente in dichtbesetzte 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). Hierfür wird im Folgenden der zugrundeliegende Datenpool auf Grundlage eines Deep-Averaging-Networks dargestellt.

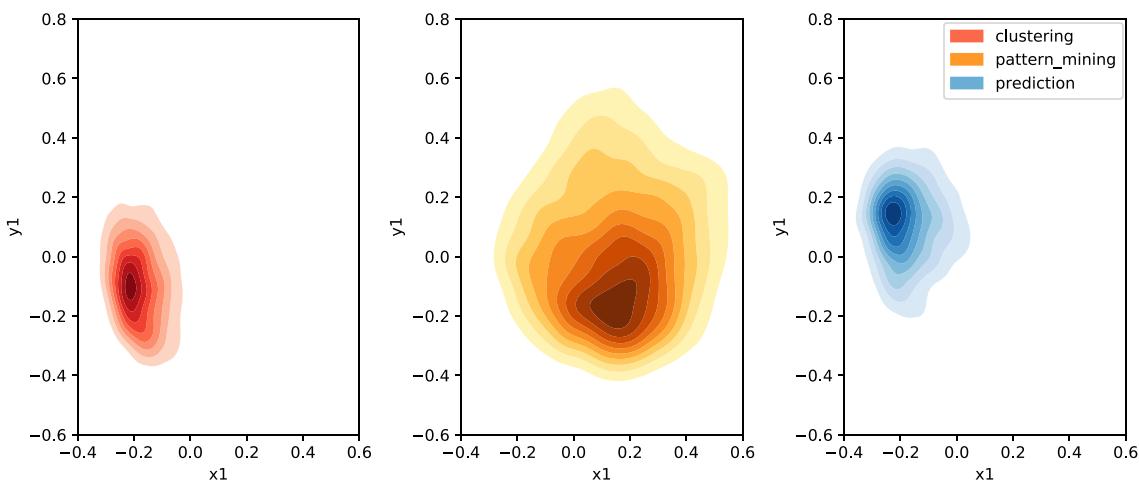


Abbildung 19: Density Graph des Datenpools (Eigene Darstellung)

Die oben illustrierten Diagramme veranschaulichen die Datenverteilung der einzelnen Dokumente in ihrer distribuierten Form. Die dunkleren Areale symbolisieren ein relativ dichtes Aufkommen semantisch ähnlicher Dokumente und die helleren, dünner besetzten Flächen, die eine höhere semantische Diskrepanz zur Ausgangsklasse aufweisen. Interpretierbar ist, dass ausgehend von der distributionellen Hypothese die dichtbesetzten Areale die Bedeutung der Zielklasse besser verinnerlichen als dünnbesetzte. Analog ist dies in den nachfolgenden Streudiagrammen verdeutlicht.

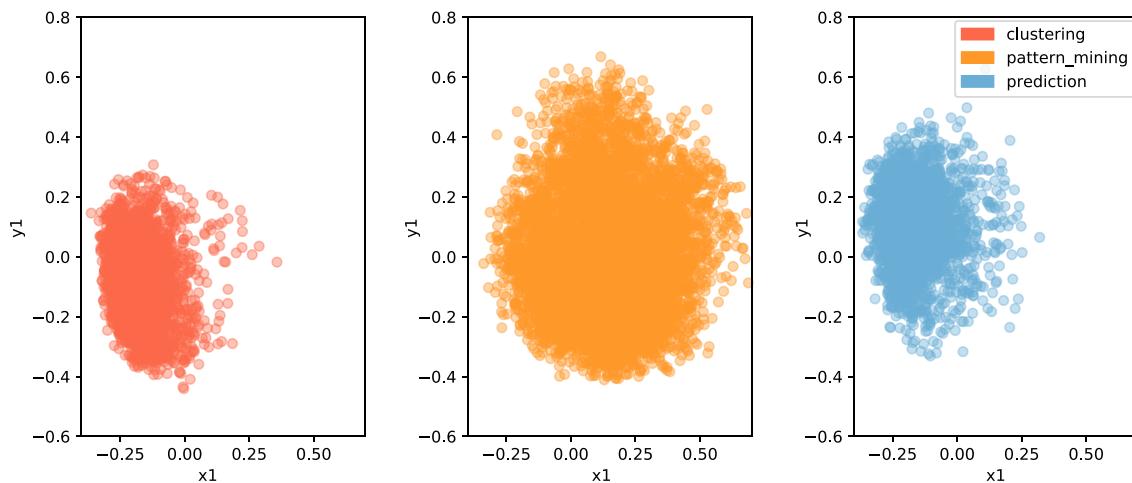


Abbildung 20: Streudiagramme des Datenpools (Eigene Darstellung)

Bedingt durch die semantische Sensitivität der Embeddings können die Ausreißer darauf aufbauende Machine-Learning- und Klassifikationsmodelle negativ beeinflussen (vgl. Rehm, Klawonn & Kruse, 2007, S. 489 und 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 Textclustering, welches sich verschiedenen Clusteringmethoden bedient, um ähnliche Textdokumente zu formieren (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 Clusteringprozesses ein Noise-Cluster gebildet, welches alle Datenpunkte beinhaltet, die sich keiner Klasse zuordnen lassen. Die Idee lässt sich auf verschiedene Clusteringalgorithmen transferieren, wobei deren Anwendung vom jeweiligen Anwendungsfall abhängt (vgl. Dave, 1991, S. 657f. und Rehm et al., 2007, S. 458). Als NC-Algorithmus wird im Folgenden das dichtebasierete Verfahren DBSCAN verwendet, um semantisch irrelevante Elemente zu identifizieren. Dieses Verfahren verspricht effizient Cluster beliebiger Form zu erkennen 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 minimale Anzahl an Punkten (*MinPts*) vorhanden sein muss, um einem Cluster zugeordnet zu werden (vgl. Ester et al., 1996, S. 228–230). Auf eine ausführliche Erläuterung des Algorithmus wird aufgrund des Rahmens der Arbeit verzichtet.

Ergebnisse und Evaluation

Das Ziel der Filterung von Ausreißern der aufgezeigten Methodik dient zum einen semantische, homogene Klassen zu erzeugen. Zum anderen soll damit auch die Separierbarkeit zwischen den Klassen erhöht werden, um eine sich anschließende Textklassifikation zu erleichtern. Zur Validierung der identifizierten Cluster wird fortlaufend der Davies-Bouldin-Index herangezogen. Diese Validationsmetrik basiert dabei alleinig auf den Clustern selbst und nicht auf den annotierten Eingangsdaten. Der Index ist ein Indikator für die Stärke der Separierbarkeit der Cluster. Ein niedriges Ergebnis lässt dabei auf eine bessere Trennschärfe der Klassen schließen. Die grundlegende Intention drückt die nachfolgende Formel aus:

$$DB = \frac{1}{K} \sum_{i=1}^k \max_{j=1, \dots, k, i \neq j} \left(\frac{diam(c_i) + diam(c_j)}{d(z_i, z_j)} \right) \quad (24)$$

$$diam(c_i) = \sqrt{\frac{1}{n_i} \sum_{x \in c_i} d(x, z_i)^2} \quad (25)$$

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_j widerspiegelt (vgl. Davies & Bouldin, 1979, S. 224ff.).

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

Tabelle 2: Übersicht der Parametersuche für Noise-Clustering

| # | Eps | MinPts | DB |
|---|-----|--------|-------|
| 1 | 0.9 | 70 | 3.767 |
| 2 | 0.9 | 10 | 3.769 |
| 3 | 0.7 | 10 | 3.770 |

Hierbei wird iterativ eine Kombination aus den beiden Parametern *Eps* und *MinPts* getestet, um anschließend den Davies-Bouldin-Index als Entscheidungsgrundlage heranzuziehen. Wenngleich auch keine großen Schwankungen zwischen den Ergebnissen liegt, wird die finale Parameterkombination des NC-Verfahrens durch den geringsten Davies-Bouldin-Index bestimmt.

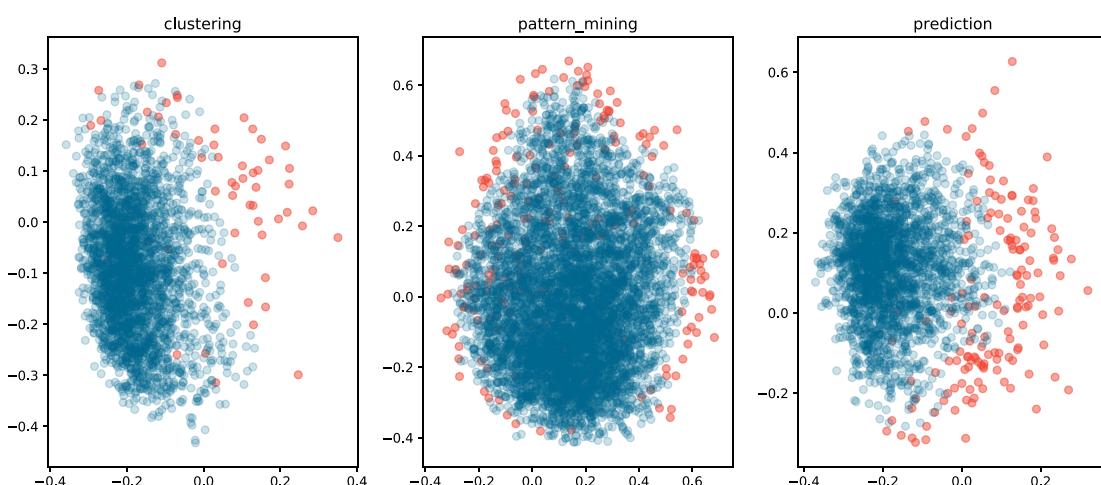


Abbildung 21: Durch DBSCAN identifizierte Ausreißer (Eigene Darstellung)

Die obige Abbildung illustriert die gefundenen Ausreißer. Anzumerken ist hierbei, dass die hohe initiale Dimensionalität der distribuierten Textrepräsentationen ein NC diffizil gestaltet. Aus diesem Grund werden die Datenpunkte mithilfe des Verfahrens der *Principle Component Analysis* (PCA) auf drei Dimensionen reduziert und anschließend dem DBSCAN-Verfahren unterzogen. Wie die roten Datenpunkte veranschaulichen, werden durch das angewendete Verfahren semantische Ausreißer adäquat gefiltert. Die Sensitivität des DBSCAN-Verfahrens gegenüber den Parametern *Eps* und *MinPts* erschwert jedoch ein exaktes Filtern aller Ausreißer. Additional werden in seltenen Fällen Datenpunkte innerhalb der Dichtezentren als Ausreißer identifiziert, was ebenfalls auf die komplexe Parameterauswahl des Verfahrens zurückzuführen ist (vgl. Halkidi, Batistakis & Vazirgiannis, 2001, S. 21).

Neben der algorithmischen Evaluation der Daten empfiehlt sich eine manuelle Begutachtung der Daten, wenngleich ein solcher Ansatz subjektiver Natur ist. Dennoch wird der zugrundeliegende Text der Ausreißer stichprobenartig auf semantische Aussagekraft überprüft. Die semantisch sinnvollen Texte werden demzufolge aus dem Noise-Cluster entfernt und erneut der Datenbasis zugeführt.

5.3 Data Augmentation²

Anschließend an die Bereinigung der Ausgangsdaten erfolgt die Expansion dieser mittels 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 und somit die Variabilität des Datensatzes erhöht wird. Insbesondere im Bereich der Computer Vision bzw. der Bildverarbeitung mittels Deep Learning haben 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 etwa den Pixeln eines Bildes, der bestehenden Daten fokussieren und somit keine eventuell vorherrschenden Zusammenhänge verändert werden. Dieses Vorgehen ist bei einer textuellen Datenbasis aufgrund der zugrundeliegenden Semantik, bestehender Relationen sowie der Ambiguität von Sprache als diffizil zu charakterisieren, da die Änderung von einzelnen Buchstaben in Textdaten den Sinn bzw. die Aussage verändern kann (vgl. Fadaee, Bisazza & Monz, 2017, S. 567 sowie Zhang & LeCun, 2015, S. 3f.).

5.3.1 Methoden zur Substitution und Reihenfolgenanpassung²

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. Zhang & LeCun, 2015, S. 3f.).

Ein weiterer Ansatz findet sich in der Variation von Satz- bzw. Paragraphenfolgen in längeren Dokumenten. Dieses Vorgehen bietet potentielle Qualitätsverbesserungen in Bezug auf Paragraph-Embeddings. Gleichwohl lässt sich dieses Verfahren auf Wortebene transferieren, indem Wortfolgen in Sätzen randomisiert werden. Diese Wordrehungen 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 für häufigkeits- bzw. verteilungsbasierte Modelle irrelevant ist und somit keine neuen Dokumente generiert werden können. Letztendlich ist demnach lediglich ein potentieller Vorteil für Paragraph-Embeddings zu vermerken.

Additional finden sich in der Literatur ebenfalls komplexere Methoden zur Augmentation von textuellen Datensets unter der Anwendung von Machine Learning sowie Deep Learning. So spezifizieren WANG & YANG (2015) ein Vorgehen, welches analog zum ersten 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 jeweils nächste Wort im Vektorraum (K-Nearest-Neighbor) und erreichen so mit neue Sätze (vgl. Wang & Yang, 2015, S. 2559f.). Hierbei ist jedoch eine starke Abhängigkeit zur Qualität des Embedding-Modells zu vermerken, da diese entscheidend für die Sinnhaftigkeit der Substitution und folglich für die Qualität der Ergebnisse ist.

5.3.2 Generierung von Textdaten¹

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 neue Textdaten erstellt werden können. In Referenz zur vorausgegangen Forschungsarbeit und den darin beschriebenen differenten Wahrscheinlichkeiten basieren Markov-Modelle im Kontext der Textgenerierung 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 be-

stimmt ist. So werden zur Vorhersage des nächsten Wortes nicht die verketteten, bedingten Wahrscheinlichkeiten aller Wörter verwendet, sondern lediglich die der vorherigen zwei Wörter (vgl. Jurafsky & Martin, 2009, S. 120ff.).

Ferner findet sich ebenfalls die Möglichkeit Deep-Learning-Methoden, wie beispielsweise RNNs oder LSTMs, zu applizieren. Hierbei existiert eine große Bandbreite an Möglichkeiten, weshalb im aktuellen Kontext lediglich ein kurzer Einblick hierzu 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 Erstellung 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 Grundlage der bereits erstellten Wortsequenz zu generieren. BOWMAN ET AL. (2016) illustrieren weiterführend die Verwendung eines Encoder-Decoder-Modells zur Textgenerierung und fokussieren hierbei sog. *Variational-Autoencoder-Modelle*. 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). JORGE ET AL. (2018) evaluieren diese Methode im Bereich der Computer Vision und zeigen signifikante Genauigkeitsverbesserungen hinsichtlich unterschiedlicher Classifier.

Weiterhin birgt das Areal der Machine- bzw. *Neural Machine Translation* (NMT) (z.Dt.: Maschinelles Übersetzen) einen weiteren Ansatz zur Augmentation von Textdaten. Resultierend aus der geringen Verfügbarkeit von qualitativen Datensätzen, welche Übersetzungspaare zweier Sprachen umfassen, erhält 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, woraus eine additionale Datenbasis gewonnen werden kann. So beschreiben SENNRICH, HADDOW & BIRCH (2016) sowie DOMHAN & HIEBER (2017) mögliche Implementierungen dieser Back-Translation in Sequence-to-Sequence-Modelle, um monolinguale Datensätze in multilinguale zu transferieren und diese somit zusätzlich zum Training eines NMT-Modells zu verwenden (vgl. Coulombe, 2018, S. 15f.).

5.3.3 Applizierte Methoden der Data Augmentation¹

Im Rahmen der vorliegenden Arbeit werden drei differente Methoden zur Augmentation der erlangten Textdaten angewandt. Anzumerken ist hierbei, dass jegliche dieser Methoden auf den Grunddaten ausgeführt werden. So werden bereits veränderte Daten nicht zur Implementierung einer Augmentationsmethode herangezogen, um eine Kummulation der Fehler aller Methoden zu vermeiden. So wird unter anderem ein Vorgehen zur Substitution von Nomen und Verben durch Synonyme festgelegt, bei dem WordNet⁸ als Quelle für potentielle Substitutionskandidaten fungiert und Stopwords wie „is“, „the“ und „a“ ausgeschlossen werden. Zur Qualitätssicherung der Substitution werden Kriterien definiert, die durch das Substitutionswort erfüllt werden müssen. Tabelle 3 listet diese Kriterien auf.

Tabelle 3: Kriterien zur Augmentierung von Daten mittels Substitution

| Kriterium | Beschreibung |
|--------------------------------|---|
| <i>POS-Tag Übereinstimmung</i> | Das vorhandene sowie das Substitutionswort müssen über denselben POS-Tag verfügen. |
| <i>Ähnlichkeitsgrenzwert</i> | Beide Wörter müssen eine bestimmte semantische Ähnlichkeit zueinander aufweisen. |
| <i>Wortdistanz</i> | Analog zum Ähnlichkeitswert müssen die Wörter eine Mindestdistanz auf Buchstabenebene aufweisen. |
| <i>Zufallswert</i> | Randomisierter Wert, der einen Grenzwert überschreiten muss und somit über die generelle Substitution eines Wortes entscheidet. |

Das erste Kriterium in obiger Tabelle stellt hierbei sicher, dass Substantive und Verben ausschließlich mit gleichartigen Wörtern ersetzt werden. Additional wird ein Grenzwert bzgl. der Ähnlichkeit beider Wörter spezifiziert, durch den sichergestellt werden kann, dass keine starke Variation der Semantik vorgenommen wird. Hierfür wird das Maß der

⁸ WordNet beschreibt eine lexikalische Online-Datenbank, die eine Ontologie der englischen Sprache abbildet und unter folgendem Link einsehbar ist: <https://wordnet.princeton.edu> [Abgerufen am: 14.11.2018]

Kosinus-Ähnlichkeit auf Basis von mittels der Architektur FastText trainierten Embeddings appliziert. FastText bietet dabei, oppositionell zu anderen Embedding-Architekturen, den Vorteil, dass unbekannte Wörter durch die Fokussierung auf Buchstaben-N-Gramme ebenfalls in Vektoren transformiert werden können. Hierdurch kann gewährleistet werden, dass adäquate Vektoren auch für potentiell nicht trainierte Wörter abgeleitet und folgerichtige 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. In der aktuellen Untersuchung wird hierfür 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 drei 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, der für eine Substitution durch eine randomisierte Zahl überschritten werden muss. Dies erfolgt zur Erhöhung der Varianz in den substituierten Sätzen. Tabelle 4 zeigt 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. Zunächst werden zur Implementierung des Modells sämtliche vorhandene Sätze mit Beginn und Endsignaturen $\langle E \rangle$ pro Satz versehen, um anschließend **Trigramme** der bestehenden Sätze zu bilden.

s = Clustering is an unsupervised method

(< E >, < E >, Clustering), (< E >, Clustering, is), ..., (method, < E >, < E >)

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

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})} \quad (26)$$

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 zwei Wörter der bestehenden Dokumente des Datensatzes 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 5 zu erkennen, garantieren die generierten Sequenzen keine syntaktische Korrektheit. In der aktuellen Untersuchung ist dies allerdings zu vernachlässigen, da diese Korrektheit vor allem für häufigkeitsbasierte Verfahren nicht von Relevanz ist und die alleinige Anwendung von Wahrscheinlichkeiten durch das Markov-Modell keine Garantie syntaktischer Korrektheit zulässt.

Analog zur Datenselektion erfolgt abschließend eine Evaluierung der augmentierten Daten anhand von Paragraph-Embeddings. Hierzu findet die Architektur eines *Deep Averaging Networks* Anwendung (vgl. Kapitel 4.1.2). Abbildung 22 illustriert die Verteilung der mittels PCA reduzierten Vektoren in Form von Dichtediagrammen.

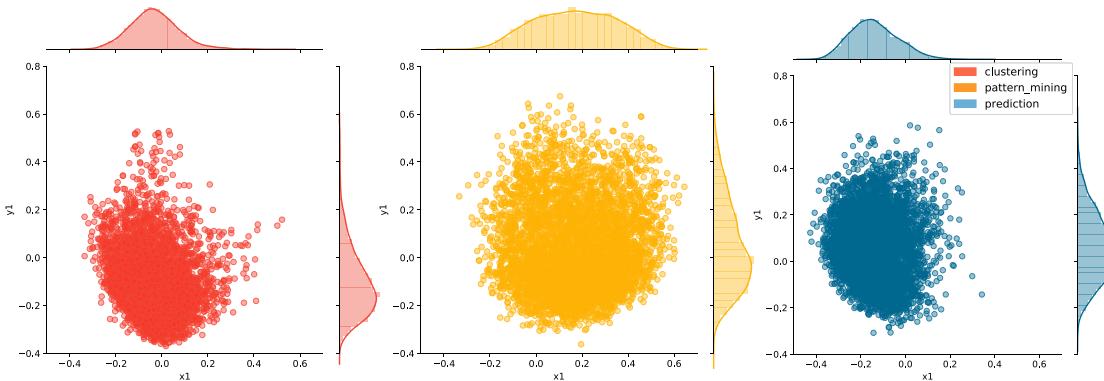


Abbildung 22: Dichtediagramm augmentierter Daten des gefilterten Datensatzes (Eigene Darstellung)

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 zurückzuführen, 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 die generierten Daten schlechter Qualität zu entfernen und so den Streuradius zu reduzieren. Hierfür werden erneut die Methoden der regelbasierten Datenreinigung sowie des Noise-Clustering angewendet.

5.4 Erstellung von Datensätzen²

Das folgende Kapitel schildert die finale Zusammenstellung der Trainings- und Validierungsdaten für die vorliegende Untersuchung. Hierfür wird zunächst auf den Prozess zur Erstellung differenter Trainingsdaten und anschließend auf Validierungsdaten eingegangen.

5.4.1 Erstellung der Trainingsdatensätze²

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, der ausschließlich Kurzzusammenfassungen der gefundenen Publikationen umfasst. Dies stützt sich auf die Annahme, dass

relevanter Kontext zu den Algorithmenklassen sowie Informationen hinsichtlich deren jeweiliger Anwendung gebündelt im Abstract zur Verfügung stehen. Zum anderen wird ein weiterer Datensatz kreiert, der eine gefilterte und optimierte Version aller Daten darstellt. Dabei enthält dieser Datensatz definitionsähnliche Inhalte, um eine möglichst hohe Datenqualität zu erzielen, da davon auszugehen ist, dass diese Inhalte notwendige Semantik besser bündeln. In Abbildung 23 wird das Vorgehen zur Erstellung dieses Datensatzes schematisch illustriert.



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

Hierbei wird die Menge der Sätze im zugrundeliegenden Datensatz nach spezifischen Schlüsselwörtern durchsucht. Diese umfassen exemplarisch die Satzteile „Clustering is“, „Clustering defines“ oder „Clusters are“ und sichern somit einen definitionsähnlichen Inhalt⁹. Weiterhin werden die so pro Zielklasse extrahierten Sätze in Vektoren überführt, um anschließend die Kosinusähnlichkeit zwischen den Sätzen und einem prädefinierten Topic zu errechnen (vgl. Abbildung 24).

⁹ Eine vollständige Liste der verwendeten Suchbegriffe findet sich im Jupyter Notebook „dataset overview.ipynb“ auf dem beigefügten Datenträger.

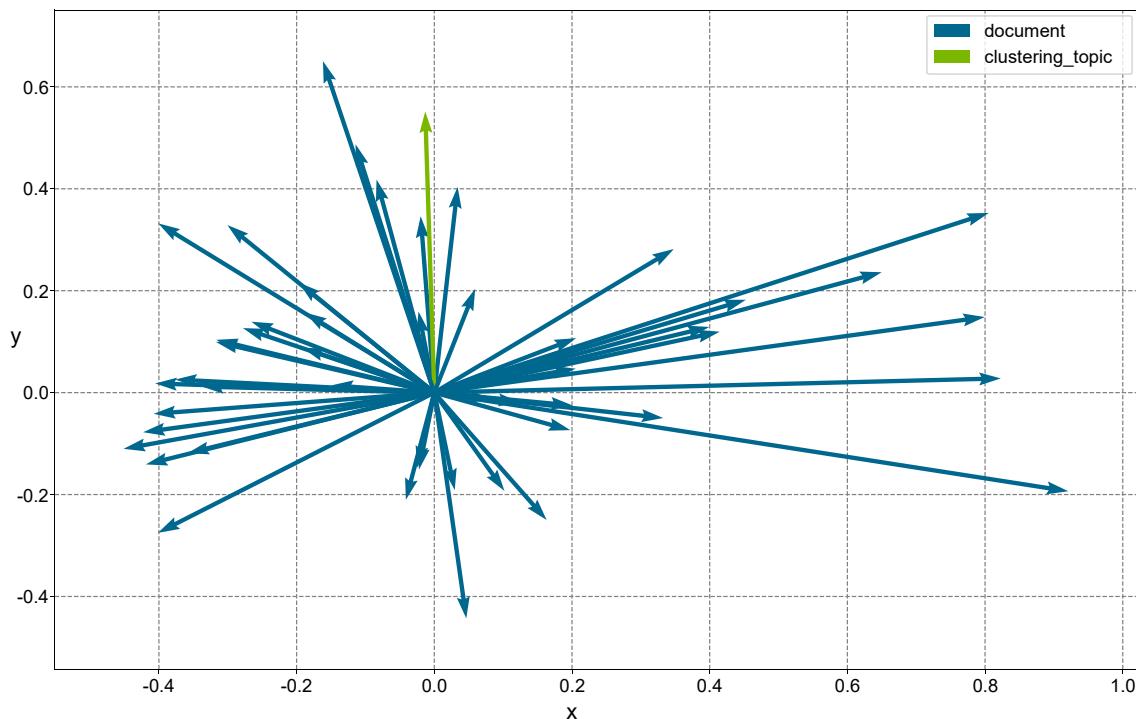


Abbildung 24: Illustration der Kosinusähnlichkeit zur Datenauswahl (Eigene Darstellung)

Diese Topics resultieren dabei aus der vorherigen Applikation von Keyword-Extraction-Methoden auf manuell zusammengetragene Definitionen des jeweiligen Algorithmus aus der Literatur. Abschließend dient das verwendete Similaritätsmaß zur Selektion der ähnlichsten Sätze. In Referenz zur obenstehenden Illustration beschreibt diese Ähnlichkeit die bzgl. ihres Winkels zueinander am nächsten gelegenen Vektoren, wobei die Top 25 Prozent dieser in den Datensatz aufgenommen werden.

Final gibt Tabelle 6 eine Übersicht über die beiden erstellten Datensätze, inklusive einer kurzen Beschreibung dieser.

Tabelle 6: Übersicht über die erstellten Datensätze

| Datensatz | Beschreibung |
|------------------------------|---|
| <i>Abstract-Datensatz</i> | Dieser Datensatz enthält Abstracts aller gefundenen 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 5.3.3 deskribierten Methoden wird der gefilterte Datensatz additional in augmentierter Form erzeugt und abgelegt. Auf diese Weise können eventuelle Unterschiede zwischen augmentierten und originalen Daten erkannt werden, um somit Rückschlüsse auf die Wirksamkeit der Augmentation zu ziehen.

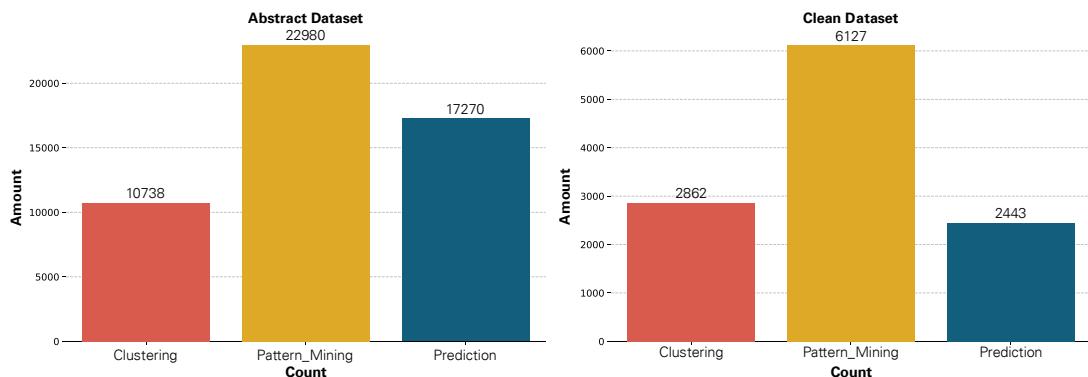


Abbildung 25: Klassenverteilung der finalen Datensätze (Eigene Darstellung)

Obenstehende Abbildung zeigt schlussendlich die Klassenverteilung in den finalen, nicht augmentierten Datensätzen. Hier zu erkennen ist, dass eine starke Ungleichverteilung der Klassen vorliegt, weshalb dieser Bias im weiteren Vorgehen beachtet werden muss.

5.4.2 Erstellung der Validationsdatensätze²

Die in Kapitel 1.1 zugrunde gelegte syntaktische Divergenz der Daten begründet die separate Erstellung eines Validierungsdatensatzes neben der klassischen Aufteilung in Trainings- und Testdaten. Konträr zu klassischen Problemstellungen beinhalten die gefilterten Daten definitionsähnliche Konstrukte und bündeln durch ihr abgegrenztes Vokabular sowie ihre Terminologie die Semantik der Zielklassen.

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

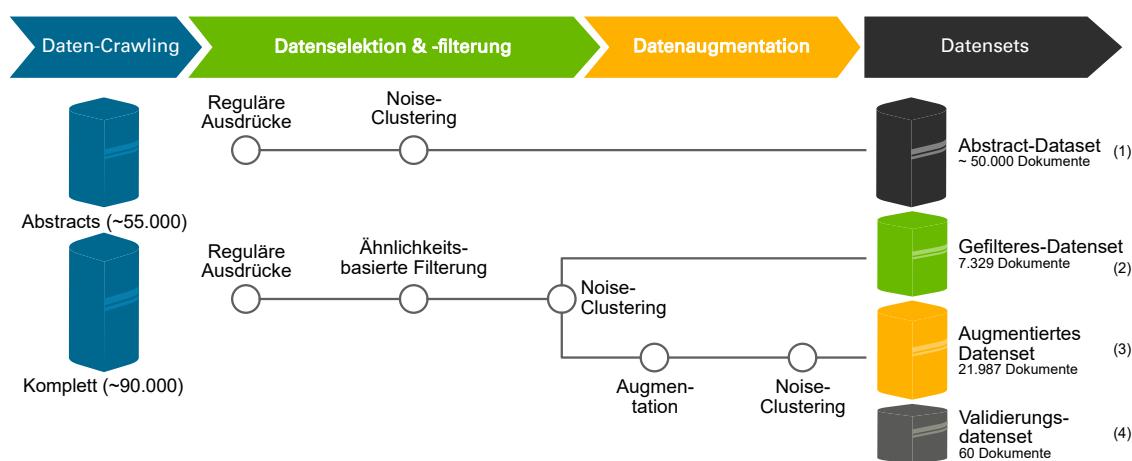


Abbildung 26: Übersicht der Datenmenge in verschiedenen Verarbeitungsschritten (Eigene Darstellung)

Wie ersichtlich ist die ursprüngliche Datenbasis auf rund 30.300 semantisch relevante Daten reduziert. Dabei wird für das Modelltraining der augmentierte Datensatz (3) sowie der gefilterte Datensatz (2) verwendet, abhängig von den resultierenden Modellgenauigkeiten. Der Validierungsdatensatz ist separat von allen anderen Datensätzen zu betrachten und bleibt im weiteren Verlauf der Arbeit in seiner originären Form.

6 Konzeption und Implementierung des Prototyps²

In Referenz zur zweiten forschungsleitenden Frage ordnet sich das folgende Kapitel in der Phase **Design und Development** im eingangs illustrierten Design-Science-Research-Prozess ein und fokussiert demnach die Entwicklung des Artefakts. Hierbei wird sich, wie in Kapitel 1.3.2 dargelegt, an einer Variation des Vorgehensmodells zur Softwareentwicklung nach PRESSMAN & MAXIM (2015) orientiert. Zu Beginn findet im Kontext der Erläuterung des Anforderungsprofils eine Unterteilung in funktionale und nicht-funktionale Anforderungen sowie eine Priorisierung dieser nach IEEE STD 830-1998 (1998) statt. Anschließend wird die Entwicklung des Frontend sowie des Backend dargelegt, um abschließend die finale Programmstruktur zu illustrieren und das entwickelte System den Anforderungen gegenüberzustellen.

6.1 Anforderungsdefinition²

Im nachfolgenden Kapitel werden die Charakterisierung und Definition von Anforderungen vorgenommen, welche an den zu entwickelnden Prototyp gestellt werden. Die resultierenden Anforderungen sind dabei aus dem in Kapitel 1.2 beschriebenen Forschungsvorhaben abgeleitet. Hierbei ist zu betonen, dass sich lediglich auf den zu entwickelnden Prototyp bezogen wird und nicht auf die zu implementierenden Methoden. Zudem handelt es sich um ein initiales Anforderungsprofil, welches in weiterführenden Forschungsvorhaben erweitert werden kann.

In der nachfolgenden Tabelle werden die nach ausführlicher Analyse identifizierten Anforderungen sowie ihre jeweilige Kategorisierung und Priorisierung aufgelistet.

Tabelle 7: Prototypbezogene Anforderungsdefinition

| A_ID | Kategorie | Priorisierung | Anforderung |
|------------|-------------------------|--------------------|--|
| A1 | <i>Funktional</i> | <i>Essentiell</i> | Zuordnung einer Problemstellung zu einer Verfahrensklasse. |
| A2 | <i>Funktional</i> | <i>Essentiell</i> | Ausgabe empfohlener Datenstruktur für Verfahrensklasse. |
| A3 | <i>Funktional</i> | <i>Essentiell</i> | Ausgabe einer personalisierten, exemplarischen Analyse. |
| A4 | <i>Funktional</i> | <i>Konditional</i> | Bessere Zuordnung als reines Raten (Cohen's Kappa > 0.5). |
| A5 | <i>Funktional</i> | <i>Konditional</i> | Extraktion potentieller Analyseentitäten aus Problemstellung. |
| A6 | <i>Funktional</i> | <i>Konditional</i> | Deterministische Zuweisung zu einer Verfahrensklasse. |
| A7 | <i>Funktional</i> | <i>Optional</i> | Möglichkeit zur Eingabe einer kurzen sowie langen Problembeschreibung. |
| A8 | <i>Funktional</i> | <i>Optional</i> | Entgegenwirken von syntaktischer Divergenz. |
| A9 | <i>Funktional</i> | <i>Optional</i> | Ausgabe von Wahrscheinlichkeiten der Zuordnung. |
| A10 | <i>Nicht-Funktional</i> | <i>Essentiell</i> | Einfachheit. |
| A11 | <i>Nicht-Funktional</i> | <i>Essentiell</i> | Erweiterbare Systemarchitektur. |
| A12 | <i>Nicht-Funktional</i> | <i>Konditional</i> | Usability des Systems. |
| A13 | <i>Nicht-Funktional</i> | <i>Konditional</i> | Plattformunabhängigkeit. |

Gemäß des Forschungsvorhabens 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 zusätzlich eine personalisierte, exemplarische Analyse als Ausgabe an den Nutzer geliefert (A3). Gleichzeitig wird beabsichtigt dem Nutzer eine Empfehlung bezüglich der zu verwendenden Datenstruktur zu geben (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 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.

Konditional besteht die Anforderung, dass Analyseentitäten aus der Eingabeproblembeschreibung extrahiert werden (A5), um diese für eine detaillierte Explikation der Datenstruktur zu nutzen. In der Regel ist es die Aufgabe der NER, Entitäten wie Personen, Organisationen, etc. im Text zu identifizieren (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 metrische oder kategorische Variablen aus der Problembeschreibung zu extrahieren und diese in der personalisierten Analyse zur Verfügung stellen. Hinsichtlich der Zuordnung soll das System dem bloßen Raten überlegen sein (A4). Dazu wird im nachfolgenden Kapitel 7.3 das entsprechende Evaluationsvorgehen sowie die -metrik deskribiert. Auch ist in diesem Kontext eine deterministische Zuweisung durch den Prototyp wünschenswert (A6), da diese eine Vergleichbarkeit von Ergebnissen gewährleistet und die nutzerseitige Nachvollziehbarkeit und folglich Akzeptanz fördert.

Darüber hinaus soll dem Anwender bei Fehlen einer ausführlichen Problembeschreibung und zur besseren semantischen Abgrenzung die Möglichkeit gegeben werden eine Kurzbeschreibung einzugeben (A7). Dies resultiert aus der Annahme, dass eine zusammengefasste Version der Problemstellung ebenfalls deren Semantik aggregiert. Durch das Zusammenfügen der beiden Beschreibungen kann der Kern der Aussage vermehrt in den zuzuordnenden Text übernommen werden, um somit die Zuweisung einer Klasse zu erleichtern. Damit die Ergebnisse der Zuordnung nachvollziehbar und schnell verständlich für den Nutzer dargestellt werden, empfiehlt sich die Ausgabe der Zuordnungswahrscheinlichkeit zu einer Verfahrensklasse (A9). Die in Kapitel 1.1 thematisierte Problematik der syntaktischen Divergenz erschwert die Zuordnung einer Problembeschreibung zu einer Verfahrensklasse. Mithilfe diverser Methodiken der distributionellen Semantik wird der syntaktischen Divergenz im Rahmen einer optionalen Anforderung begegnet (A8).

Eine einfache Bedienbarkeit, Systemerweiterbarkeit und Plattformunabhängigkeit bilden die nicht-funktionalen Anforderungen des Prototyps (A10-13). Die Systemerweiterbarkeit soll durch eine objektorientierte Programmierung, in der Implementation der Software gewährleistet werden. Parallel stellen webbasierte Technologien sowie eine abschließende Kompilierung der Software in einem Docker-Container die Plattformunabhängigkeit sicher.

6.2 Frontend Konzeptionierung¹

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



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

In obenstehender Abbildung ist diesbezüglich ein konzeptioneller Entwurf der Hauptseite des Prototyps dargestellt. Wie hieran erkennbar wird in diesem Konzept, in Referenz zu den Anforderungen A7 und A10, ein einfaches Design gewählt, welches schnell zu überblicken ist. Zentral auf der Startseite findet sich demnach direkt die Möglichkeit, 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 die ersten beiden als verpflichtende Eingaben markiert sind.

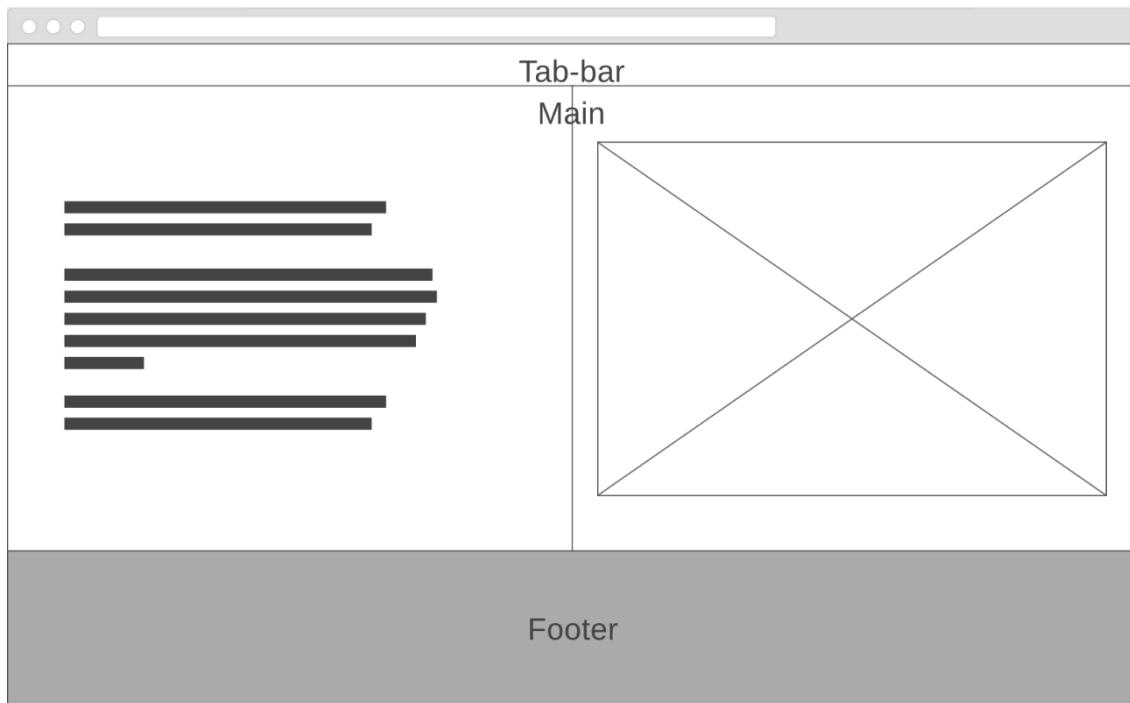


Abbildung 28: Wireframe der Ergebnisseite des Prototyps (Aufgenommen auf Wireframe.cc)

Weiterhin wird jene Einfachheit und Übersichtlichkeit ebenfalls für das Design 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 deren generelle Zuweisung zu einer Verfahrensklasse ausgegeben (A1). Die andere Seite hingegen zeigt die Ausgabe einer exemplarischen Analyse in Form eines Diagramms sowie der benötigten Datenstruktur (A2, A3).

6.3 Entwicklung des Backend²

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 Perspektive auf das System ermöglichen (vgl. Abbildung 29).

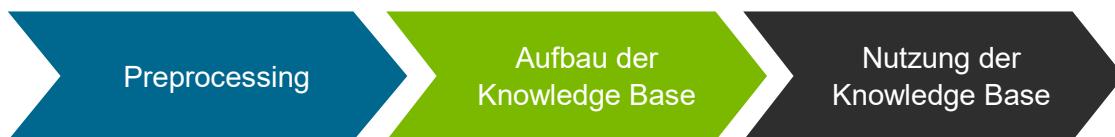


Abbildung 29: Illustration der identifizierten Entwicklungsebenen (Eigene Darstellung)

Wie in obiger Abbildung dargestellt wird 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 die Durchführung dieses Prozesses nicht als streng sequentiell zu erachten ist, da keine generelle Abhängigkeit zwischen den Methoden einzelner Phasen besteht. Beispielsweise werden Methoden zum Aufbau der Knowledge Base appliziert, die keine Notwendigkeit zur umfassenden Vorverarbeitung der Daten aufweisen. Zur Nachvollziehbarkeit ist die Struktur des folgenden Kapitels an diesem Prozess orientiert, wobei der Fokus auf der Illustration der Vorgehensweise sowie der Implementierung liegt. Das Erstellen bzw. Trainieren der differenten Modelle sowie die resultierende Auswahl der jeweiligen Algorithmen wird in Kapitel 7 deskribiert.

6.3.1 Vorbereitung und Vorverarbeitung der Daten²

Im Rahmen der ersten Entwicklungsebene des Systems besteht das Ziel in der Vorverarbeitung und Säuberung der Ausgangsdaten, um darauf aufbauend Methoden zur Ex-

traktion von semantisch relevanten Informationen zu applizieren und damit die Knowledge Base zu kreieren. Hierbei ist anzumerken, dass die differenten Algorithmen in der darauffolgenden strukturellen Ebene auf unterschiedlich vorverarbeitete Ausgangsdaten zurückgreifen, weshalb im Rahmen der Implementierung ein Spektrum an möglichen Vorverarbeitungsmethoden des NLP berücksichtigt wird. Tabelle 8 zeigt hierfür die Zuordnung von Vorverarbeitungsmethoden zu den Methoden der zweiten Ebene.

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

| Preprocessing | Keyword Extraction | Topic Modeling | Embeddings |
|------------------------------------|---------------------------|-----------------------|-------------------|
| <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 in der vorherigen Forschungsarbeit deskribierten Methoden verwendet, um die Ausgangsdaten im Vorfeld zu prozessieren. Hierbei resultiert die explizierte Zuordnung der Methodengruppen auf Basis deren Natur bzw. Funktionalität. So zeigen CAMACHO-COLLADOS & PILHEVAR (2018b), dass eine starke Vorverarbeitung und Bereinigung von Textdaten keine signifikanten Qualitätsverbesserungen im Areal von Embeddings hervorbringen, weshalb auf die Anwendung multipler Vorverarbeitungsmethoden verzichtet wird. Jedoch werden Vermeidung der Abbildung ähnlicher Wörter im Vektorraum, welche sich lediglich durch das Hinzufügen von Zahlen oder Punktierungen unterscheiden (z.B. „clustering“, „clustering?“ und „-clustering“), derartige Zeichen entfernt. Weiterführend sind alle der oben aufgelisteten Methoden im Bereich der Keyword Extraction anwendbar, da durch ein Preprocessing 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 präferiert wird. Dies ergibt sich daraus, dass mittels Stemming Wörter durch Abschneiden von Endungen reduziert werden und hierdurch keine Garantie auf korrekte Wörter gegeben ist. Exemplarisch ist 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, 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 LDA-Algorithmus, da hier ebenfalls Operationen auf Häufigkeiten in Form von Wahrscheinlichkeitsverteilungen durchgeführt werden. Oppositionell zu manchen Modellen der Keyword Extraction findet hier allerdings kein POS-Tagging statt, da alleinig die Zeichenketten, unabhängig deren syntaktischer Rolle, durch den Algorithmus behandelt werden.

Gegeben dieses Sachverhalts, ist im Rahmen der Implementierung zu konstatieren, dass zur vielschichtigen Verarbeitung von Daten eine einheitliche Datenbasis erforderlich ist. Deshalb wird im Kontext der Implementation zunächst eine Korpus-Klasse definiert, die zur Sammlung der Daten aus verschiedenen Quelldateien sowie zur einheitlichen Bereitstellung dieser dient.

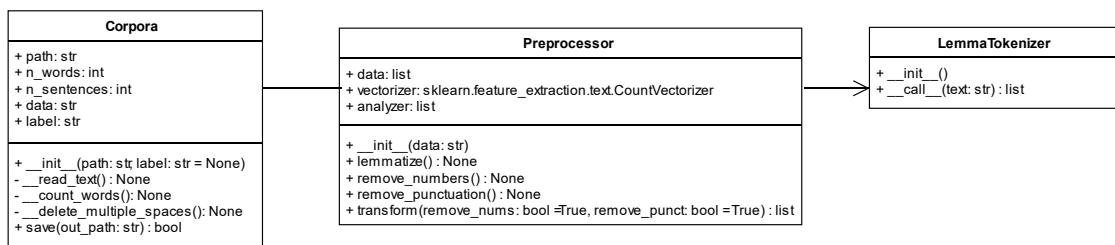


Abbildung 30: Klassenübersicht zur Datenvorverarbeitung (Eigene Darstellung)

Wie in Abbildung 30 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 Base²

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

strukturierter Form speichert. Dabei soll diese die erforderlichen Fakten repräsentieren, welche für die Prognose adäquater Empfehlungen für den Nutzer vonnöten sind. Die Herausforderung besteht hierbei in der Entwicklung eines konzeptionellen Entwurfs, welcher die Applikation aller aufgezeigten Methoden unterstützt. Der grundlegende Aufbau des Systems ist durch nachfolgendes Entity-Relationship-Diagramm dargestellt, welches die einzelnen Entitäten, Relationen, Attribute und Schlüssel eines konzeptionellen Datenschemas veranschaulicht (vgl. Unterstein & Matthiessen, 2013, S. 76ff.). Hervorzuheben ist, dass es sich hier um einen ersten Entwurf der Knowledge Base handelt und nicht um ein konkretes Schema zur Implementierung dieser. Daher werden nur die Entitäten modelliert, welche essentiell für die in Kapitel 6.3.1 aufgezeigten Methoden sind.

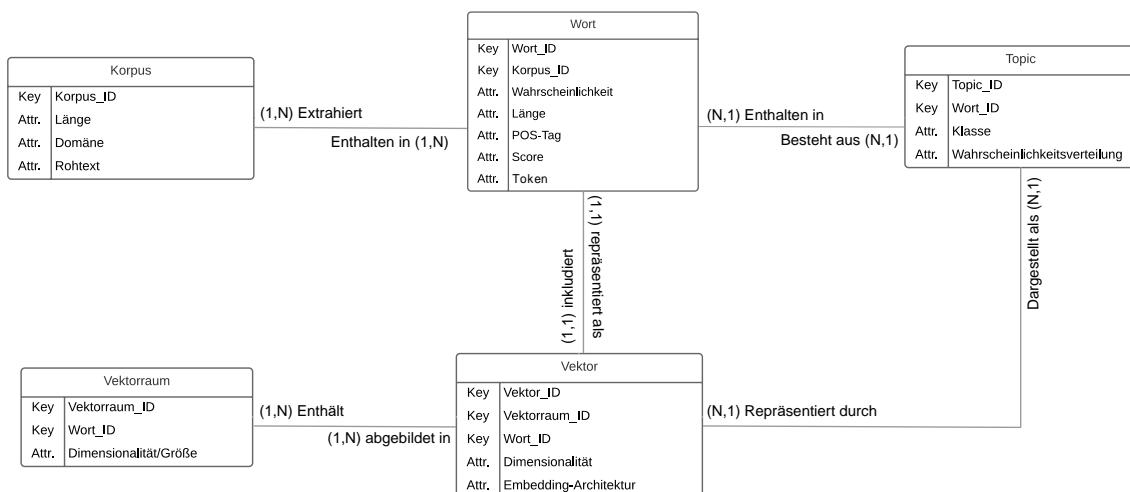


Abbildung 31: Konzeptioneller Entwurf der Knowledge Base (Eigene Darstellung)

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 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 distributionellen Semantik. Im Zusammenspiel mit der Fremdschlüsselbeziehung wird additional die Auftrittswahrscheinlichkeit der einzelnen Wörter im Korpus festgehalten. Deren Berechnung ist andernfalls bei großen Textmen-

gen mit hohem Rechenaufwand verbunden, weswegen sie bereits in der KB vorgehalten werden. Durch die Fremdschlüsselbeziehung erklärt sich ebenfalls das Domänenattribut in der Korpus-Entität. Zum einen 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 Klasse zurückgeführt werden können. Hierbei wird den Polysemen ein klarer semantischer Bezug gegeben und diese werden aufgrund ihres Vorkommens einem klaren Konzept der fachspezifischen Terminologie zugeordnet. Die Differenzierung in Konzepte hilft der in Kapitel 1.1 aufgezeigten linguistischen Problematik entgegenzuwirken.

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

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

Zur Veranschaulichung der Zuordnung von Methoden der vorangegangenen Forschungsarbeit zum konzeptionellen Entwurf dient die nachfolgende Tabelle.

Tabelle 9: Zuordnung der Methoden zu Entitäten der aufgebauten Knowledge Base

| | Korpus | Wort | Vektor | Vektorraum | Topic |
|--------------------|--------|------|--------|------------|-------|
| Keyword Extraction | X | X | - | X | X |
| Topic Modeling | X | X | - | X | X |

| | | | | |
|---|---|---|---|---|
| <i>Klassifikation mithilfe von Embeddings</i> | X | - | X | X |
|---|---|---|---|---|

Die im Rahmen der Arbeit vorgestellten Keyword-Extraction-Methoden fundieren auf graphbasierten (z.B. TopicRank) oder rein statistischen (z.B. TF-IDF) Algorithmen. Im Vergleich zu rein statistik-basierten Keyword-Extraktoren benötigen graphbasierte Verfahren den ganzen zugrundeliegenden Korpus, um die Schlüsselwörter eines Textes zu extrahieren. Demgegenüber greifen Verfahren wie TF-IDF oder TF-IGM ausschließlich auf Wahrscheinlichkeiten zurück, welche sich aus der Wort-Entität ableiten lassen. Generell werden Keyword-Extraction-Methoden zur Erstellung einzelner Topics bezüglich der differenten Verfahrensklassen verwendet. Die resultierenden Topics können wiederum optional in distribuierte Vektoren transformiert werden, um deren semantische Nuancen aufzugreifen sowie einen Vergleich dieser im Vektorraum zu ermöglichen.

Das Topic Modeling fokussiert sich im aktuellen Forschungsvorhaben auf das deskribierte LDA-Verfahren, da dieses das populärste und folglich meist verwendete Modell des Areals sowie eine Weiterentwicklung anderer Modelle des Bereichs darstellt (vgl. Barde & Bainwad, 2017, S. 746ff. und Todor et al., 2016, S. 735). Das Verfahren berechnet, basierend auf konditionalen Wahrscheinlichkeiten, die Wortzugehörigkeit zu einem Topic sowie zu einem Dokument, woraus sich die Einordnung in die Wort- sowie Korpus-Entität begründet.

Die Klassifikation von Embeddings vereint zum einen die Anwendung klassischer Machine-Learning-Klassifikatoren, wie beispielsweise SVM oder K-Nearest-Neighbor-Klassifikationen, zum anderen aber auch Deep-Learning-Ansätze, wie die in Kapitel 4.2 deskribierten LSTM- oder GRU-Architekturen. Hinzukommt, dass deren Input durch Embedding-Modelle determiniert wird und diese wiederum initial durch den Korpus angeleert werden müssen. Dabei unterstützt der oben illustrierte konzeptionelle Entwurf, dass Modelle der distributionellen Semantik auf mehr als einem Korpus trainiert werden können, um beispielsweise deren Generalisierungsfähigkeit zu verbessern. Der oben illustrierte konzeptionelle Entwurf unterstützt dabei insofern, dass sich der Vektorraum aus den resultierenden Embeddings ergibt und als Input für die differenten Klassifikatoren appliziert werden kann. Darüber hinaus können im Falle einer hohen zugrundeliegenden Klassenanzahl optional die Topic-Verteilungen für das Modellieren eines Klassifikators mit herangezogen werden.

6.3.3 Nutzung der Knowledge Base¹

Die Nutzung der KB charakterisiert sich durch die Ausgabe der finalen Empfehlung für den 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. Hierbei werden potentielle Methoden aufgezeigt und im Rahmen des Prototyps miteinander in Beziehung gesetzt, wohingegen die finale Auswahl der tatsächlich angewendeten Methoden auf Basis der Ergebnisse des Evaluationskapitels erfolgt (vgl. Kapitel 7). Im Anschluss an die Deskription der Methodeneinbettung wird das verwendete Vorgehen zur Extraktion der textuellen Entitäten beschrieben, die zur Ausgabe der exemplarischen Analyse eingesetzt werden.

Einordnung der Problemstellung zu einer Verfahrensklasse

Insgesamt dienen zur Einordnung von Problemstellungen multiple potentielle Vorgehen, welche final im Rahmen eines Verfahrens des Ensemble Learnings aggregiert werden. In Abbildung 32 wird hierfür eine strukturelle Übersicht der Methoden gegeben.

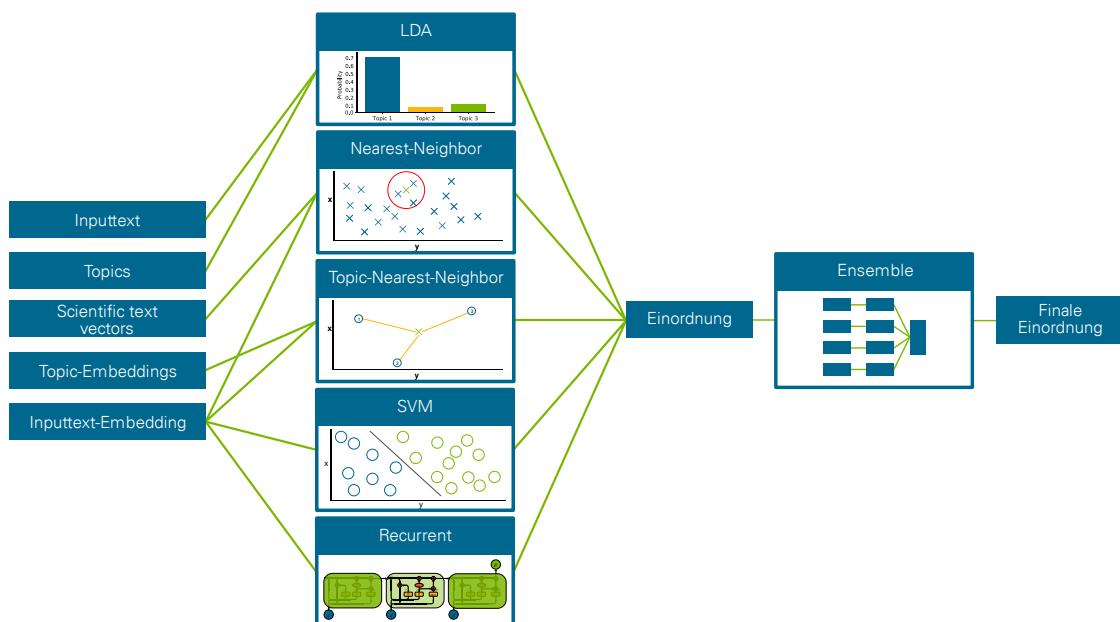


Abbildung 32: Illustration von potentiellen Vorgehensweisen in der Klassifikation (Eigene Darstellung)

Zunächst ist, in Referenz zu vorherigem Kapitel, auf das LDA-Modell zu verweisen, welches indirekt, durch die Identifikation 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 verwendet. Diese Art der Anwendung basiert auf der Voraussetzung, 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 Topic-Verteilungen zu trainieren. Aufgrund der schlechten Modellgüte dieser Kombination in der aktuellen Untersuchung wird diese allerdings vernachlässigt (vgl. Kapitel 7.3.2).

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 auf. In der aktuellen Untersuchung wird sich aufgrund dieser Vielfalt einerseits auf die Anwendung einfacherer Methoden mit K-Nearest-Neighbor-Klassifikationen und Support Vector Machines sowie andererseits auf Methoden des Deep Learning mit RNN-basierten Modellen konzentriert.

Ersterer Algorithmus findet hier potentiell in zweierlei Ausführung Verwendung. Zum einen kann die KNN-Klassifikation zur Eruierung der Klasse des nächsten Vektors im Vektorraum angewendet werden, wobei der Input durch einen Paragraphvektor des übergebenen Textes ausgedrückt wird. Diesem gegenübergestellte Vektoren stellen dabei distribuierte Repräsentationen der mittels Keyword-Extraction-Algorithmen errechneten Topics dar¹⁰. Zum anderen können jene Input-Embeddings in Relation zu allen Vektorrepräsentationen der Texte des gefilterten Datensatzes gesetzt werden, um die maximal repräsentierte Klasse im Umfeld des Input-Embeddings als Entscheidungskriterium heranzuziehen. Die Transformation 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 FastText-

¹⁰ Diese werden im weiteren Verlauf als Topic-Embeddings bezeichnet.

Modell in Wortvektoren überführt und diese mithilfe eines DAN-Modells in Paragraph-Embeddings transferiert.

Zeitgleich bildet diese Paragraphrepräsentation des Eingabetextes den Input zur Einordnung im Rahmen klassischer Textklassifikationsaufgaben. So besteht die Option SVM-Modelle zur Einordnung der Problemstellung auf Basis dieser Embeddings zu verwenden. Additional können die unter Kapitel 4.2 deskribierten RNN-basierten Modelle LSTM und GRU angewendet werden, wobei diese wiederum hinsichtlich der Architekturformen One-to-One und Many-to-One zu differenzieren sind. Hinsichtlich letzterer Struktur ist zu vermerken, dass aufgrund der sequentiellen Verarbeitung von Wörtern durch diese, alle Worte in Form von FastText-Embeddings als Eingabeparameter dienen.

Wie aus diesen Beschreibungen zu entnehmen, werden multiple Algorithmen appliziert, welche jeweils Vorhersagen bzgl. der Zielklasse einer Problembeschreibung berechnen. Zur Zusammenführung dieser Prädiktionen wird das Areal des Ensemble Learning (vgl. Kapitel 4.4) 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. Folglich wird zur Zusammenführung der vorhergesagten Klassenwahrscheinlichkeiten das Prinzip des *Weighted Averaging* angewendet, wobei die Gewichte der einzelnen Prädiktoren durch deren jeweiligen Cohen's-Kappa-Wert bestimmt wird. Exemplarisch wird dies in Abbildung 33 anhand von vier fiktiven Modellen gezeigt.

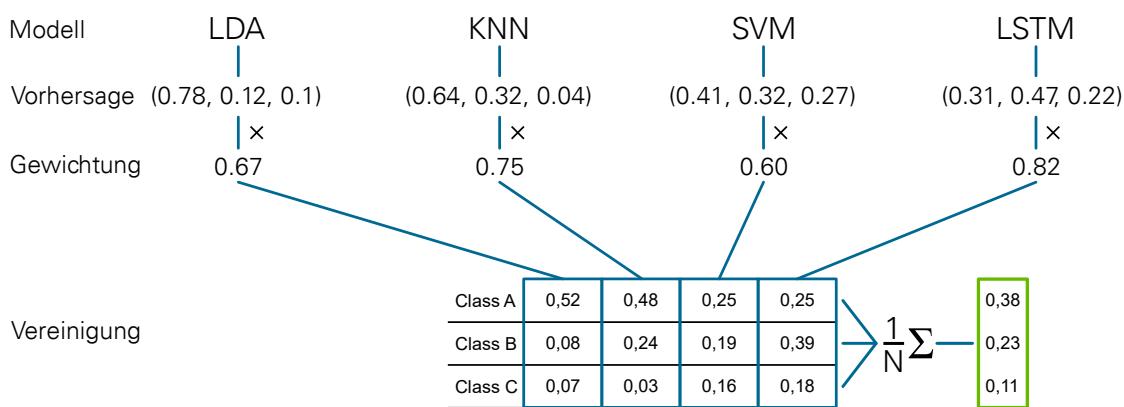


Abbildung 33: Das Weighted-Averaging-Prinzip an einem fiktiven Beispiel (Eigene Darstellung)

Wie der Darstellung zu entnehmen, werden mittels gewichteter Mittelwertbildung alle Vorhersagen der Modelle zusammengeführt, woraus eine finale Vorhersage resultiert.

Extraktion von textuellen Analyseentitäten

Neben dieser Zuweisung im Rahmen einer Klassifikation besteht, wie unter Kapitel 1 bereits aufgezeigt, die Anforderung eine personalisierte, exemplarische Analyse an den Nutzer zurückzugeben. Zu deren Erfüllung wird, die unter Kapitel 4.5 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 Elemente des Satzes aufzeigt. Anschließend werden die relevanten Entitäten mittels regelbasiertem Chunking extrahiert. Im Allgemeinen werden durch regelbasiertes Chunking reguläre Ausdrücke definiert, die die gesuchte Wortfolgenstruktur widerspiegeln. Vor dem Hintergrund der Extraktion von Analyseentitäten 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 den Ausdrücken zu entnehmen werden alle aufeinander folgende Nomen, auf Artikel folgende Nomen sowie die Abfolge von Verb und mindestens einem Nomen herausgefiltert. Abbildung 34 dient zur Illustration eines exemplarischen Resultats einer derartigen Extraktion.

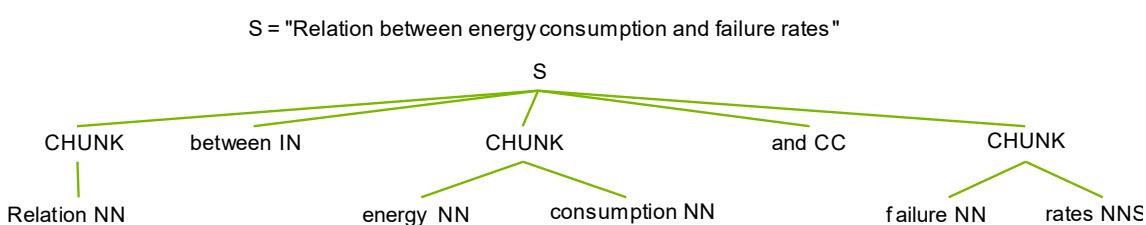


Abbildung 34: Illustration des Vorgehens zur Extraktion der Analyseentitäten (Eigene Darsellung)

In obenstehender Abbildung ist zu erkennen, dass durch dieses Vorgehen nicht lediglich relevante Entitäten extrahiert werden, sondern ebenfalls alleinstehende Nomen des Satzes (hier: „Relation“). Resultierend aus diesem Sachverhalt werden die Entitäten, ähnlich zum Vorgehen von Keyword-Extraction-Methoden, mittels einer Funktion gewichtet und 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 werden. Dies resultiert aus der Annahme, dass der erste Teil eines Satzes einen einleitenden Charakter aufweist, darauf folgend allerdings das Problem näher 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 Gesamtmenge an Wörtern der Problemstellung N gesetzt, sodass sich die folgende Bewertungsformel ergibt.

$$score(c) = \frac{P_c + F_c}{N} \quad (27)$$

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. hybriden NER-Systemen zuordenbar sind und somit eine ausreichend große Datenbasis, mit idealerweise gelabelten Daten, erfordern (vgl. Jurafsky & Martin, 2009, S. 761f., 768ff.). Aufgrund der Inexistenz eines derartigen Datensatzes wird hier das oben beschriebene, regelbasierte Vorgehen zur Extraktion und Bewertung verwendet.

Technische Realisierung

Zur Implementierung dieses Ensembles an Modellen und Architekturen wird das Interface *Model* erstellt, welche als Elternklasse aller oben beschriebenen Klassifikationsmodelle fungiert. So definiert diese Klasse die Mindestfunktionalitäten der einzelnen Modellklassen. Hierbei ist anzumerken, dass zur Initiierung 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 zum Lesen der jeweiligen Modellformate und Initialisieren der Modelle bei jedem Objekt dieses Typs die Methode *initialize()* integriert werden muss. Weiterführend wird zur Vereinheitlichung des Formats der Modellvorhersagen eine Klasse *Prediction* implementiert, mit deren Hilfe eine derartige Standardisierung erreicht werden kann. In die-

sem Rahmen dient weiterhin die Klasse *LabelMap* zum Auslesen der Klartext Verfahrensklasse aus der Modellvorhersage. So verwendet die Modellklasse eine LabelMap, um der Klasse Prediction die korrekten Werte zu übergeben.

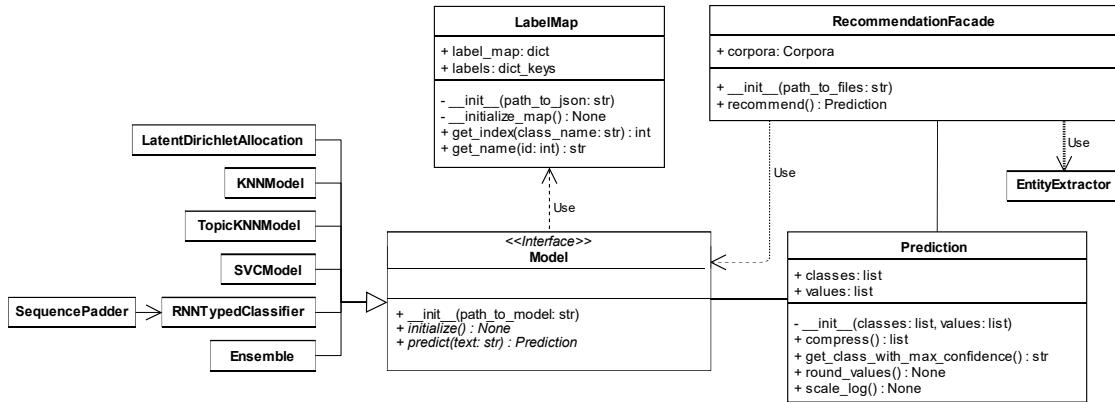


Abbildung 35: Klassenstruktur der benötigten Klassen zur Verfahrenszuweisung (Eigene Darstellung)

In Anschluss an diese Implementierung wird das *Facade-Pattern* aus der Softwareentwicklung verwendet. Dieses Muster erlaubt es generell eine einheitliche Schnittstelle auf einer höheren Ebene zu definieren. Hierbei liegt die Motivation in der Reduktion von Abhängigkeiten im Subsystem sowie der hieraus resultierenden vereinfachten und weniger fehleranfälligen Nutzung des Subsystems durch diese Schnittstelle (vgl. Gamma, Helm, Johnson & Vlissides, 2015, S. 237f.). Eine Implementierung dieses Musters erfolgt in der Klasse *RecommendationFacade*, die die Initialisierung, Anwendung und Zusammenführung aller Klassifikationsalgorithmen steuert. Zudem nutzt diese Fassade die Klasse *EntityExtractor*, die für die Durchführung der oben deskribierten Extraktion von Analyseentitäten zuständig ist.

6.4 Programmstruktur²

Das folgende Kapitel gibt nach der Illustration einzelner Komponenten eine Übersicht über die gesamte Programmstruktur. Im Zuge dessen ist im Anhang in Abbildung A2

das gesamte Klassendiagramm dargestellt¹¹. Zur Verknüpfung des Frontends mit dem Backend wird sich für das Framework *Flask* entschieden, welches eine Implementierung eines Python-Backends in einer webbasierten Technologie erlaubt. Das Frontend wird weiterhin in den unterschiedlichen Sprachen und Frameworks *HTML*, *CSS*, *Jinja*, *Sass*, *Materialize* und *D3.js* angelegt und ermöglicht so die Strukturierung der Webseite sowie die clientseitige, dynamische Anzeige von Berechnungen und Analysen. Weiterhin wird die Technologie *Asynchronous Java Script* (AJAX) und *jQuery* Rahmen einer Middleware verwendet, um einen asynchronen Datenaustausch zwischen Nutzer und System zu ermöglichen ohne die Webapplikation neu laden zu müssen. In Abbildung 36 wird dieser konzeptionelle Aufbau illustriert.



Abbildung 36: Model-View-Controller-Architektur des Prototyps (Eigene Darstellung)

Dementsprechend wird sich in der Entwicklung des Systems an dem Architekturenschema *Model-View-Controller* orientiert. In diesem beschreibt das Model-Objekt das Anwendungsobjekt und umfasst demnach die interne Programmstruktur, ergo das Backend des Systems. Das View-Objekt hingegen repräsentiert die grafische Benutzeroberfläche und inkludiert somit die visuelle Repräsentation des Programms. Letztendlich dient das Controller-Objekt zur Verbindung dieser beiden Objekte und steuert die Reaktionen auf Nutzereingaben im View-Objekt. Der Vorteil dieser Methodik zeigt sich in der Entkopplung von Programmlogik und Benutzeroberfläche, woraus eine höhere Flexibilität und Wiederverwendbarkeit der Software resultiert (vgl. Gamma et al., 2015, S. 30).

¹¹ Additional kann das gesamte Projekt inklusive dem Programmcode unter folgendem Link abgerufen werden: <https://github.com/rsmttud/Recommender-System>.

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

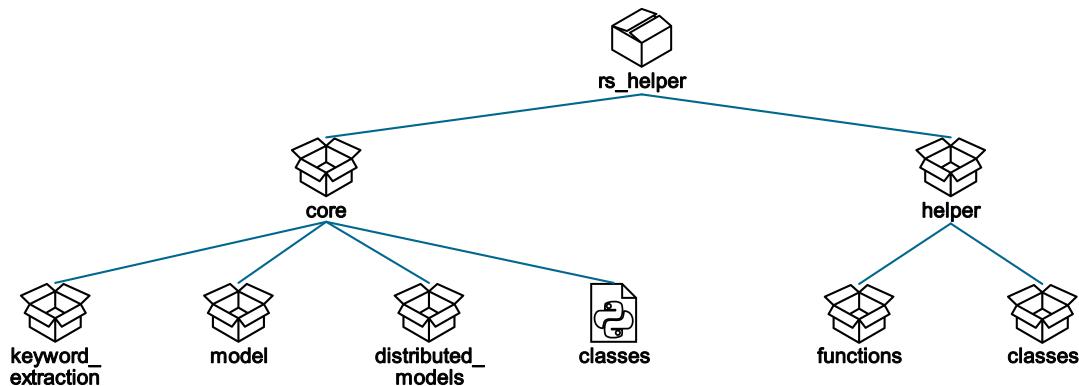


Abbildung 37: Struktur des implementierten Python-Pakets auf Model-Ebene (Eigene Darstellung)

Wie in Abbildung 37 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 ersteres alle generischen Klassen enthält, die hinsichtlich der Lauffähigkeit des Systems essenziell sind. Das Subpaket „*helper*“ hingegen umschließt Methoden und Klassen, welche primär im Rahmen des Trainings sowie der Evaluation, der unter Kapitel 6.3 deskribierten, methodischen Herangehensweisen hinzugezogen werden und fortführend verwendet werden können. Eine vollständige Dokumentation aller Klassen, Strukturen und Methoden findet sich im elektronischen Anhang der Arbeit in der Datei „*documentation.html*“ wieder.

6.5 Anforderungsgegenüberstellung¹

Nach Darlegung der Entwicklung von Front- und Backend erfolgt die Gegenüberstellung der unter Kapitel 6.1 aufgestellten Anforderungen hinsichtlich des Prototyps. Hierbei wird zunächst auf die funktionalen und anschließend auf die nicht-funktionalen Anforderungen eingegangen. In Tabelle 10 sind die Bewertungen der definierten Anforderungen dargestellt.

Tabelle 10: Bewertung der definierten 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 (Cohen's Kappa > 0.5). | ✓ |
| A5 | Extraktion potentieller Analyseentitäten aus Problemstellung. | ○ |
| A6 | Deterministische Zuweisung zu einer Verfahrensklasse. | ✓ |
| A7 | Möglichkeit zur Eingabe einer kurzen sowie langen Problebeschreibung. | ✓ |
| A8 | Entgegenwirken von syntaktischer Divergenz. | ○ |
| A9 | Ausgabe von Wahrscheinlichkeiten der Zuordnung. | ✓ |
| A10 | Einfachheit. | ✓ |
| A11 | Erweiterbare Systemarchitektur. | ✓ |
| A12 | Usability des Systems. | ✓ |
| A13 | Plattformunabhängigkeit. | ✓ |

✓: Gegeben; ○: 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 präferiert, welche durch die Verwendung vollständig trainierter, deterministischer Algorithmen und der Ausgabe von einzelnen Zuweisungswahrscheinlichkeiten nach Aggregation aller Vorhersagen im Rahmen des Ensemble Learning erreicht werden kann. Hierbei ist anzumerken, dass A6 lediglich im Rahmen des Prototyps gegeben ist, allerdings bei Inklusion kollaborativer Aspekte eines Recommender-Systems, beispielsweise durch die 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 erörtert.

Weiterhin beschreibt A8 die Anforderung, dass der unter Kapitel 1.1 dargelegten syntaktischen Divergenz 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, womit dieser Problematik entgegenwirkt wird. Anzumerken ist jedoch, dass diese Architektur nicht Bestandteil aller aufgestellten Vorgehen bzw. Einordnungen ist, sodass diese Anforderung letztendlich als teilweise gegeben bewertet wird.

Neben der Zuweisung einer Klasse fokussiert A3 die Ausgabe einer personalisierten Analyse an den Nutzer, um die berechnete Verfahrensklasse exemplarisch zu illustrieren. Dabei weist die Anforderung eine starke Zusammengehörigkeit mit A5 auf, da die hierbei extrahierten Analyseentitäten in der beispielhaften Analyse visualisiert werden sollen. Betreffend diese Anforderungen zeigt Kapitel 6.3.3 die applizierten Methoden zur Erlangung der nötigen Informationen und Abbildung 38 einen Ausschnitt des Prototyps mit inkludierter Beispielenalyse.

Summary

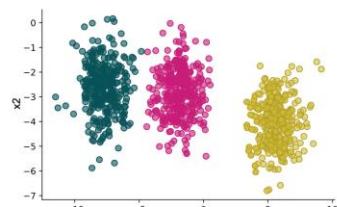
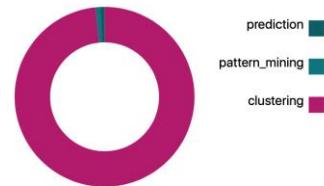
Your Text Inputs:

- = Short Description
- = Long Description

Extracted Potential Features:

error types

Recommendation: Clustering



Clustering

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and



Abbildung 38: Illustration der Ergebnisseite des Prototyps (Eigene Darstellung)

Hierbei werden die potentiellen Analyseentitäten durch ein regelbasiertes Vorgehen im Kontext der NER sowie durch eine Gewichtung der resultierenden Entitäten erlangt. Dennoch ist die 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 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 als auch 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 wird.

In Hinblick auf die nicht-funktionalen Anforderungen ist A11 als relevanteste zu charakterisieren. Durch die Verwendung objektorientierter Programmierung wird eine einfache Erweiterbarkeit des Systems gewährleistet. So besteht die Option neue Modelltypen und -architekturen im Rahmen einer Kindklasse des Interfaces *Model* zu implementieren. Analog können weitere Embedding-Modelle im Paket *distributed_models* als Kindklasse von *EmbeddingModel* aufgenommen werden. Final sichert zudem die Anwendung des *Facade-Patterns* eine 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, sodass eine fehlerfreie Erweiterung erleichtert wird.

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 die Anwendung ebenfalls unabhängig der Plattform durchgeführt werden kann und alle Beziehungen zwischen Software-Paketen bereits vorhanden sind.

Abschließend fokussieren A10 und A12 hingegen die Einfachheit und Nutzbarkeit der Software. Ersteres bezieht sich hierbei auf das anwendungsbezogene Verständnis des Nutzers. Zur Erreichung dieser Anforderung findet eine unkomplizierte Struktur im Frontend der Software Anwendung, welche für den Anwender 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 umfasst, erweist sich in der Bewertung als diffizil. Generell ist die Usability eines Systems von multiplen Faktoren abhängig und beeinflusst. Sie kann daher nicht vollständig im Rahmen der vorliegenden Arbeit untersucht werden, sondern sollte in einer anschließenden Analyse mit einer ausreichenden Anzahl 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. Exemplarisch werden unterschiedliche Bereiche des Systems durch komplexe Datentypen standardisiert, um die Kommunikation zwischen verschiedenen Klassen weniger fehleranfällig zu gestalten. In diesem Sinne werden bei allen implementierten Modell- oder Keyword-Extraction-Klassen die Ausgabe standardisierter Objekte forciert. 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.

7 Evaluation der Methoden sowie des Gesamtsystems²

Das folgende Kapitel gibt eine Übersicht zu den applizierten Methoden der Modell- und Systemevaluation und bildet somit zur Beantwortung der Forschungsfrage III 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 erläutern.

7.1 Evaluation von Embeddings²

Der in Kapitel 3 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 Verfügbarkeit und Relevanz von distribuierten, semantischen Repräsentationen hat unterschiedliche Evaluationsmethoden dieser hervorgebracht. So ist allgemein summieren, dass sich zwei differente Ansätze 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. Erstgenannte messen die Performance anhand sich anschließenden Aufgaben, wie beispielsweise einer Textklassifikation, wohingegen die zweitgenannten versuchen rein linguistische Aspekte zu analysieren (vgl. Schnabel, Labutov, Mimno & Joachims, 2015, S. 298). Dementsprechend erfolgt die nachstehende Evaluierung der distribuierten Repräsentation in zwei Kategorien. Zunächst wird die extrinsische Methodik deskribiert, die überwachte Verfahren des Machine Learning nutzt, um die Qualität der Embeddings zu bewerten (vgl. Bakarov, 2018, S. 3). Dabei bestimmt die resultierende Performance des Verfahrens die Qualität der zugrundeliegenden Embeddings.

Nichtsdestotrotz ist eine rein extrinsische Evaluation nicht ausreichend. Eine themen-spezifische Terminologie sowie die Herausforderung, Korrelationen zwischen den Ergebnissen verschiedener Machine-Learning-Verfahren aufzuweisen, verlangen weitere semantische Evaluierungsmethoden (vgl. Bakarov, 2018, S. 6). Hierbei werden intrinsische Methoden verwendet, um semantische und syntaktische Beziehungen der distribuierten Repräsentationen zu ermitteln (vgl. Schnabel et al., 2015, S. 298).

Im Rahmen der Arbeit steht vordergründig die Funktionalität und Genauigkeit des zu entwickelnden Systems. Aus diesem Grund wird zunächst eine extrinsische Evaluation der Methoden vollzogen, woran sich die Bewertung der Modelle anhand von intrinsischen Methoden anschließt.

7.1.1 Extrinsische Evaluation²

Extrinsische Evaluationsmetriken basieren auf der Anwendbarkeit von Embeddings in überwachten Machine-Learning-Verfahren. Bei einer rein extrinsischen Evaluation, wird angenommen, dass distribuierte Repräsentation eines Verfahrens gute Ergebnisse in nachfolgenden NLP-Anwendungen erzielen (vgl. Bakarov, 2018, S. 3). Beispielsweise haben TSVETKOV ET AL. (2015) extrinsische Evaluationen verwendet, um die Embedding-Performance anhand von Textklassifikationsaufgaben zu messen. Weiterhin zeigen SCHNABEL ET AL. (2015) extrinsische Evaluationen mittels Sentiment-Analysen. Das Spektrum an möglichen NLP-Anwendungen ist generell sehr umfangreich, weshalb aus Komplexitätsgründen nicht auf weitere Verfahren eingegangen wird (vgl. Bakarov, 2018, S. 4ff.).

Für das Auffinden der optimalen Modellkombinationen wird ein Grid-Search¹² ähnliches Vorgehen gewählt. Hierbei wird auf Basis der Kombination verschiedenster Parameterausprägungen jeweils ein separates Embedding-Modell trainiert und anschließend anhand eines überwachten Klassifikationsverfahrens evaluiert. Die Parameterkombination

¹² Grid-Search ist eine der klassischen Methoden, um die idealen Hyperparameter eines Modells zu identifizieren. Dabei bedient sich das Verfahren der Exhaustionsmethode (engl.: *Brute-Force*), um auf Basis manuell erstellter Parameterkonfigurationen die beste Konfiguration zu erkennen (vgl. Bergstra & Bengio, 2012, S. 282ff.).

mit der höchsten Modellgenauigkeit fungiert somit als Ausgangsbasis für das finale Modelltraining. Das zugrundeliegende Klassifikationsverfahren stellt hierbei eine multinomiale Support Vector Machine dar, die sowohl mit einem linearen als auch einem radialen Kernel und einem C-Wert von 12 trainiert wird. Der Gamma-Wert der SVM-Modelle hingegen wird durch die Anzahl der Features N und der Standardabweichung des Datensatzes bestimmt und wie folgt berechnet:

$$\gamma = \frac{1}{N * \sqrt{\frac{1}{N} * \sum_{i=1}^n (x_i - \bar{x})^2}} \quad (28)$$

Die Datengrundlage bilden hierbei die in Kapitel 5.4 deskribierten, gefilterten Daten. Für die Operationalisierung der Ergebnisse wird sich der Modellgenauigkeit bedient. Es ist zu betonen, dass die SVM als Klassifikator rein arbiträr gewählt wird und jedes andere Klassifikationsverfahren ebenfalls zur extrinsischen Evaluation geeignet ist. Zudem ist anzumerken, dass im Rahmen der Arbeit zunächst 100-dimensionale FastText-Modelle trainiert werden, um Wort-Embeddings zu erhalten, die weiterführend als Input für das Training der DAN-Modelle fungieren. Somit werden die DAN-Modelle auf Basis der besten FastText-Modelle trainiert.

FastText

In Bezug auf das oben erläuterte Grid-Search-Vorgehen ergeben sich insgesamt 36 verschiedene Modellmöglichkeiten. Aufgrund der Rechenintensität der möglichen Modellkombinationen wird sich auf vier essentielle FastText-Parameter beschränkt. In der nachfolgenden Abbildung werden ausgewählte Korrelationen zwischen den Modellparametern sowie den resultierenden Modellgenauigkeiten illustriert.

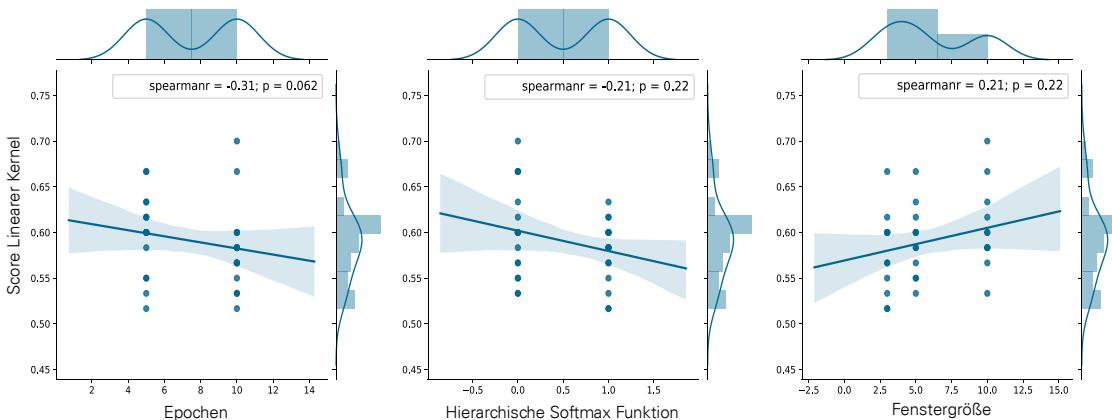


Abbildung 39: Ausgewählte Korrelationen von Modellparametern (FastText) (Eigene Darstellung)

Auf Basis der Evaluationsergebnisse der trainierten FastText-Modelle, lässt sich erkennen, dass eine höhere Filtergröße der Embedding-Modelle zu höheren Modellgenauigkeiten führt. Darüber hinaus deuten die Ergebnisse darauf hin, dass die Verwendung einer hierarchischen Softmax-Funktion zu geringeren Modellgenauigkeiten führt. Aufgrund der trainierten Modellmenge sind diese Korrelationen allerdings nicht belastbar und dienen lediglich zur Orientierung in der vorliegenden Arbeit.

Gemäß der distributionellen Hypothese sollten semantisch ähnliche Wörter bzw. Paragraphen im Vektorraum näher beieinanderliegen als unähnliche (vgl. Zhang et al., 2016, S. 5). In der nachfolgenden Abbildung werden mithilfe der Paragraphvektoren des gefilterten Datensatzes einer klassischen Kompositionsfunktion illustriert.

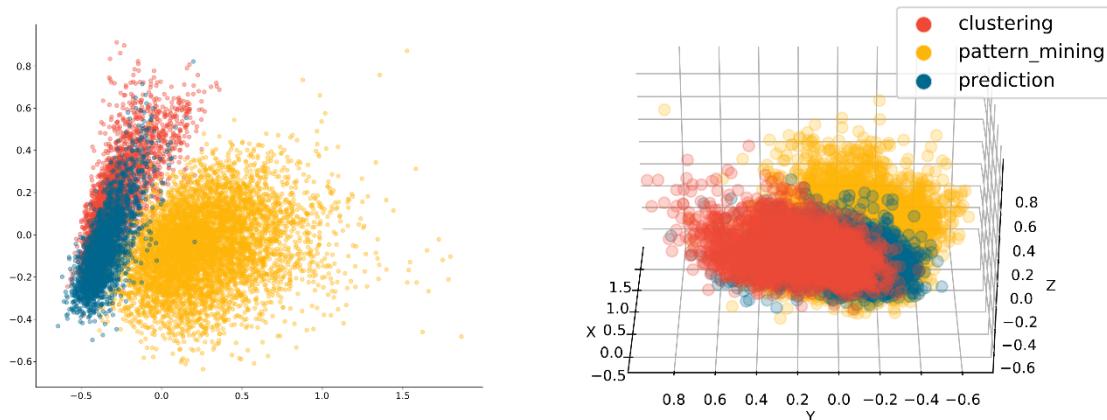


Abbildung 40: Streudiagramme von FastText-Embeddings der gefilterten Daten (Eigene Darstellung)

Wie zu erkennen, erfolgt zwar eine semantische Diskriminierung der Daten zwischen den Klassen, allerdings gibt es starke semantische Überschneidungen zwischen den Klassen *Clustering* und *Prediction*. Die semantische Aussagekraft wird im Folgekapitel mittels geeigneter intrinsischer Evaluationsmetriken untersucht.

Deep-Averaging-Networks

In Anbetracht 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 werden Korrelationen zwischen diesen berechnet. So zeigen sich nahezu keinerlei Korrelationen zwischen den eigentlichen Hyperparametern des Modells. Allerdings ist eine sehr leicht negative Korrelation zwischen der Epochenzahl und der Modellgenauigkeit

(`valid_score_linear_kernel`) sowie zwischen der Anzahl an Hidden-Layern und der Genauigkeit zu konstatieren (vgl. Abbildung 41).

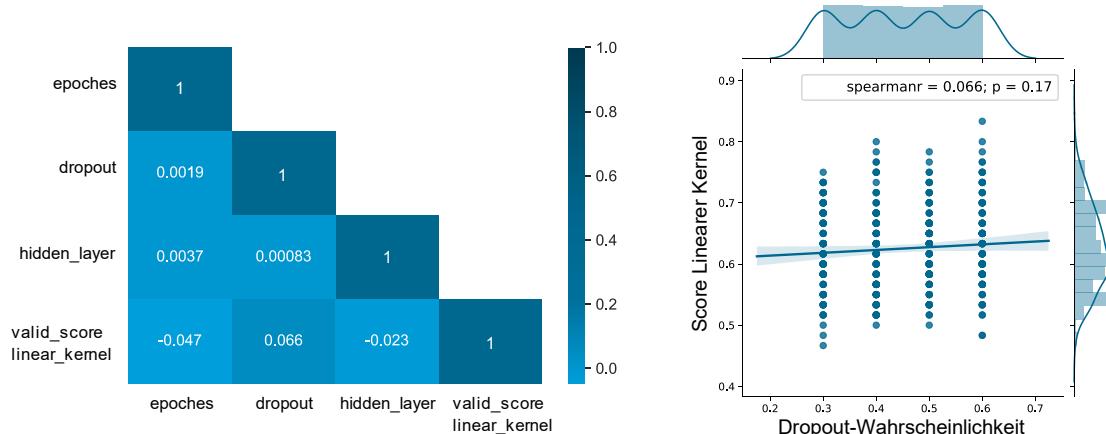


Abbildung 41: Ausgewählte Korrelationen von Modellparametern (DAN) (Eigene Darstellung)

Wie die obige Korrelationsmatrix veranschaulicht, besteht zudem eine sehr geringe Korrelation zwischen der Dropout-Wahrscheinlichkeit und den Modellergebnissen. Zwar zeigt das beste Modell die höchste Dropout-Wahrscheinlichkeit, allerdings besteht eine Schwankung in der Güte hinsichtlich aller Werte dieses Parameters, wodurch im Rahmen der Untersuchung ein lediglich tendenzieller Zusammenhang festgestellt werden kann. Dies unterstreicht teilweise die Ergebnisse von IYYER ET AL. (2015), die bessere Modellgenauigkeiten durch höhere Wort-Dropout-Wahrscheinlichkeiten erzielen. Zurückführen lässt sich dies, auf eine potentiell bessere Generalisierungsfähigkeit der resultierenden distribuierten Repräsentationen.

Einer der elementaren Gründe, warum DANs ihre Anwendung in der vorliegenden Arbeit finden, ist die Fähigkeit syntaktische Divergenzen zu überwinden (vgl. Iyyer et al., 2015, S. 1687). Die folgende Abbildung illustriert beispielhaft die differenzierenden Eigenschaften eines DANs. Es werden die besten Modelle, kategorisiert nach ihrer Layeranzahl, von links nach rechts dargestellt. Dabei wird sich nur auf die Schichten bezogen, die zwischen dem Average-Layer und den sich anschließenden, vollvernetzten Klassifikationslayern liegen (vgl. Kapitel 4.1.2).

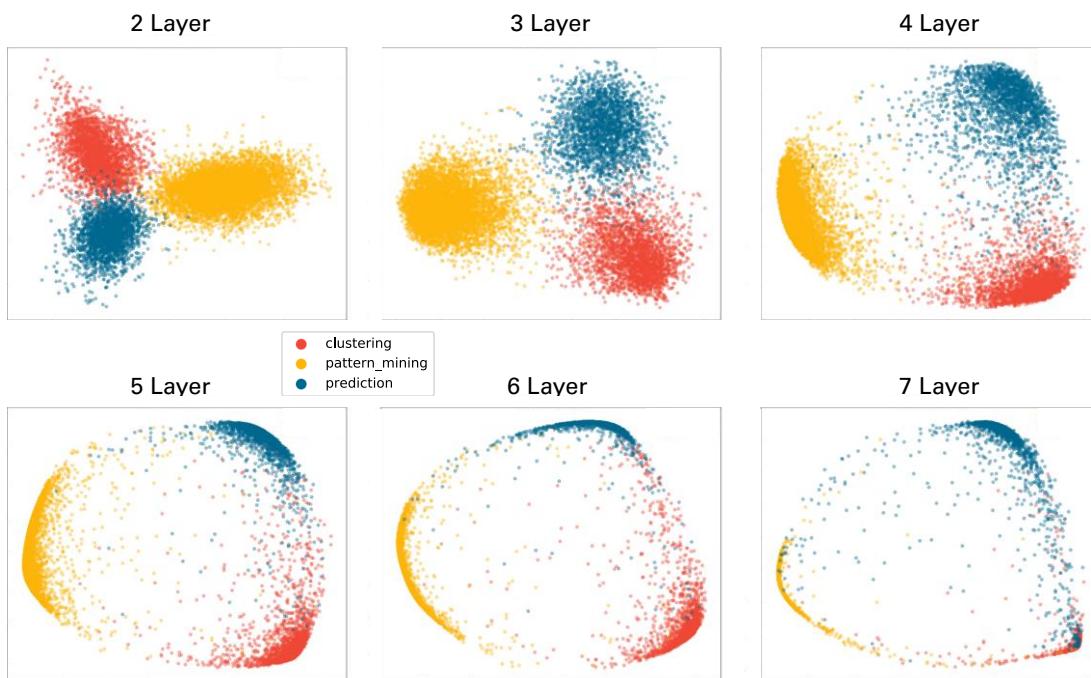


Abbildung 42: Deep-Averaging-Netzwerke mit multipler Layer-Anzahl (Eigene Darstellung)

Im Vergleich zu Abbildung 40 ist zu erkennen, dass die Klassen durch das Verwenden eines zusätzlichen DAN-Modells besser separiert werden. Dies kann sich als gewinnbringend für nachfolgende Klassifikationsverfahren erweisen. Hinzukommt, dass die Streudiagramme die Erkenntnisse aus der Korrelationsanalyse unterstreichen. So separieren Modelle mit wenigen Hidden-Layern, vor den Klassifikations-Layern, stärker in sich geschlossene, konvexe Cluster mit wenigen semantischen Ausreißern. Wohingegen Modelle mit einer hohen Hidden-Layer-Anzahl mehr konkave Gruppen bündeln und auf den ersten Blick mehr semantische Ausreißer beinhalten. Die Anzahl an potentiellen semantischen Ausreißern kann sich anschließende überwachte Verfahren des Machine Learning negativ beeinflussen, weshalb deren Vermeidung in der Erstellung von DANs relevant ist (vgl. Rehm et al., 2007, S. 489).

7.1.2 Intrinsische Evaluation¹

Im Rahmen der intrinsischen Evaluation finden differente Kennzahlen Anwendung, welche Aussagen über die Qualität der jeweiligen Modelle liefern ohne diese in speziellen Applikationen anzuwenden. Zunächst wird zu dieser Bewertung die sog. *Purity* (z.Dt.

Reinheit) der, durch die Anwendung des Embedding-Modells auf die Ausgangsdaten erlangten, Cluster fokussiert. Die Intention bei einer derartigen Kategorisierung von Wörtern ist es, distribuierte Repräsentationen ihren zugrundeliegenden Gruppen durch Clusteringalgorithmen zuzuordnen. Jene Repräsentationen werden anhand eines Clusteringmodells in n verschiedene Gruppen unterteilt, wobei n die Anzahl der natürlich zugrundeliegenden Wortkategorien des Korpus darstellt. Die letztendliche Qualität der Embeddings wird mithilfe einer Clustering-Evaluationsmetrik bestimmt (vgl. Baroni, Dinu & Kruszewski, 2014, S. 241 und Schnabel et al., 2015, S. 301). Die Herausforderung besteht hierbei zum einen in der Auswahl des richtigen Clusteringverfahrens, zum anderen im Finden einer geeigneten Evaluationsmetrik (vgl. Bakarov, 2018, S. 12). Besonders in Hinsicht der hochdimensionalen Daten ist das zu wählende Clusteringverfahren von essentieller Bedeutung, um die Funktionalität der Clusteringalgorithmen und somit die Belastbarkeit der Ergebnisse zu gewährleisten (vgl. Pavithra & Parvathi, 2017, S. 2895ff.).

Im Rahmen der vorliegenden Arbeit wird sich auf ein partitionierendes Verfahren konzentriert. Der Vorteil eines partitionierenden Verfahrens liegt in der notwendigen, initialen Bestimmung der Clusteranzahl, welche hier durch die Menge der abgezielten Verfahrensklassen, d.h. drei, bestimmt werden kann und somit additional erlaubt die Unterscheidbarkeit der Klassen im Vektorraum einzuschätzen. Zur Ausführung wird sich des KMeans-Algorithmus bedient, da er simpel in seiner Anwendung und weniger rechenintensiv als andere partitionierende Verfahren ist (vgl. Bhukya & Ramachandram, 2010, S. 46ff.).

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

$$purity(C, K) = \frac{1}{N} \sum_k \max_j |c_k \cap k_j| \quad (29)$$

Mit $C = \{c_1, c_2, \dots, c_K\}$ als Menge an extrahierten Clustern und $K = \{k_1, k_2, \dots, k_j\}$ als Menge der zugrundeliegenden Klassen 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. Sätze, die sich unterschiedlichen Verfahrensklassen zuordnen lassen, sollten dabei eine geringe Ähnlichkeit zueinander aufweisen. Abbildung 43 zeigt eine derartige Matrix, welche für jedes trainierte Modell berechnet wird.

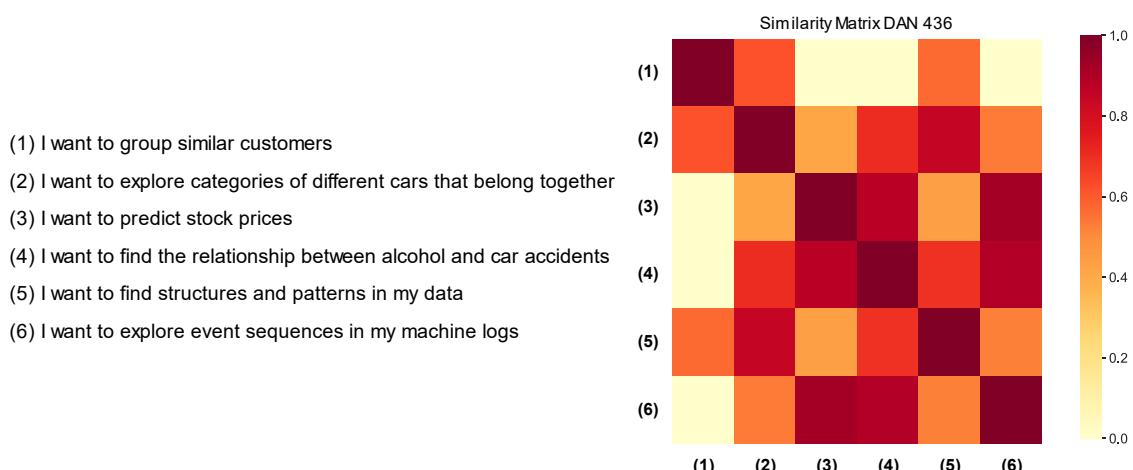


Abbildung 43: Ähnlichkeitsmatrix zur Evaluierung von Embedding-Modellen (Eigene Darstellung)

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. Im Anschluss hieran werden diesen Werten die mittels des Modells errechneten Werte gegenübergestellt, um sowohl den Mean Error *ME* (z.Dt.: gemittelter Fehler) als auch die Spearman-Rangkorrelation r_{SP} der Ähnlichkeitswerte nach den folgenden Formeln zu errechnen. Diese können weiterführend zur Bewertung der Modelle herangezogen werden (vgl. Bojanowski et al., 2017, S. 139ff.).

$$ME = \sum_{i=1}^N \frac{x_i - \hat{x}_i}{N} \quad (30)$$

$$r_{SP} = \frac{\sum_{i=1}^N ((rg_{x_i} - \bar{rg}_x)(rg_{y_i} - \bar{rg}_y))}{\sqrt{\sum_{i=1}^N (rg_{x_i} - \bar{rg}_x)^2} \sqrt{\sum_{i=1}^N (rg_{y_i} - \bar{rg}_y)^2}} \quad (31)$$

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 fortführend angewendeten Klassifikationsalgorithmen.

7.1.3 Ergebnisse der Evaluation¹

Insgesamt werden im Rahmen der vorliegenden Untersuchung 36 FastText-Modelle sowie 480 Deep-Averaging-Netzwerke trainiert und gegenübergestellt. Wie oben bereits erwähnt, werden aufbauend auf den Ergebnissen der Evaluation der FastText-Modelle, verschiedene DAN-Modelle mit unterschiedlichen Hyperparametern trainiert, weshalb das allgemeine Vorgehen der Embedding-Evaluation als Zyklus zu verstehen ist. Der Umfang dieser Trainingsschritte sowie deren Evaluation resultiert aus der Inferenz von Paragraph-Embeddings mittels des DAN-Modells auf Basis von Word-Vektoren des FastText-Modells sowie dem Sachverhalt, dass resultierende Embeddings ausschlaggebend für die Qualität weiterer Methoden im Bereich der Klassifikation sind. Die folgende Tabelle zeigt eine Übersicht der verwendeten Hyperparameter, welche aufgrund deren jeweiligen Kombinationsmöglichkeiten in die Anzahlen der trainierten Modelle münden.

Tabelle 11: Übersicht der Parameterkombinationen für Embedding-Modelle

| Parameter | Mögliche Werte |
|---------------------|----------------|
| FastText | |
| Epochen | [5, 10] |
| Minimale Häufigkeit | [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 extrinsischen Evaluation bewertet werden, wobei extrinsischen Ergebnissen eine höhere Relevanz zugestanden wird. In Tabelle 12 werden die Ergebnisse der Evaluation der einzelnen FastText-Modelle am Beispiel der vier besten Modelle illustriert, 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 interner Struktur, damit ausgewählte Modelle keine starken, internen Inkonsistenzen aufweisen.

Tabelle 12: Ergebnisübersicht der Ähnlichkeitsevaluation von Embedding-Modellen

| Modell | EP | WS | MC | HS | SVM (linear) | SVM (RBF) | P | ME | r_{SP} |
|--------------------|----|----|----|----|--------------|--------------|--------------|--------------|--------------|
| <i>fasttext_31</i> | 10 | 10 | 5 | 0 | 0.700 | 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 |
| <i>fasttext_13</i> | 5 | 10 | 5 | 0 | 0.633 | 0.650 | 0.754 | 0.224 | 0.342 |

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

Ersichtlich aus der Tabelle erzielen die Modelle *fasttext_31* und *fasttext_12* die besten Ergebnisse in Klassifikationsaufgaben, wohingegen in den intrinsischen Kennzahlen lediglich sehr geringe Unterschiede zu verzeichnen sind.

Anschließend erfolgt auf Basis der Auswahl der fünf besten 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 sowie die der

intrinsischen zur Kontrolle der internen Struktur herangezogen. Mit Tabelle 13 wird eine Übersicht zu den besten Modellen hinsichtlich deren Generalisierungskraft gegeben¹³.

Tabelle 13: Übersicht der Evaluationsergebnisse von DAN-Modellen

| Modell | FastText | EP | DR | SH | HL | SVM (linear) | SVM (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 | 0.750 | 0.971 | 0.332 | 0.205 |
| DAN_279 | fasttext_31 | 50 | 0.6 | [200] | 4 | 0.800 | 0.700 | 0.973 | 0.302 | 0.099 |
| DAN_246 | fasttext_12 | 10 | 0.4 | [200] | 4 | 0.800 | 0.733 | 0.967 | 0.329 | 0.180 |
| DAN_47 | fasttext_12 | 10 | 0.4 | [100, 50] | 2 | 0.783 | 0.700 | 0.958 | 0.304 | 0.333 |

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

Dementsprechend kann durch die Anwendung der DAN-Architektur ein Unterschied in der Modellgüte von ca. 13% erreicht werden. Zeitgleich ist ein Anstieg der Purity und der damit verbundenen Zusammengehörigkeit der Klassen im Vektorraum von ca. 21% zu verzeichnen. Konträr 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. Die Wahl des FastText-Modells resultiert hierbei aus der vergleichsweise hohen Generalisierungsfähigkeit des Modells sowie aus dem Sachverhalt, dass die besten der trainierten DAN-Modelle auf Basis dieses Modells agieren. Das Modell *DAN_436* erweist sich zudem als bestes Modell für die anschließende Klassifikationsaufgabe.

¹³ Eine vollständige Übersicht der trainierten FastText- und DAN-Modelle findet sich im elektronischen Anhang.

7.2 Evaluation von Topic-Modellen²

Das folgende Kapitel dient zur Darstellung der verwendeten Evaluationsmethoden sowie deren Ergebnissen 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 zu diesen allerdings nicht erfasst werden kann. Ähnlich zum Areal der Embeddings besteht somit in der Literatur eine Vielzahl an Kennzahlen, die zur Bewertung der berechneten Topics herangezogen werden können (vgl. Newman, Lau, Grieser & Baldwin, 2010). Diese fokussieren allerdings die Ähnlichkeit der Wörter in einem Topic und bewerten folglich dessen Kohärenz. Eine Evaluation der Allgemeingültigkeit und korrekten Abbildung des erwarteten semantischen Konstrukts können daher nur indirekt vorgenommen werden.

7.2.1 Überblick der Evaluationsmethoden²

NEWMAN ET AL. (2010) geben in ihrer Publikation eine Übersicht zu möglichen 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. Newman et al., 2010, S. 102ff.). Die Hauptunterschiede der Methoden bestehen in der Bewertungsart der Wortähnlichkeiten. So spezifizieren LEACOCK, CHODOROW & MILLER (1998) die Verwendung der WordNet-Ontologie zur Berechnung des kürzesten Pfads zwischen zwei Wörtern $sp(w_1, w_2)$, welche anschließend in Relation mit der maximalen Tiefe von WordNet D gesetzt und logarithmiert wird:

$$S(w_1, w_2) = -\log\left(\frac{sp(w_1, w_2)}{2 * D}\right) \quad (32)$$

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 WordNet-Ontologie. Demzufolge wird folgende Formel zur Berechnung der Ähnlichkeit herangezogen:

$$S(w_1, w_2) = \frac{2 * \text{depth}(\text{LCS}_{w_1, w_2})}{\text{depth}(w_1) + \text{depth}(w_2) + 2 * \text{depth}(\text{LCS}_{w_1, w_2})} \quad (33)$$

Weiterhin zeigt SCHÜTZE (1998) ein Verfahren zur Generierung von Vektoren aus dem Kontext der Wörter im Korpus, um diese zur Ähnlichkeitsbewertung 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 mitteln¹⁴.

Darüber hinaus zeigen 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* $raco_{w_i, w_{i+1}}$ 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.

$$raco_{w_i, w_{i+1}} = \frac{|(\cup_{l \in ol(w_i)} cat(l)) \cap (\cup_{l \in ol(w_{i+1})} cat(l))|}{|(\cup_{l \in ol(w_i)} cat(l))| + |(\cup_{l \in ol(w_{i+1})} cat(l))|} \quad (34)$$

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

Ferner 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 der vorliegenden Arbeit 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

¹⁴ Dieses Vorgehen wird im Folgenden als FastText-Kohärenz bezeichnet.

Extraction anhand der Qualitätskennzahlen Precision, Recall sowie F1-Score, um zu evaluieren, wie viele der im Korpus enthaltenen Schlüsselwörter durch den jeweiligen Algorithmus extrahiert werden. Aufgrund der Notwendigkeit gelabelter Daten für eine derartige Evaluation und deren Inexistenz im Rahmen der aktuellen Untersuchung wird sich jedoch lediglich auf die Kohärenz der extrahierten Topics fokussiert.

Daneben besteht im Kontext 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 Perplexity (z.Dt.: Perplexität) sowie der Log-Likelihood angewendet. Da der Wert der Perplexity bei steigender Log-Likelihood des Testdatensatzes monoton absinkt, wird im Folgenden lediglich ersteres Maß erläutert. Perplexity beschreibt das Inverse der wortspezifischen, logarithmierten Wahrscheinlichkeit eines ungesesehenen Testdatensatzes, welche mit der Anzahl der Wörter N im Testdatensatz normalisiert ist (Blei, Ng & Jordan, 2003, S. 1008).

$$\text{Perplexity} = \exp \left\{ -\frac{\sum_{i=1}^M \log p(w_i)}{\sum_{i=1}^M N_i} \right\} \quad (35)$$

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

7.2.2 Ergebnisse der Evaluation von Topic-Modellen¹

Nachfolgend werden die Ergebnisse der erstellten Topic-Modelle dargestellt, wobei dahingehend eine Unterteilung in auf Keyword Extraction fundierende Topics sowie in trainierte Modelle des Topic Modeling erfolgt.

Generell werden im Rahmen der aktuellen Untersuchung die sechs vorgestellten und deskribierten Keyword-Extraction-Algorithmen TF-IDF, TF-IGM, TextRank, YAKE, PositionRank sowie TopicRank angewendet. Hierbei werden jegliche Kombinationen der Algorithmen angewendet, woraus insgesamt 61 differente Topics pro zugrundeliegender

Zielkategorie (*Clustering*, *Prediction* sowie *Pattern Mining*) resultieren, die gegenübergestellt und evaluiert werden, um auf die final zu verwendende Kombination zu schließen. Die folgende Tabelle gibt eine Übersicht der durchgeföhrten Evaluationen auf dem gefilterten Datensatz anhand der besten Kombinationen, wobei die Summe der differenten Kohärenzkennzahlen pro Topic zur Auswahl der adäquaten Kombination verwendet wird.

Tabelle 14: 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.

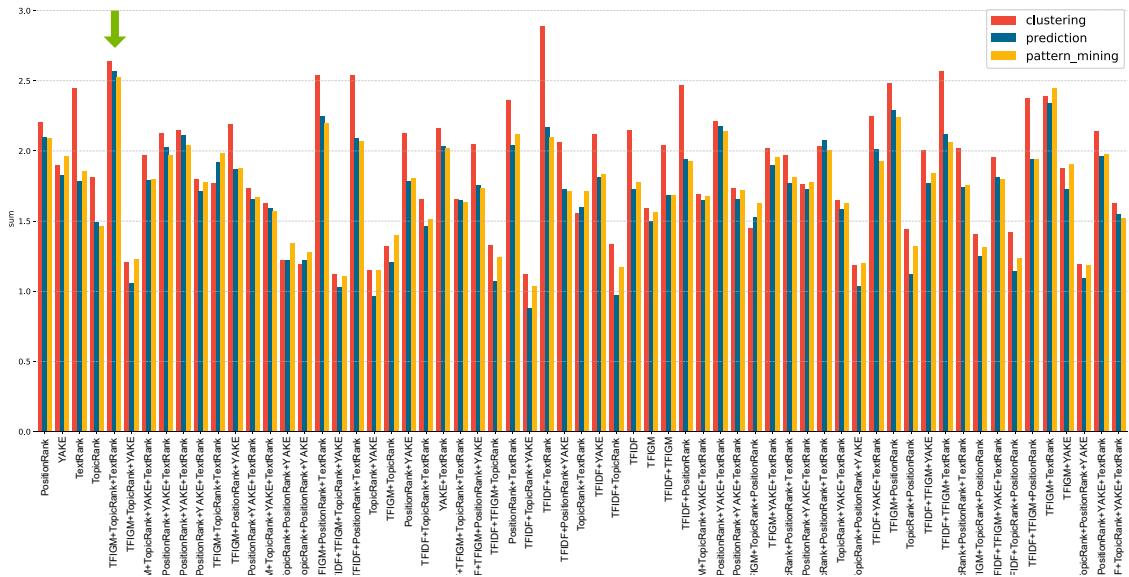


Abbildung 44: Illustration der Evaluationsergebnisse der Keyword Extraction (Eigene Darstellung)

Als Resultat dieser Analyse ist festzustellen, dass die Kombination der Algorithmen TF-IGM, PositionRank und TextRank die höchste Kohärenz bzgl. aller drei generierten Topics aufweist (Grüner Pfeil in Abbildung 44) und somit für den Aufbau der Wissensbasis Verwendung findet. Eine Darstellung der Ergebnisse auf Basis des augmentierten Datensatzes findet sich im Anhang in Tabelle A2 und Abbildung A1.

Hinsichtlich der Evaluierung von LDA-Modellen wird sich der Methodik des Grid-Search bedient, welche auf Basis gegebener Kombinationsmöglichkeiten von 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 vorgenommen, um die jeweilige Trainingsdatenmenge nicht zu stark zu reduzieren. Diese Angaben stützen sich auf die Annahme, dass die Wahrscheinlichkeit für das Vorhandensein nur eines Topics pro Dokument hoch ist. Resultierend werden insgesamt 600 LDA-Modelle trainiert, aus welchen das mit den besten Testergebnissen ausgewählt wird. Weiterhin wird der Abstract-Datensatz verwendet, um die Wahrscheinlichkeit mehrerer Topics pro Dokument in Referenz zum definitionsähnlichen, gefilterten Datensatz zu erhöhen. Aufbau-

end auf dieser Evaluation werden, analog zur Evaluation der Embedding-Modelle, Korrelationen der Modellparameter untersucht. In Abbildung 45 werden diese Korrelationen dargestellt, wobei sich eine negative Korrelation der Parameter Alpha und Beta zur resultierenden, gemittelten Testbewertung herauskristallisiert. Diese Korrelation unterstützt dabei die oben angeführte Annahme zur Topic-Verteilung in Dokumenten.

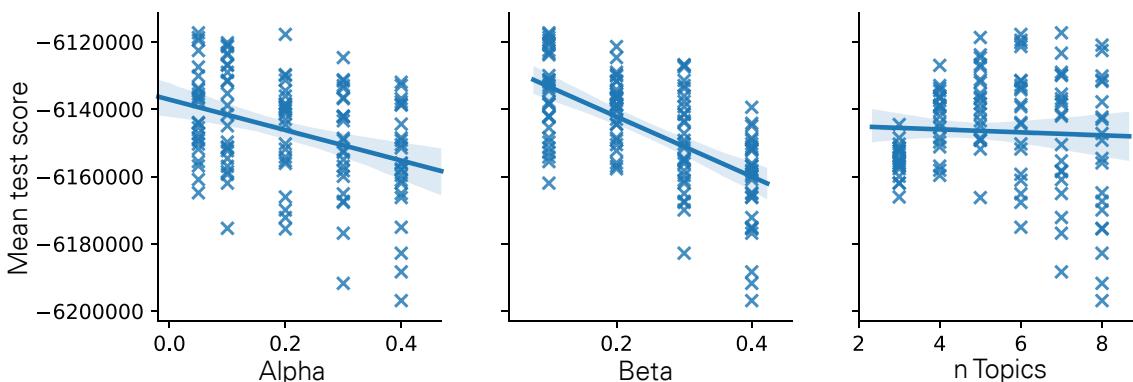


Abbildung 45: Parameterkorrelationen der LDA-Modelle mit $n > 3$ (Eigene Darstellung)

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 Topic-Verteilung in einem Dokument zu trainieren, um eine Zuweisung des Dokumentes bzw. der Problemstellung zu einer Verfahrensklasse zu ermöglichen. An dieser Stelle ist auf das folgende Kapitel zu verweisen, das die Evaluation von Klassifikationsmodellen thematisiert.

Aufgrund der ebenfalls möglichen Verwendung von LDA zur direkten Zuweisung von Verfahrensklassen wird aufbauend auf diesem Ergebnis erneut die Methode des Grid-Search benutzt, um ein Modell zu trainieren, welches über drei Topics verfügt. Oppositionell zu vorheriger Operation wird hierzu das gefilterte Datenset verwendet, da auf Topic abgezielt wird, welche möglichst optimal das jeweilige semantische Konstrukt der Verfahrensklassen abbilden. Aufgrund des definitionsähnlichen Inhaltes dieses Datensatzes und der damit einhergehenden geringeren Menge an Rauschen in den Daten

werden bessere Ergebnisse erwartet¹⁵. 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, woraus final 100 differente LDA-Modelle resultieren.

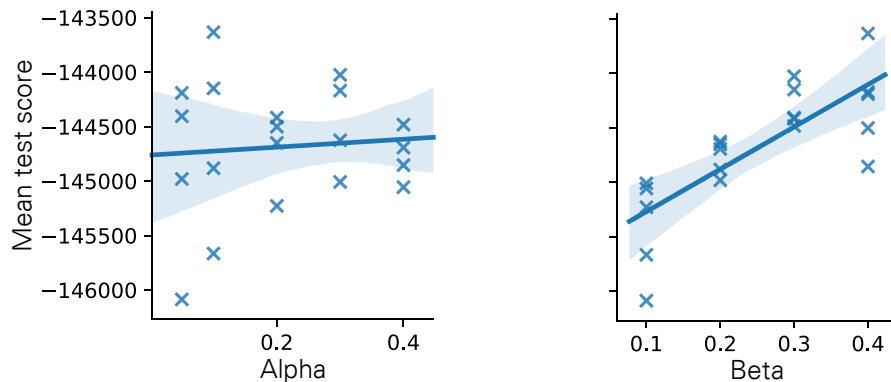


Abbildung 46: Parameterkorrelationen der LDA-Modelle mit n=3 (Eigene Darstellung)

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 Modells hat. Final werden die Parameter Alpha und Beta mit den Werten 0.1 bzw. 0.4 gewählt.

Neben der Auswahl der passenden Modelle werden diese hinsichtlich der Unterscheidbarkeit der Topics untersucht (vgl. Abbildung 47).

¹⁵ Additional wird ebenfalls zur Bekräftigung dieser Aussage ein LDA-Modell auf dem Abstract-Datensatz mit drei Topics trainiert. Dieses Modell erreicht allerdings im Rahmen einer Klassifikation lediglich eine Modellgüte von 35.55%.

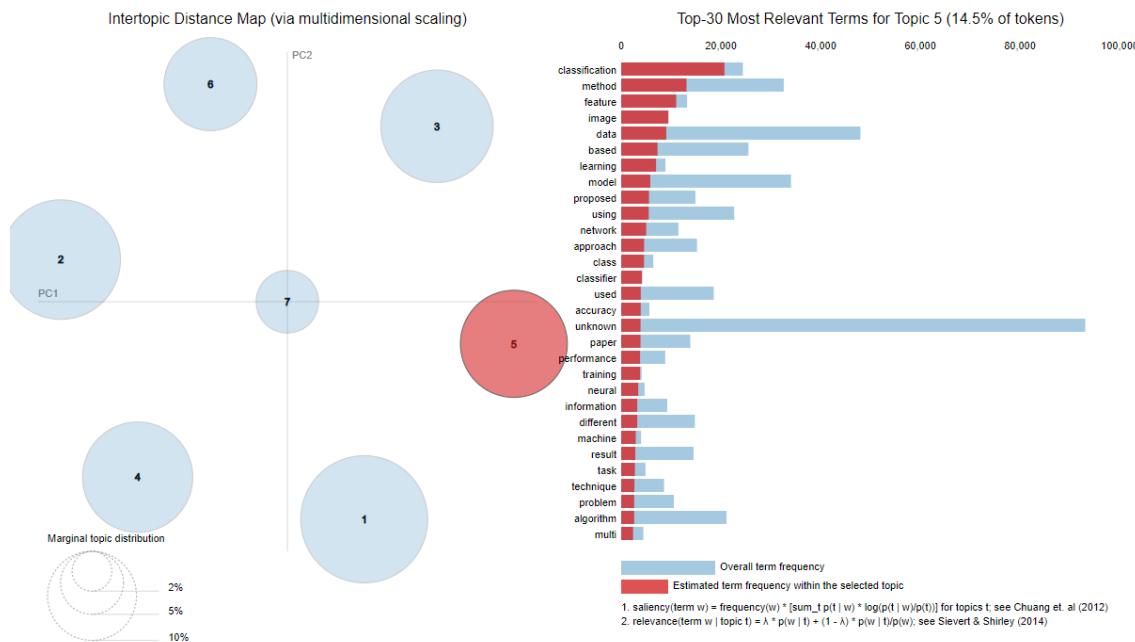


Abbildung 47: Visualisierung eines LDA-Modells mittels pyLDAvis

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 illustriert¹⁶. Ergänzend wird in nachfolgender Abbildung eine Übersicht zu den Topics der beiden erstellten LDA-Modelle mit den jeweils zehn wahrscheinlichsten Wörtern pro extrahiertem Thema gegeben.

| LDA 3 Topics | | | LDA 7 Topics | | | | | | | | | |
|----------------|------------------|----------------|------------------|-------------|-----------------|--------------|-----------------|--------------|------------------|---------|--|--|
| Topic 0 | Topic 1 | Topic 2 | Topic 0 | Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 | Topic 6 | Topic 7 | | |
| Clustering | Prediction | Pattern Mining | Classification | Medical | Geographical | Clustering | Medical Pattern | Prediction | 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 48: Übersicht von Schlüsselwörtern der trainierten LDA-Modelle (Eigene Darstellung)

¹⁶ Da Abbildung 47 eine Momentaufnahme einer interaktiven Darstellung ist, kann diese Zuordnung hier nur bedingt gezeigt werden. Die vollständige Visualisierung findet sich jedoch im elektronischen Anhang der Arbeit in den Dateien „pyldavis_{3_topic, 7_topic}.html“ wieder.

Wie hieran zu erkennen, inkludiert das Modell mit den drei Topics die relevanten Begriffe der einzelnen Verfahrensklassen und erlaubt so dessen Verwendung zur direkten Klassifikation mittels extrahierter Topic-Verteilungen. Im Modell mit sieben Topics hingegen liegen diese Kernbegriffe verstreut vor. 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 Klassifikationsmodelle¹

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

7.3.1 Überblick multinomialer Evaluationsmetriken¹

Wie in Kapitel 1.1 deskribiert, 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 deren Evaluation in binäre, multinomiale und hierarchische Metriken, um die Qualität des Modells zu bewerten. Im Zuge der binären Klassifikation werden die Eingabedaten in ausschließlich zwei disjunkte Klassen ($C_1 \neg C_2$) unterteilt. Die resultierende Performance eines Klassifikators wird häufig in Form einer binären Konfusionsmatrix dargestellt, aus welcher differente Evaluationsmetriken abgeleitet werden können. Im Falle der multinomialen Klassifikation wird zwar analog zur binären ebenso ein Eingabedatenpunkt x_i einer Klasse C_n zugeordnet, jedoch stehen bei der Zuordnung von x_i mehr als zwei Klassen zur Verfügung. Hierbei fundieren die Evaluationsmetriken auf einer multidimensionalen Konfusionsmatrix, welche bestimmte Evaluationsmetriken der binären Klassifikationsevaluation ausschließen (vgl. Sokolova & Lapalme, 2009, S. 427ff.). 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 verdreifacht dies allerdings den Trainingsaufwand, weshalb aus Komplexitätsgründen nur auf ausgewählte Evaluationsmetriken der multinomialen Klassifikation eingegangen wird. Hinzukommt, dass Evaluationsmetriken der hierarchischen Klassifikation ebenfalls unberücksichtigt bleiben, da diese keine Relevanz für die vorliegende Arbeit aufweisen. Die nachfolgende Tabelle illustriert die von SOKOLOVA & LAPALME (2009) zusammengetragenen Evaluationsmetriken der multinomialen Klassifikation.

Tabelle 15: Überblick multinomialer Evaluationsmetriken (in Anlehnung an SOKOLOVA & LAPALME (2009))

| Metrik | Berechnung | Fokus |
|--------------------|--|---|
| Average Accuracy | $\frac{\sum_{i=1}^N \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{N}$ | Durchschnittliche Modellgenauigkeit eines Klassifikators. |
| Error Rate | $\frac{\sum_{i=1}^N \frac{fp_i + fn_i}{tp_i + fn_i + fp_i + tn_i}}{N}$ | Durchschnittliche Missklassifikationsrate eines Klassifikators. |
| Precision $_{\mu}$ | $\frac{\sum_{i=1}^N tp_i}{\sum_{i=1}^N (tp_i + fp_i)}$ | Durchschnittliche Klassenübereinstimmung der annotierten sowie der vorhergesagten Daten, gemittelt über jede zugrundeliegende Klasse. |
| Precision M | $\frac{\sum_{i=1}^N \frac{tp_i}{tp_i + fp_i}}{N}$ | Durchschnittliche Klassenübereinstimmung der annotierten sowie der vorhergesagten Daten. |
| Recall $_{\mu}$ | $\frac{\sum_{i=1}^N tp_i}{\sum_{i=1}^N (tp_i + fn_i)}$ | Durchschnittliche Effektivität eine Klasse richtig zu klassifizieren, gemittelt über jede zugrundeliegende Klasse. |
| Recall M | $\frac{\sum_{i=1}^N \frac{tp_i}{tp_i + fn_i}}{N}$ | Durchschnittliche Effektivität eine Klasse richtig zu klassifizieren. |

tp: True positives; *tn:* True negatives; *fp:* False positives; *fn:* False negatives; *N:* Anzahl gelabelter Daten

Basierend auf einer nicht binären Konfusionsmatrix lassen sich Accuracy, Precision und Recall 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 unterscheiden werden. Erstgenanntes eruiert den Durchschnitt über die Ergebnisse bzgl. aller Klassen. Das zweitgenannte Verfahren errechnet erst einen Durchschnittswert pro zugrundeliegender Klasse 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 sich Precision und Recall am besten zur Evaluierung textbasierter Klassifikatoren eignen. Vor dem Hintergrund des aktuellen Forschungsvorhabens ist Precision als die Kennzahl zu interpretieren, welche Aufschluss darüber gibt, wie viele der einer spezifischen Verfahrensklasse zugeordneten Dokumente tatsächlich dieser zugehörig sind. Wohingegen Recall den Anteil der tatsächlich richtig klassifizierten Dokumente an der Menge aller Dokumente bemisst. 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 nach folgender Berechnung zu erlangen.

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (36)$$

Nichtsdestotrotz wird zur Vollständigkeit in der folgenden Ergebnisauswertung Precision und Recall separat aufgelistet. Der F1-Score stellt zwar eine unikale Kennzahl zur Evaluierung des Systems dar, ist aber durch die Fusion zweier Kennzahlen schwer zu interpretieren. Zur Bewertung von multinomialen Klassifikationsproblemen bietet sich neben dem F1-Score daher die Cohen's-Kappa-Statistik an. Dabei gilt die Metrik als ein Maß für die Intrarater-Reliabilität, welche im Rahmen des ML die Verbesserung der Performance eines multinomialen Klassifikators gegenüber purem Raten quantifiziert (vgl. Vieira, Kaymak & Sousa, 2010, S. 924ff.). Berechnet wird die Metrik wie folgt:

$$k = \frac{\text{Pr}(a) - \text{Pr}(e)}{1 - \text{Pr}(e)} \quad (37)$$

$\text{Pr}(a)$ gibt dabei die relative Übereinstimmung von Klassifikator und Rater und $\text{Pr}(e)$ die zufällig erwartete Übereinstimmung derer an. Ein höherer Wert dieser Metrik impliziert hierbei eine bessere Performance. Dabei gilt, je höher der Wert, desto besser ist der

Klassifikator einzuschätzen, wobei Werte unter 0.4 als nicht annehmbar und somit als purer Zufall zu charakterisierend sind (vgl. Greve & Wentura, 1997, S. 111). Darüber hinaus sind Werte zwischen 0.41-0.6 als durchschnittlich und Werte zwischen 0.61-0.8 als gut zu definieren. Ein nahezu perfekter Klassifikator besitzt schließlich Werte zwischen 0.81-1.00 (vgl. McHugh, 2012, S. 279).

7.3.2 Ergebnisse der Evaluation von Klassifikationsmethoden²

Abschließend wird ein Überblick zu den trainierten Klassifikatoren und ihren Evaluationsergebnissen gegeben. Dabei werden alle Klassifikationsmodelle jeweils auf den in Kapitel 4.1 beschriebenen Modellen der distributionellen Semantik trainiert sowie evaluiert. In Bezug auf den USE wird das Modell auf dem Abstract-Datensatz neu angelernt und somit der Aspekt des Transfer Learnings mit in die Evaluation aufgenommen. Alle nachfolgenden Evaluationsmetriken basieren dabei auf den in Kapitel 5.4.2 charakterisierten, manuell erstellten Validierungsdaten. Anzumerken ist zudem die Zweiteilung der folgenden Evaluation. Zum einen werden die Modelle auf den gefilterten Daten und zum anderen auf den augmentierten Daten trainiert. Die in Kapitel 5.4.1 tangierte Klassenimbalance wird umgangen, indem jeweils gleich viele Instanzen aus den entsprechenden Datensets entnommen werden. Hierbei werden die Instanzen zufällig aus dem zugrundeliegenden Datenset gewählt. Letztendlich reduzieren sich die deskribierten Datentöpfe somit auf 7.329 Instanzen der gefilterten Daten und 21.987 der augmentierten Daten. Weiterhin werden für jede Modellklasse verschiedene Parameterkombinationen mithilfe des Grid-Search-Ansatzes getestet. Aus Gründen der Übersichtlichkeit sind diese lediglich im elektronischen Anhang dargestellt und werden an dieser Stelle nicht weiter beschrieben. Insgesamt werden hierbei 14.876 differente Modellkombinationen trainiert, welche sich auf die verschiedenen Modellklassen verteilen. An dieser Stelle ist zu betonen, dass prinzipiell weitere Parameterkombinationen in die Analyse einbezogen werden können und dies sogar zu empfehlen ist. Aufgrund des Umfangs der vorliegenden Arbeit findet allerdings eine Restriktion auf die verwendeten Parameter statt.

Die nachfolgende Tabelle 16 zeigt die verschiedenen Evaluationsmetriken der Modelle auf, welche sich aus dem gefilterten Datensatz ergeben.

Tabelle 16: Evaluationsergebnisse auf gefilterten Daten

| Model | FastText | | DAN | | USE | |
|-------------------------|----------------|--------------|----------------|---------------|----------------|--------------|
| | F ₁ | k | F ₁ | k | F ₁ | k |
| SVM | 0.733 | 0.600 | 0.850* | 0.775* | 0.667 | 0.500 |
| KNN | 0.783 | 0.675 | 0.817 | 0.725 | 0.600 | 0.400 |
| Topic-KNN | 0.617 | 0.425 | 0.717 | 0.575 | 0.500 | 0.250 |
| LSTM (N-1) | 0.767 | 0.650 | 0.733 | 0.600 | 0.817 | 0.725 |
| GRU (N-1) | 0.833 | 0.750 | 0.733 | 0.600 | 0.817 | 0.725 |
| LSTM (1-1) | 0.716 | 0.575 | 0.800 | 0.700 | 0.700 | 0.550 |
| GRU (1-1) | 0.783 | 0.675 | 0.817 | 0.725 | 0.700 | 0.550 |
| $\sim \frac{1}{N} \sum$ | 0.747 | 0.620 | 0.781 | 0.670 | 0.686 | 0.529 |

F₁: F1-Score; k: Cohen's-Kappa; *: Verwendet in Ensemble; N-1: Many-to-One; 1-1: One-to-One

Dabei ist zu erkennen, dass neben der SVM auch GRUs mit einer Many-to-One-Architektur die besten Ergebnisse hinsichtlich des F1-Scores und der Cohen's-Kappa-Metrik liefern¹⁷. Die schlechtesten Modelle hingegen bilden, die auf dem USE trainierten SVM-, KNN- sowie Topic-KNN-Modelle, mit einer Cohen's-Kappa-Metrik zwischen 25% und 50%. Zudem wird der vermutete, positive Effekt der Verwendung von DANs zur Berechnung von Paragraphvektoren bestätigt. Dies zeigt sich darin, dass DANs bei Modellen, die entsprechend einen direkten Paragraphvektor als Input verwenden¹⁸, mit einer durchschnittlichen Cohen's-Kappa-Metrik von 70% bessere Resultate gegenüber der FastText-Architektur mit einem Durchschnittswert von 59% hervorbringen. Dennoch erweisen sich FastText-Modelle, unter Verwendung eines klassischen Mittelwertsverfahrens als Kompositionsfunktion, als zielführendes Grundvorgehen. Konträr zeigen neu an-gelernte USE-Modelle tendenziell schlechte Ergebnisse und eignen sich demnach nicht zur Generierung domänenpezifischer Embeddings. Zwar erzielen derartige Modelle

¹⁷ Im elektronischen Anhang der Arbeit finden sich zusätzlich die Micro- sowie Macro-gemittelten Kennzahlen zu den Metriken: Recall, Precision und F1-Score.

¹⁸ Diese Modelle umfassen die in der Tabelle dargestellten Modelle exklusive der RNN-basierten Many-to-One-Architekturen.

State-of-the-Art-Ergebnisse in vielen NLP-Benchmarks, allerdings sind die für das Transfer Learning angedachten Modelle nicht auf domänenspezifischen Daten trainiert, weshalb sich ein Vergleich diffizil gestaltet. CHEN ET AL. (2018) berichten ähnliche Ergebnisse auf domänenspezifischem Vokabular und unterstreichen, dass ein Transfer Learning fundiertes Vorgehen mittels USE zu unzulänglichen Ergebnissen führt. In Anbetracht geringer themenspezifischer Diskrepanzen bleibt anzumerken, dass die im Rahmen der Arbeit trainierten DAN-Modelle für fortführende Forschungsvorhaben als Transfer-Learning-Modell verwendet bzw. evaluiert werden können¹⁹. Darüber hinaus zeigt sich hinsichtlich der Many-to-One-Architekturen eine gute Performance der FastText-Modelle. Dies resultiert aus dem Sachverhalt, dass derartige Modelle prinzipiell einzelne Wörter sequentiell verarbeiten und daher keine Paragraphvektoren verwenden. Weiterhin begründen sich die vergleichsweisen schlechten Ergebnisse des DAN-Modells hinsichtlich dieser Architekturen dadurch, dass die DAN-Modelle ausschließlich auf Paragraphen trainiert sind und nicht auf einzelnen Wörtern oder deren N-Grammen. Im Kontrast dazu steht der von CER ET AL. (2018) propagierte USE, welcher neben einzelnen Paragraphen auch auf Wörtern sowie N-Grammen trainiert ist und somit die Inferenz von qualitativen Wort-Embeddings ermöglicht und somit die Ergebnisse in obiger Tabelle erklärt.

In Tabelle 17 werden, analog zu den gefilterten Daten, die Ergebnisse auf Basis der augmentierten Daten dargestellt. Grundsätzlich ist diesbezüglich ein Anstieg in der Güte der Deep-Learning-Verfahren in Form der Many-to-One- wie auch One-to-One-Architektur zu verzeichnen. Insgesamt erreichen diese Modelle durch die größere Datenbasis bessere Ergebnisse als klassische Methoden des Machine Learning.

¹⁹ Die dazu nötigen Tensorflow-Checkpoint-Files sind im elektronischen Anhang der Arbeit zu finden.

Tabelle 17: 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 | 0.475 | 0.650 |
| KNN | 0.783 | 0.675 | 0.800 | 0.700 | 0.567 | 0.350 |
| Topic-KNN | 0.650 | 0.475 | 0.700 | 0.550 | 0.550 | 0.325 |
| LSTM (N-1) | 0.850* | 0.775* | 0.833 | 0.750 | 0.783 | 0.675 |
| GRU (N-1) | 0.833 | 0.750 | 0.800 | 0.700 | 0.783 | 0.675 |
| LSTM (1-1) | 0.750 | 0.625 | 0.833 | 0.750 | 0.767 | 0.650 |
| GRU (1-1) | 0.733 | 0.600 | 0.833* | 0.750* | 0.767 | 0.650 |
| $\sim \frac{1}{N} \sum$ | 0.76 | 0.65 | 0.80 | 0.71 | 0.67 | 0.57 |

F₁: F1-Score; k: Cohen's-Kappa; *: Verwendet in Ensemble; N-1: Many-to-One; 1-1: One-to-One

Parallel fällt auf, dass die ersten drei Verfahren in obiger Tabelle, ergo die verwendeten ML-Verfahren, auf den augmentierten Daten schlechtere Ergebnisse erzielen als auf den gefilterten. Im Vergleich bildet das beste Modell auf der augmentierten Datenbasis ein LSTM (N-1) mit einer Cohen's-Kappa-Metrik von 77.5%. Analog zu den nicht-augmentierten Daten ist das schlechteste Modell in den auf USE-Vektoren trainierten Modellen mit einer Cohen's-Kappa-Metrik von 32.5% zu verorten. Im Allgemeinen stechen die klassischen Machine-Learning-Verfahren mit einer vergleichsweise niedrigen Performance unter der Verwendung des USE hervor. Als eine mögliche Ursache hierfür ist auf die fünfmal höhere Dimensionalität der USE-Vektoren zu verweisen.

Nichtsdestotrotz ist zu konstatieren, dass die Ergebnisse aller untersuchten Algorithmen auf beiden Datensets nah beieinanderliegen. Auffallend ist jedoch, dass sich die Performance der Deep-Learning-Modelle mit zunehmender Datenmenge verbessert und die der klassischen Machine-Learning-Methoden verschlechtert. Folglich kann keine Präferenz bei der Auswahl des Datensets formuliert werden.

Wie in Kapitel 6.3.3 bereits dargelegt, kann darüber hinaus auch das Topic-Modeling-Modell LDA zur Klassifikation von unbekannten Problemstellungen herangezogen werden. Hierbei besteht einerseits die Option einer direkten Zuordnung durch ein LDA-Mo-

dell mit Topics, die repräsentativ für die aufgestellten Zielklassen sind. Andererseits existiert die Möglichkeit einen weiteren Klassifikationsalgorithmus zu trainieren, durch welchen eine Zuordnung auf Basis der Topic-Verteilung in einem Dokument stattfindet, wobei in diesem Fall keine Restriktion hinsichtlich der Topic-Anzahl im zugrundeliegenden LDA-Modell vorliegt. Die entsprechenden Evaluationsergebnisse sind in nachfolgender Tabelle dargelegt.

Tabelle 18: Ergebnisse der Klassifikationen mit LDA-Modellen

| Modell | F₁ | k |
|------------------------------------|----------------------|---------------|
| <i>Direkte Zuweisung durch LDA</i> | 0.700* | 0.550* |
| <i>LDA (n=7) mit SVM</i> | 0.450 | 0.188 |

F₁: F1-Score; *k*: Cohen's Kappa; *: Verwendet in Ensemble

Im ersten Fall zeigt das unter Kapitel 7.2.2 mittels Grid-Search identifizierte und evaluierte Modell mit drei Topics eine ähnliche Güte wie die Zuweisung auf Basis der Ähnlichkeit zwischen Problembeschreibung und Topic-Embedding (Topic-KNN). Hinsichtlich des zweiten Vorgehens wird das trainierte LDA-Modell mit insgesamt sieben Topics herangezogen, um dessen Vorhersagen bzgl. des gefilterten Datensets als Input für einen Klassifikationsalgorithmus zu verwenden. Zur Vergleichbarkeit werden hierbei verschiedene SVMs trainiert, wobei das beste Modell einen F1-Score von 45% und ein Cohen's Kappa von 18.8% erreicht.

Abschließend werden für eine finale Zuordnung die differenten Modelle im Rahmen des EL vereint. Somit ist die Bewertung dieser Modelle als finale Systemevaluation zu betrachten. Aufbauend auf den dargelegten Evaluationsergebnissen erfolgt die Zusammensetzung anhand der besten Modelle der oben dargestellten Evaluationstabellen und umfasst demnach die dort mit einem Stern gekennzeichneten Modelle. In der nachfolgenden Tabelle werden somit die drei besten Modellkombinationen des Ensemble Learning aufgelistet, wobei sich an dem Vorgehen in Kapitel 6.3.3 orientiert wird²⁰.

²⁰ Die vollständige Tabelle ist im elektronischen Anhang der Arbeit dargelegt.

Tabelle 19: Ergebnisse des Ensemble Learning

| Modelle | F₁ | k |
|------------------------------|----------------------|--------------|
| SVM + LSTM (N-1) + GRU (1-1) | 0.900 | 0.850 |
| LDA + SVM + GRU (1-1) | 0.883 | 0.825 |
| LDA + LSTM (N-1) + GRU (1-1) | 0.883 | 0.825 |

F₁: F1-Score; *k*: Cohen's Kappa

Wie zu erkennen, kann durch die Verwendung des Ensemble Learning eine Verbesserung bezüglich der Cohen's-Kappa-Metrik um etwa 9% erreicht werden. Dabei erzielt die Kombination einer SVM, eines LSTM (N-1) und eines GRU (1-1) das beste Ergebnis, um die Zielstellung der vorliegenden Arbeit zu erlangen.

Summierend ist zu konstatieren, dass sich für die Zuordnung der Verfahrensklasse mit einer hohen Qualität die Nutzung einer Ensemble Learning basierenden Methodik empfiehlt. Hinzukommt, dass bei One-to-One-Architekturen die Verwendung von DAN-Embeddings bessere Ergebnisse erzielt als andere Modelle der distribuierten Semantik. Im Vergleich dazu erreichen FastText-Modelle bessere Resultate in Bezug auf Many-to-One-Architekturen.

8 Ergebnisse¹

Wie in Kapitel 1.3.2 beschrieben besteht das Resultat eines Design-Science-Projektes aus einem nutzbaren Artefakt sowie gefundenen Design-Prinzipien, welche in weiterer Forschung und Entwicklung ihre 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 Schlüsselwortinformationen, 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 eine webbasierte Software überführt. Abbildung 49 zeigt die finale Startseits des entwickelten Systems.

| | | |
|--------------|----------------|---------------|
| INPUT | PROCESS | RESULT |
|--------------|----------------|---------------|

Please Enter Your Problem Description

Title
Evaluation of machine errors

Short description
I have different machine types and want to find similar error structures based on recent error history. Thereby the errors should be grouped in different categories.

Long description

SUBMIT ➤

Abbildung 49: 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 38). Weiterhin zeigen die Ergebnisse der Modellevaluierung, dass das Ziel eine bessere Einschätzung als das manuelle Raten hervorzubringen, mit einem Cohen's-Kappa-Wert des Ensemble-Modells von 85% 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 zwölf Design Prinzipien abzuleiten, welche in nachfolgender Tabelle zusammengefasst sind.

Tabelle 20: 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. |
| 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 <i>semantischen</i> 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, dabei sollten die Trainingsdaten eine hohe Qualität aufweisen und möglichst nur die Zielklassen repräsentieren. |
| P10 | Klassifikation von kurzen Paragraphen verspricht bessere Ergebnisse als von längeren. |
| P11 | Ensemble Learning verspricht ebenfalls bei differenten Herangehensweisen Vorteile und sollte gerade bei geringer Datenbasis verwendet werden. |
| P12 | Interfaces und definierte, komplexe Datentypen sorgen für die Erweiterbarkeit des Systems. |

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 7.3.2 dargestellt erreichen potentielle Klassifikationsalgorithmen durch spezifisch trainierte Embeddings höhere Gütwerte und stärkere Generalisierungskraft im Rahmen der abgezielten Nutzung. Weiterhin ist zu konstatieren, dass bei geringen Datenmengen, aufgrund der unter Kapitel 4.1.1 aufgezeigten Vorteile, auf die FastText-Architektur 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 21: Minima und Maxima der Bewertung von Embedding-Modellen

| Modell (Basis) | SVM (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 | 0.19 | 0.37 | 0.13 | 0.33 |
| <i>DAN (fasttext_31)</i> | 0.51 | 0.8 | 0.94 | 0.98 | 0.17 | 0.37 | 0.06 | 0.30 |
| <i>DAN (fasttext_1)</i> | 0.46 | 0.7 | 0.94 | 0.99 | 0.19 | 0.35 | 0.05 | 0.28 |

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

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 wird die Relevanz der detaillierten Untersuchung des Vektorraums dadurch verdeutlicht, 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 beispielsweise festgestellt werden, dass eine Zielklasse zwei stark konzentrierte Areale im Vektorraum aufweist und jene demnach weiter unterteilt werden sollte. Zeitgleich erlaubt es diese Untersuchung potentielle 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 zu konstatieren, dass die Kombination differenter Keyword-Extraction-Methoden empfehlenswert ist (P7). Wie in Kapitel 7.2.2 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 Vektorenrichtung 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 7.3.2 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. Weiterhin zeigt sich, dass das Areal des Ensemble Learning ebenfalls Vorteile unter der Anwendung stark differenzierender Herangehensweisen aufweist. Aufgrund dessen wird empfohlen, vor allem vor dem Hintergrund einer geringen Datenbasis, verschiedene Vorgehen durch EL zu kombinieren anstatt eine Methode bzw. ein Modell zu fokussieren.

Additional ist festzuhalten, dass die Verarbeitung und Zuordnung von kürzeren Paragraphen bessere Ergebnisse verspricht als die von längeren Dokumenten und Texten (P10). Dies resultiert aus differenten Sachverhalten. So bestehen auf der einen Seite potentiell

Probleme hinsichtlich der Verarbeitung von langen Sequenzen durch RNN-basierte Modelle (vgl. Kapitel 4.2). Auf der anderen Seite besteht die Gefahr schwammiger Paragraph-Embeddings, wenn diese aus einer großen Menge einzelner Wort-Embeddings generiert werden. Resultierend wird empfohlen, sofern möglich, eine Einschränkung auf kürzere Dokumente vorzunehmen.

Abschließend folgt das letzte Prinzip aus der durchgeführten Implementierung. Ausgehend von dieser wird empfohlen komplexe Datentypen sowie Interfaces zu definieren und implementieren, die eine leichte Erweiterung des Systems gewährleisten. Zeitgleich sichern die komplexen Datentypen, wie bspw. *Prediction* und *Keyword*, die fehlerfreie Verarbeitung von Ausgaben und die vereinfachte Kommunikation zwischen Klassen durch die einhergehende Standardisierung von Formaten.

9 Kritische Würdigung¹

Das grundlegende Forschungsziel der Arbeit ist die Entwicklung eines Recommender-Systems zur Selektion von Machine-Learning- bzw. Data-Mining-Verfahrensklassen, um eine Unterstützung für fachfremde Personen sowie Data Scientists zu ermöglichen. Im ersten Teil der Arbeit wird hierzu die initiale Datenbeschaffung, -selektion und -augmentation erörtert, um darauffolgend adäquate Methoden zur Erfüllung der Zielstellung aufzuzeigen. Weiterführend wird im zweiten Teil ein Vorgehen zur Implementierung eines Prototyps sowie dessen Softwarearchitektur expliziert. Abschließend werden alle Methoden zur finalen Modellselektion evaluiert sowie ein Überblick zu den Gesamtergebnissen und den identifizierten Design-Prinzipien gegeben. Aufbauend hierauf erfolgt im folgenden Kapitel eine kritische Würdigung des Vorhabens.

Zunächst ist zu erwähnen, dass durch den in Kapitel 5 beschriebenen Datenbeschaffungsprozess eine hohe Anzahl an Daten erlangt wird, diese jedoch lediglich zu Trainings- und Testzwecken einsetzbar sind. Dies begründet sich aus der Beschaffenheit dieser Daten. Zwar beinhalten diese semantisch relevante Konstrukte der zu klassifizierenden Verfahrensklassen, allerdings liegen sie nicht in Form von Problembeschreibungen vor, sondern in Form von wissenschaftlichen Artikeln. Als Resultat sind keine Validierungsdaten aus den initialen Daten zu extrahieren, weshalb diese manuell zusammengetragen werden. Diese manuelle Erstellung der Daten hat allerdings eine subjektive Auswahl dieser zur Folge, was unter dem Gesichtspunkt der Evaluationsergebnisse als kritisch zu erachten ist. Auch liegen diese Daten in Form kurzer Problembeschreibungen vor, weshalb keine Evaluation langer Problembeschreibungen vorgenommen wird. Additional besteht hierdurch eine starke, mengenseitige Diskrepanz zwischen den Datensets, welche ebenfalls zu weniger belastbaren Aussagen hinsichtlich der Generalisierungsfähigkeit der trainierten Modelle führt. Summierend kann durch den Erstellungsprozess der Validierungsdaten sowie deren Umfang konstatiert werden, dass potentielle Evaluationsergebnisse zukünftiger Arbeiten von den in dieser Untersuchung dargelegten abweichen können, sowie dass eine weiterführende Evaluation der Methodik auf Basis einer größeren Datenmenge durchgeführt werden sollte. Im selbigen Kon-

text ist anzumerken, dass die angewendeten Verfahren der Datenaugmentation den frühen Entwicklungen des Forschungsareals zuzuordnen sind. Schlussfolgernd können vermeintlich bessere Ergebnisse und eine höhere Qualität der Trainingsdaten erreicht werden indem komplexere und fortgeschrittenere Methoden, wie beispielweise die angeprochenen Variational-Autoencoder-Modelle, mithilfe einer größeren Datenbasis appliziert werden.

Additional wird sich im Rahmen der vorliegenden Arbeit beim Training von Embedding-Modellen auf die Architekturen DAN, FastText sowie USE beschränkt, wobei die ersten beiden mit dem Ziel 100-dimensionaler Vektoren trainiert werden²¹. Die Wahl der Dimensionalität erfolgt hierbei auf Basis der in der Literatur verwendeten FastText-Modelle. Im Zuge weiterführender Forschungsvorhaben ist allerdings eine Untersuchung additionaler Dimensionalitäten empfehlenswert, da sich dies eventuell positiv auf die Inklusion der Semantik auswirken kann. Gleichzeitig besteht indes die Gefahr einer zu großen Diffusität der resultierenden Vektorrepräsentationen und somit des Vektorraums. Außerdem ist die Untersuchung weiterer Architekturen vor dem Hintergrund der Vielzahl an möglichen Verfahren der distributionellen Semantik als sinnvoll zu erachten. Im selben Kontext werden in der Arbeit Parameterkorrelationen der Verfahren untersucht, um teilweise richtungsweisende Informationen zur Anpassung der Hyperparameter zu erhalten. Generell ist allerdings zu konstatieren, dass diese Korrelationen auf teilweise geringen Mengen an trainierten Modellen und Kombinationen fußen, wodurch deren Belastbarkeit und Aussagekraft beschränkt ist. Zudem wird häufig das Verfahren der PCA angewendet, um die Dimensionalität der Embeddings zu reduzieren. Die Wahl dieses Verfahrens erfolgt hierbei aufgrund seines parameterfreien, deterministischen Charakters, der resultierenden Vergleichbarkeit der Ergebnisse sowie der geringen Rechenintensität im Vergleich zu anderen Verfahren. Nichtsdestotrotz besteht die Option in fortführenden Forschungsvorhaben andere Reduktionsalgorithmen, wie beispielsweise T-SNE²², zu verwenden (vgl. Van Der Maaten, Postma & Van den Herik, 2009, S. 66ff.).

²¹ Aufgrund der Verwendung des USE im Rahmen des Transfer Learning wird dieses mit den vortrainierten 512-dimensionalen Vektoren verwendet.

²² T-SNE steht hierbei für T-distributed stochastic neighbor embedding.

Ferner werden im Kontext der Evaluation von Topic-Modellen externe Kennzahlen deskribiert, welche ebenfalls implementiert, aber letztlich nicht verwendet werden. So verspricht die *Related-Article-Concept-Overlap-Kennzahl* prinzipiell eine gute Eignung für die Evaluation der resultierenden Topics, da diese Wikipedia und die enthaltenen manuell erstellen, semantischen Zusammenhänge als Quelle nutzt. Allerdings geht die Abfrage der unterschiedlichen Verlinkungen mit einem sehr hohen Zeitaufwand einher, weshalb in diesem Punkt auf weitere Forschungen zu verweisen ist.

In Anbetracht der eigentlichen Klassifikation hat sich die Kombination differenter Klassifikatoren im Rahmen des Ensemble Learning als die beste Methodik herausgestellt. Je doch ist anzumerken, dass in der Literatur, neben den deskribierten, weitere Methoden des Areals existieren. Zudem ist das im Rahmen der Arbeit verwendete Weighted-Averaging-Verfahren als vergleichsweise simpel gegenüber den Methoden Bagging und Boosting anzusehen, welche i.d.R. State-of-the-Art-Performance in Hinblick auf Klassifikationen versprechen (vgl. Sagi & Rokach, 2018, S. 1249). Im selben Kontext zeigt sich eine bessere Güte der Vorhersage bei der Verwendung kurzer Problembeschreibungen bzw. Paragraphen. Dies resultiert vermeintlich aus der Verwendung der ausgewählten Embedding-Architekturen, da diese durch eine hohe Anzahl an Wörtern und somit einer größeren Menge an Rauschen und Informationen potentiell schwammigere Ergebnisse hervorbringen, welche anschließend durch Classifier eingeordnet werden. Zur Begegnung dieser Thematik wäre beispielsweise eine satzbasierte Evaluation längerer Paragraphen möglich, welche in weiterführenden Arbeiten untersucht werden kann.

Neben der Zuordnung einer übergebenen, natürlichsprachlichen Problemstellung zu einer Verfahrensklasse wird mit der vorliegenden Arbeit ein regelbasierter Ansatz der Named-Entity-Recognition illustriert, um Analyseentitäten aus dem Text zu extrahieren. Ähnlich wie bei dem eben beschriebenen Sachverhalt, ist das Gebiet der NER als komplex und umfassend anzusehen. Beispielsweise listen GOYAL ET AL. (2018) insgesamt 48 verschiedene lernbasierte, regelbasierte sowie hybride NER-Ansätze auf, womit die Komplexität dieses Forschungsfeldes betont wird. Somit ist in sich anschließenden Forschungsvorhaben, bestenfalls unter der Zuhilfenahme von realen Problemstellungen, ein komplexeres Modell mit höherer Güte zu entwickeln. Das hier deskribierte, regelbasierte Vorgehen kann dabei als Grundlage für ein hybrides System dienen.

10 Fazit und Ausblick²

Die Data Science ist ein wachsendes Forschungsareal, welches zunehmend an Relevanz gewinnt. Die fachliche Interdisziplinarität sowie das erforderliche Domänenwissen, um die zugrundeliegende Problemstellung sowie die Daten fachgerecht zu bewerten, unterstreichen den Bedarf die Lücke zwischen Domänenexperten und Data Scientists zu schließen. An dieser Stelle setzt der vorliegende Forschungsbeitrag an und präsentiert ein Vorgehen sowie ein implementiertes Recommender-System zur Überbrückung dieser Lücke. Hierbei ist zunächst aufgezeigt, wie eine Datenbasis erstellt werden kann, die weiterhin zur Umsetzung der Zielstellung dient. Des Weiteren demonstriert der Prozess der Datenbeschaffung ein Vorgehen zum Entfernen semantischer Ausreißer mit Hilfe von Modellen der distributionellen Semantik und differente Methoden zur Daten-augmentation auf. Aufbauend hierauf werden verschiedene Methoden zur Umsetzung der zentralen Zielstellung des Forschungsbeitrages expliziert und evaluiert. Hierbei werden verschiedene Möglichkeiten zur Evaluation von Embedding-Modellen, Modellen des Topic Modeling sowie Klassifikationsmodellen deskribiert. Im Zuge dessen erfolgt auf Basis der illustrierten Methoden eine Evaluation des Gesamtsystems.

Im Kontext der Implementation des Prototyps als webbasierte Applikation wird neben der reinen Empfehlung einer Verfahrensklasse eine exemplarische Beispielanalyse durch die Verwendung des Named-Entity-Recognition-Verfahrens integriert. Dies dient einerseits dazu, Data Scientists einen ersten Einblick in die domänenspezifische Problemstellung zu gewähren, andererseits kann der Nutzer auf diese Weise die eruierte Verfahrensklasse mit seinen Vorstellungen abgleichen. Zur Reflexion werden im nachfolgenden Abschnitt die zentralen Erkenntnisse vor dem Hintergrund der aufgestellten Forschungsfragen zusammengefasst.

Forschungsfrage I:

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

Im Rahmen der vorliegenden Forschungsarbeit werden wissenschaftliche Datenbanken zur **Extraktion** von Trainingsdaten verwendet. Hierbei werden der Elsevier-Datenbankverbund, die arXiv-Datenbank sowie das Online-Journal Medium mit klar definierten und voneinander abgegrenzten Suchanfragen durchsucht. Daraus ergeben sich insgesamt ca. 90.000 initiale Textdokumente, welche nachfolgend von Rauschen bereinigt werden. Zunächst wird hierzu eine regelbasierte Reinigung der Textdaten mittels regulärer Ausdrücke durchgeführt, um *syntaktisch* irrelevante Textbausteine sowie Dokumente, welche ein bestimmtes Maß an Rauschen überschreiten, zu entfernen. Basierend auf der distributionellen Hypothese werden anschließend Ausreißer mithilfe eines dichtebasierten Noise-Clusteringverfahrens entfernt, um *semantisch* irrelevante Textdaten auszuschließen. Generell wird dieser Prozess sowohl auf allen geladenen Publikationen sowie additional auf nur den Abstracts dieser durchgeführt. Anschließend werden über ein Ähnlichkeitsbasiertes Vorgehen definitionsähnliche Sätze aus den gesamten Daten extrahiert, welche zur weiteren Expansion genutzt werden.

Diese **Expansion** strebt eine höhere Variabilität und eine Erweiterung der zugrundeliegenden Daten an. Aufgrund der Ambiguität natürlicher Sprache werden dazu spezielle Methoden der Datenaugmentation im Bereich des NLP aufgezeigt. Hierbei bieten sich die Möglichkeiten einer Substitution oder Reihenfolgenanpassung von Wörtern und Paragraphen mithilfe eines differenten Vorgehens oder auch das automatisierte Generieren neuer Textdaten an. Im vorliegenden Kontext werden hierbei zunächst differente Kriterien definiert, welche über die Substitution eines Wortes in den Daten entscheiden. Weiterhin wird ein Markov-Modell trainiert, das auf Grundlage der konditionalen Wahrscheinlichkeiten von Wortfolgen im zugrundeliegenden Datensatz neuartige Sätze generiert. Als Resultat dieser Schritte kann die gefilterte Datenbasis in der vorliegenden Untersuchung verdreifacht werden, sodass final 7.329 Instanzen im gefilterten und 21.987 im augmentierten Datensatz zur Verfügung stehen. Additional besteht der oben angesprochene Abstract-Datensatz aus insgesamt 50.988 Texten.

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 werden insgesamt zwölf Anforderungen an den zu erstellenden Prototyp aufgelistet. Dabei sind diese in funktionale und nicht-funktionale Anforderungen unterteilt und mittels der Kategorien essentiell, konditional und optional priorisiert. Vor dem Hintergrund des deskribierten Gestaltungsziels der Forschungsarbeit ist die wichtigste Anforderung die Zuordnung der eingegebenen Problembeschreibung zu einer der drei analysierten Verfahrensklassen, da diese die Kernfunktionalität des Systems beschreibt. Hierbei ist zu konstatieren, dass die Zuweisung durch die linguistische Herausforderung der syntaktischen Divergenz erschwert wird. Daher wird das Entgegenwirken dieser als Anforderung an den Prototyp gestellt. Eine weitere daraus resultierende Anforderung ist, dass das System in seiner Zuordnung zuverlässiger als bei reinem Raten agiert, um eine adäquate Anwendung der Software zu garantieren. Zur Standardisierung und Vergleichbarkeit wird die Cohen's-Kappa-Metrik als unikale Kennzahl verwendet. Additional wird intendiert dem Nutzer durch die automatisierte Extraktion von Analyseentitäten eine personalisierte, exemplarische Analyse auszugeben. Hinsichtlich der nicht-funktionalen Anforderungen erweist sich die Erweiterbarkeit des Systems als wichtigste Komponente, da diese das Einbinden neuer Modelle und Verfahren in zukünftigen Vorhaben sicherstellt.

In Anbetracht des zweiten Teils der Forschungsfrage empfiehlt sich ein agiles Vorgehen, um eine anforderungsgerechte Prototypentwicklung zu ermöglichen. Additional zu dem zugrundeliegenden Design-Science-Research-Prozess ist die Verwendung des Prototyping-Modells nach PRESSMAN & MAXIM (2015) zu präferieren. Durch die Rückkopplungen im Prozess können die Anforderungen über den Softwareentwicklungsprozess hinweg iterativ evaluiert und novelliert werden. Ergänzend zeigt sich das Model-View-Controller-Pattern im Rahmen der Entwicklung als hilfreich, da das Frontend des Systems von dessen Backend getrennt werden kann. Zudem ist objektorientierte Programmierung als zielführend anzusehen, um die angesprochene Erweiterbarkeit mit abstrakten Klassen und Interfaces zu gewährleisten und unterschiedliche Aspekte des Systems durch komplexe Datentypen zu standardisieren. Abschließend ist auf das implementierte

Facade-Pattern zu verweisen, wodurch die zentrale Verwaltung und Steuerung des Systems und somit dessen Nutzbarkeit in weiteren Forschungsvorhaben erreicht wird.

Forschungsfrage III:

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

Mit Fokus auf die Entwicklung des Prototyps werden unterschiedliche Methodengruppen sowie Algorithmen identifiziert und appliziert. Die Evaluation dieser sowie die des Gesamtsystems teilt sich daher in die verschiedenen Kategorien und ist an Methoden aus der jeweiligen Fachliteratur orientiert. Hinsichtlich der distribuierten Repräsentationen wird zwischen einer extrinsischen und intrinsischen Bewertung unterschieden. Im Zuge der ersten Variante 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 anhand ähnlichkeitsbasierter Analysen und der Reinheit der im Vektorraum geclusterten Klassen untersucht. In Bezug auf die Evaluation von Topic-Modellen ist festzustellen, dass die Bewertung der Passgenauigkeit eines Topics auf ein semantisches Konstrukt im Allgemeinen als diffizil anzusehen ist. So werden für Methoden der Keyword Extraction unterschiedliche Kohärenzkennzahlen verwendet, welche die Zusammengehörigkeit der Wörter in einem Topic bemessen. Weiterhin wird die Methode des Grid-Search angewendet, um das beste LDA-Modell anhand der Kennzahl Log-Likelihood zu identifizieren. Ferner erfolgt die Evaluation von Klassifikationsmodellen sowie des Gesamtsystems, das aufgrund der Anwendung des Ensemble Learning zur Zusammenführung differenter Klassifikatoren ebenfalls durch Klassifikationsmetriken evaluiert wird. Aufgrund des Sachverhalts einer multinomialen Klassifikation wird hier das Verfahren des Micro-Averaging hinsichtlich der Kennzahlen Precision, Recall sowie F1-Score verwendet. Zur finalen Bewertung wird additio-nal die Kennzahl Cohen's-Kappa genutzt, um die Verbesserung der Performance jener Modelle gegenüber purem Raten abzubilden. Darüber hinaus werden für alle Methodengruppen Korrelationen zwischen den Modellparametern und den Testergebnissen analysiert, um diese als Kriterium zur Sinnhaftigkeit potentieller Anpassungen zu verwenden.

Ausblick

Ausgehend von den deskribierten Erkenntnissen der Arbeit lassen sich differente Ansatzzpunkte für weitere Forschungsvorhaben charakterisieren.

Im Hinblick auf die Datenbeschaffung empfiehlt es sich neben den beiden abgefragten Datenbankverbunden weitere wissenschaftliche Datenbanken mit einzubeziehen. Hinzukommt, dass neben den applizierten Methoden der Datenaugmentierung weitere Methodiken untersucht werden sollten, um qualitativ hochwertigere sowie größere Datensets zu generieren. Zum einen bietet dies den distributionellen, semantischen Modellen mehr lexikalischen Kontext, um die entsprechenden Embeddings zu erlernen. Zum anderen haben die durchgeföhrten Untersuchungen aufgezeigt, dass die berechneten Deep-Learning-Modelle bessere Performance bei größeren Datenmengen erzielen. Grundsätzlich kann das deskribierte Vorgehen zur Datenbeschaffung adaptiert werden, um eine hochqualitative Datenbasis zu erzeugen.

Weiterführend lassen sich die im Rahmen der Arbeit thematisierten Methoden in die Kategorien Embeddings, Topic Modeling, Keyword Extraction sowie Machine-Learning-Klassifikatoren unterteilen. Hierbei gibt die Arbeit Aufschluss über deren Eignung. In fortführenden Forschungsbeiträgen sollten jedoch weitere Methoden zur Erfüllung der gegebenen Zielstellung analysiert werden. Ein interessanter Ansatz findet sich hier beispielsweise in der Neural Machine Translation, bei welcher die differenten Vokabulare der Personengruppen als unterschiedliche Sprachen behandelt werden können. Hierfür besteht allerdings die Notwendigkeit eines umfangreichen, multilingualen Korpus, der zum Training potentieller Encoder-Decoder-Modelle verwendet werden kann. Im Zuge dessen ist hinsichtlich der distributionellen Semantik ebenfalls die Untersuchung weiterer Methoden denkbar, die das Grundgerüst zum Erlernen der Klassifikatoren bilden. Neben der reinen Empfehlung einer Verfahrensklasse propagiert der vorliegende Forschungsbeitrag zudem ein regelbasiertes Vorgehen zur Extraktion spezifischer Datenentitäten. Dieses kann weiterhin als Grundlage für die Entwicklung eines hybriden NER-Systems verwendet werden, indem es Eigenschaften eruiert, die als Input zum Training eines weiteren Classifiers dienen können. Durch eine derartige Struktur kann additional eine Unterscheidung zwischen Analyseeigenschaft und Analyselabel vorgenommen werden, die die Erstellung exemplarischer Analysen und potentieller Datenstrukturen vereinfacht.

Zur Begegnung der Problematik von syntaktischen Divergenzen fungieren die aufgezeigten, domänenspezifisch trainierten FastText-Modelle und Deep-Averaging-Netzwerke als vielversprechendes Grundverfahren, insbesondere bei einer geringen Datenbasis wie es hier der Fall ist. Dennoch ist ein Training der DAN-Modelle auf der Ebene von Wörtern bzw. N-Grammen zu empfehlen, da auf diese Weise bessere Ergebnisse in rekurrenten Many-to-One-Architekturen und folglich eine Verbesserung der Gesamtperformance des Systems erzielt werden können.

Wenngleich sich der Schwerpunkt der vorliegenden Arbeit auf die initiale Erstellung eines Prototyps bezieht, erscheint eine Verschmelzung des entwickelten Systems mit bereits existierenden Experten- oder Meta-Learning-Systemen nützlich. Dementsprechend könnten deren Synergieeffekte für additionale Informationen genutzt werden, um dem Data Scientist nicht nur eine grundlegende Verfahrensklasse zur Problemstellung auszugeben, sondern weiterführend auch eine konkrete Methode zu deren Lösung. Denkbar ist an dieser Stelle z.B. ein zweigeteiltes System, welches zunächst den entwickelten Prototypen zur Auswahl der Verfahrensklasse anwendet und daraufhin die Methodik weiterführender Systeme. Einen Überblick zu multiplen möglichen Systemen liefern SERBAN ET AL. (2013). Vor dem Hintergrund kollaborativer Aspekte von Recommender-Systemen bietet sich das Speichern von vergangenen Anfragen an, um diese bei der Einordnung neuer Problemstellungen heranzuziehen. Hierbei kann eine Feedbackschleife zur Sicherung der Korrektheit der gespeicherten Ergebnisse implementiert werden.

Der konstruierte Prototyp sowie die konzipierte Softwarearchitektur dessen bieten ein Grundgerüst für tiefergehende Forschungsvorhaben und eröffnen hierzu gleichzeitig multiple Ansatzpunkte, die es im Weiteren zu eruieren gilt. Aufgrund der leicht zu erweiternden Architektur und der hieraus resultierenden Anschlussfähigkeit des Prototyps können weiterführende Methoden implementiert und untersucht werden.

Literaturverzeichnis

- Aggarwal, C. C. (2014). An Introduction to Frequent Pattern Mining. In C. C. Aggarwal & J. Han (Hrsg.), *Frequent Pattern Mining* (S. 1–17). Cham: 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.
- Bag, S., Ghadge, A. & Tiwari, M. K. (2019). An integrated recommender system for improved accuracy and aggregate diversity. *Computers & Industrial Engineering*, 130, 187–197.
- Bagali, S. & Waggoner, 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*, 339–350.
- Bakarov, A. (2018). A Survey of Word Embeddings Evaluation Methods. *arXiv preprint arXiv:1801.09536*.
- Balabanović, M. & Shoham, Y. (1997). Fab: Content-based, Collaborative Recommendation. *Communications of the ACM*, 40(3), 66–72.
- Barde, B. V. & Bainwad, A. M. (2017). An overview of topic modeling methods and tools. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, 745–750.
- Baroni, M., Dinu, G. & Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Long Papers)*, 1, 238–247.
- Bayer, J., Wierstra, D., Togelius, J. & Schmidhuber, J. (2009). Evolving Memory Cell Structures for Sequence Learning. In C. Alippi, M. Polycarpou, C. Panayiotou, & G. Ellinas (Hrsg.), *Artificial Neural Networks – ICANN 2009, Part II* (S. 755–764). Berlin: Springer.

- Becker, J., Holten, R., Knackstedt, R. & Niehaves, B. (2003). *Forschungsmethodische Positionierung in der Wirtschaftsinformatik: epistemologische, ontologische und linguistische Leitfragen* (Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 93). Abgerufen von <http://hdl.handle.net/10419/59562> [08.10.2018].
- Becker, J., Niehaves, B. & Knackstedt, R. (2004). Bezugsrahmen zur epistemologischen Positionierung der Referenzmodellierung. In J. Becker & P. Delfmann (Hrsg.), *Referenzmodellierung: Grundlagen, Techniken und domänenbezogene Anwendung* (S. 1–17). Heidelberg: Physica-Verlag.
- Bergstra, J. & Bengio, Y. (2012). Random Search for Hyper-parameter Optimization. *Journal of Machine Learning Research*, 13, 281–305.
- Bhukya, D. P. & Ramachandram, S. (2010). 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(1), 51–59.
- Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3, 993–1022.
- 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*, 543–551.
- 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*, 10–21.
- Brazdil, P. B., Carrier, C. G., Soares, C. & Vilalta, R. (2009). *Metalearning: Applications to Data Mining*. Berlin: Springer.
- Brazdil, P. B. & Henery, R. J. (1994). Analysis of Results. In D. Michie, D. J. Spiegelhalter

- & C.C. Taylor (Hrsg.), *Machine Learning, Neural and Statistical Classification* (S. 175–212). New York: Ellis Horwood.
- Camacho-Collados, J. & Pilehvar, M. T. (2018a). From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. *Journal of Artificial Intelligence Research*, 63, 743–788.
- Camacho-Collados, J. & Pilehvar, M. T. (2018b). 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*, 40–46.
- Campos, Ra., Canuto, S., Salles, T., de Sá, C. C. A. & Gonçalves, M. A. (2017). Stacking Bagged and Boosted Forests for Effective Automated Classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 105–114.
- Campos, Ri., 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: ECIR 2018* (S. 684–691). Cham: Springer International Publishing.
- Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., ... Kurzweil, R. (2018). Universal Sentence Encoder. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 169–174.
- 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.
- Chen, Q., Peng, Y. & Lu, Z. (2018). BioSentVec: creating sentence embeddings for biomedical texts. *arXiv preprint arXiv:1801.09536*.
- 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*, 103–111.
- Chung, J., Gülcöhre, Ç., Cho, K. & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*.

- Colah, C. (27. August 2015). Understanding LSTM Networks [Blogeintrag]. Abgerufen von <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> [20.03.2019].
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. & Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12, 2493–2537.
- Coulombe, C. (2018). Text Data Augmentation Made Simple By Leveraging NLP Cloud APIs. *arXiv preprint arXiv:1812.04718*.
- Danubianu, M. & Socaciu, T. (2011). Efficient Selection of Data Mining Method. *Analele Universității Eftimie Murgu Reșița. Fascicula de Inginerie*, 18(3), 101–108.
- Dave, R. N. (1991). Characterization and detection of noise in clustering. *Pattern Recognition Letters*, 12(11), 657–664.
- Davies, D. L. & Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2), 224–227.
- 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*, 1500–1505.
- 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), 1–32.
- 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*, 226–231.
- Faber, P. & Rodríguez, C. I. L. (2012). Terminology and specialized language. In P. Faber (Hrsg.), *A Cognitive Linguistics View of Terminology and Specialized Language* (S. 9–32). Berlin: De Gruyter Mouton.
- Fadaee, M., Bisazza, A. & Monz, C. (2017). Data Augmentation for Low-Resource Neural

- Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, 2, 567–573.
- 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*, 406–414.
- 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 (Long Papers)*, 1, 1105–1115.
- Fowler, M. (2019). *Refactoring: Improving the Design of Existing Code* (2. Aufl.). Boston: Pearson Education.
- Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (2015). *Design Patterns: Entwurfsmuster als Elemente wiederverwendbarer objektorientierter Software*. Frechen: mitp Verlag.
- Gers, F. A., Schmidhuber, J. & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10), 2451–2471.
- Graud-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*, 113–119.
- Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research*, 57, 345–420.
- Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. Cambridge: 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.

- 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.
- Gregor, S. & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly*, 37(2), 337–355.
- Greve, W. & Wentura, D. (1997). *Wissenschaftliche Beobachtung: eine Einführung* (2. Aufl.). Weinheim: Beltz.
- Gridach, M. (2017). Character-level neural network for biomedical named entity recognition. *Journal of Biomedical Informatics*, 70, 85–91.
- 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*, 4132–4140.
- 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.
- Halkidi, M., Batistakis, Y. & Vazirgiannis, M. (2001). On Clustering Validation Techniques. *Journal of Intelligent Information Systems*, 17(2-3), 107–145.
- 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 (Long Papers)*, 1, 1262–1273.
- 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 Computation*, 9(8), 1735–1780.
- 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*, 1, 495–503.
- Hulth, A. (2003). Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural*

- Language Processing*, 216–223.
- IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications* (Std 830-1998).
- Iyyer, M., Manjunatha, V., Boyd-Graber, J. & Daumé III, H. (2015). 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 (Long Papers)*, 1, 1681–1691.
- Jorge, J., Vieco, J., Paredes, R., Sanchez, J. A. & Benedí, J. M. (2018). Empirical Evaluation of Variational Autoencoders for Data Augmentation. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 96–104.
- 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 (Short Papers)*, 2, 427–431.
- Jurafsky, D. & Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2. Aufl.). New Jersey: Prentice-Hall.
- Kalousis, A. & Hilario, M. (2001). Model Selection via Meta-Learning: A Comparative Study. *International Journal on Artificial Intelligence Tools*, 10(4), 525–554.
- 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* (NIPS 2015) (S. 3294–3302). Red Hook, NY: Curran Associates.
- Kuncheva, L. I. & Rodríguez, J. J. (2014). A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems*, 38(2), 259–275.
- Le, Q. & Mikolov, T. (2014). Distributed Representations of Sentences and Documents.

- In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, 32, 1188–1196.
- Leacock, C., Miller, G. A. & Chodorow, M. (1998). Using Corpus Statistics and WordNet Relations for Sense Identification. *Computational Linguistics*, 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.
- 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*, 488–495.
- Logeswaran, L. & Lee, H. (2018). An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.
- Manning, C. D., Raghavan, P. & Schütze, H. (2008). *An Introduction to Information Retrieval*. Cambridge: 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.
- McAfee, A. & Brynjolfsson, E. (2012). Big Data: The Management Revolution. *Harvard Business Review*, October 2012. 59–68.
- McCallum, A. (2005). Information extraction: Distilling Structured Data from Unstructured Text. *ACM Queue*, 3(9), 48–57.
- McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochem Medica*, 22(3), 276–282.
- Mehra, N. & Gupta, S. (2013). Survey on Multiclas Classification Methods. *International Journal of Computer Science and Information Technologies*, 4(4), 572–576.
- Meshram, S. B. & Shinde, S. M. (2015). A Survey on Ensemble Methods for High Dimensional Data Classification in Biomedicine Field. *International Journal of Computer Applications*, 111(11), 5–7.
- Mihalcea, R. & Tarau, P. (2004). TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 404–411.

- 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)*, 503–512.
- Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.
- 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*, 263–266.
- 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*, 25–30.
- 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*, 195–204.
- Nadeau, D. & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1), 3–26.
- Newman, D., Karimi, S. & Cavedon, L. (2009). External evaluation of topic models. In *Proceedings of the 14th Australasian Document Computing Symposium (ADCS 2009)*, 1–8.
- Newman, Lau, J. H., Grieser, K. & Baldwin, T. (2010). Automatic Evaluation of Topic Coherence. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 100–108.
- 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/> [01.12.2018].
- Pavithra, M. & Parvathi, R. M. S. (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. *International Journal of Information Technology and Computer Science*, 11(6), 44–53.
- Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V. & Bragge, J. (2006). The Design Science Research Process: A Model for Producing and Presenting Information Systems Research. In *Proceedings of the 1st International Conference on Design Science in Information Systems and Technology (DESRIST 2006)*, 83–106.
- 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*, 101–102.
- 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)*, 1532–1543.
- Perez, L. & Wang, J. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv preprint arXiv:1712.04621*.
- Perone, C. S., Silveira, R. & Paula, T. S. (2018). Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Polikar, R. (2012). Ensemble Learning. In C. Zhang & Y. Ma (Hrsg.), *Ensemble Machine Learning: Methods and Applications* (S. 1–34). Boston: Springer.
- Pressman, R. S. & Maxim B. R (2015). *Software Engineering: A Practitioner's Approach* (8. Aufl.). New York: McGraw-Hill Education.
- 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*, 127–134.
- 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* (S. 563–594). Boca Raton: Taylor & Francis Group.

- Rehm, F., Klawonn, F. & Kruse, R. (2007). A Novel Approach to Noise Clustering for Outlier Detection. *Soft Computing*, 11(5), 489–494.
- 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*, 1, 308–314.
- 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*, 175–186.
- Resnik, P. (1995). Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1, 448–453.
- Rinaldi, F., Hess, M., Dowdall, J., Mollá, D. & Schwitter, R. (2004). Question answering in terminology-rich technical domains. In M. T. Maybury (Hrsg.), *New directions in question answering* (S. 133–140). Menlo Park: AAAI Press.
- Robertson, S. & Robertson, J. (2012). *Mastering the Requirements Process: Getting Requirements Right* (3. Aufl.). Upper Saddle River: Addison-Wesley.
- Sagi, O. & Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4), 1249–1267.
- Saha, S. K., Narayan, S., Sarkar, S. & Mitra, P. (2010). A composite kernel for named entity recognition. *Pattern Recognition Letters*, 31(12), 1591–1597.
- Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513–523.
- 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*, 298–307.
- Schneider, J. (2017). Topic Modeling based on Keywords and Context. *arXiv preprint arXiv:1710.02650*.
- Schutt, R. & O’Neil, C. (2013). *Doing Data Science: Straight Talk from the Frontline*. Sebastopol: O'Reilly Media.
- Schütze, H. (1998). Automatic Word Sense Discrimination. *Computational Linguistics*,

- 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 (Long Papers)*, 1, 86–96.
- Serban, F., Vanschoren, J., Kietz, J.-U. & Bernstein, A. (2013). A survey of Intelligent Assistants for Data Analysis. *ACM Computing Surveys (CSUR)*, 45(3), 31:1–31:35.
- 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.
- Sokolova, M. & Lapalme, G. (2009). A Systematic Analysis of Performance Measures for Classification Tasks. *Information Processing & Management*, 45(4), 427–437.
- Sussillo, D. & Abbott, L. F. (2014). Random Walk Initialization for Training Very Deep Feedforward Networks. *arXiv preprint arXiv: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*, 1017–1024.
- Todor, A., Lukasiewicz, W., Athan, T. & Paschke, A. (2016). Enriching Topic Models with DBpedia. In C. Debruyne, H. Panetto, R. Meersman, T. Dillon, E. 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*, 444–455.
- Tsai, C.-W., Lai, C.-F., Chiang, M.-C. & Yang, L. T. (2014). Data Mining for Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16(1), 77–97.
- 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*, 2049–2054.
- Unterstein, M. & Matthiessen, G. (2013). *Anwendungsentwicklung mit Datenbanken* (5. Aufl.). Berlin: Springer.

- Van Der Maaten, L. J. P., Postma, E. O. & Van den Herik, H. J. (2009). Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*, 10, 66–71.
- Vanschoren, J. (2010). *Understanding Machine Learning Performance with Experiment Databases* (Phd Dissertation). Katholieke Universiteit Leuven, Belgien.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & 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* (NIPS 2017) (S. 5998–6008). Red Hook, NY: Curran Associates.
- Vieira, S. M., Kaymak, U. & Sousa, J. M. C. (2010). Cohen's Kappa Coefficient as a Performance Measure for Feature Selection. In *Proceedings of the 2010 IEEE International Conference on Fuzzy Systems* (FUZZ-IEEE 2010), 924–931.
- Vijaymeena, M. K. & Kavitha, K. (2016). A Survey on Similarity Measures in Text Mining. *Machine Learning and Applications: An International Journal (MLAIJ)*, 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* (KDD'11), 448–456.
- 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(Part B), 806–814.
- 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*, 2557–2563.
- 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.
- Webster, J. & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, 26(2), xiii–xxiii.

- Wong, W., Liu, W. & Benamoun, M. (2012). Ontology Learning from Text: A Look Back and into the Future. *ACM Computing Surveys*, 44(4), 20:1–20:36.
- Wu, Z. & Palmer, M. (1994). Verbs Semantics and Lexical Selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, 133–138.
- Xu, J. A. & Araki, K. (2006). A SVM-based personal recommendation system for TV programs. In H. Feng, S. Yang & Y. Zhuang (Hrsg.), *The 12th International Multi-Media Modelling Conference (MMM2006)* (S. 401–404). Beijing: IEEE Press.
- 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*, 2145–2158.
- Yan, T. W. & Garcia-Molina, H. (1995). SIFT: A Tool for Wide-area Information Dissemination. In *Proceedings of the 1995 USENIX Technical Conference Proceedings*, 177–186.
- 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.
- Zhang, X. & LeCun, Y. (2015). Text Understanding from Scratch. *arXiv preprint arXiv:1502.01710*.
- Zhang, Y., Rahman, M. M., Braylan, A., Dang, B., Chang, H.-L., Kim, H., McNamara, Q., Angert, A., Banner, E., Khetan, V., McDonnell, T., Nguyen, A. T., Xu, D., Wallace, B. X. & Lease, M. (2016). Neural Information Retrieval: A Literature Review. *arXiv preprint arXiv:1611.06792*.

Anhang

Suchanfragen zum Crawling der Ausgangsdatenbasis

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 5.1 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. Additional ist in untenstehender Tabelle zu erwähnen, dass bei den Suchanfragen der Datenbank *arXiv* ein Platzhalter „{}“ eingefügt ist, welcher jeweils durch die kategorische Einschränkung „(Category(astro-ph*) OR cond-mat* OR gr-gc* OR hep-ex* OR hep-lat* OR hep-ph* OR hep-th* OR nlin* OR nucl-ex* OR nucl-th* OR quant-ph* OR math* OR physics*))“ ersetzt wird.

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) |

| Datenbank | Anfrage |
|------------------|---|
| <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> | Abstract(clustering AND data) AND Title(clustering) ANDNOT {} |
| <i>arXiv</i> | Abstract(regression AND data) AND Title(regression) ANDNOT {} |
| <i>arXiv</i> | Abstract(clustering) AND Title(clustering) ANDNOT {} |
| <i>arXiv</i> | Abstract(regression) AND Title(regression) ANDNOT {} |
| <i>arXiv</i> | Abstract(classification) AND Title(classification) ANDNOT {} |
| <i>arXiv</i> | Abstract('regression is') AND Title(regression) ANDNOT {} |
| <i>arXiv</i> | Abstract('classification is') AND Title(classification) ANDNOT {} |
| <i>arXiv</i> | Abstract('frequent pattern' AND data) AND Title('frequent pattern') |
| <i>arXiv</i> | Abstract('frequent pattern' AND data) AND Title('frequent pattern') ANDNOT {} |
| <i>arXiv</i> | Abstract('frequent pattern') AND Title('pattern') ANDNOT {} |
| <i>arXiv</i> | Abstract('association rule' AND data) AND Title('association rule') |
| <i>arXiv</i> | Abstract('association rule' AND data) AND Title('association rule') ANDNOT {} |
| <i>arXiv</i> | Abstract('association rule') AND Title('association rule') ANDNOT {} |
| <i>arXiv</i> | Abstract('sequential pattern' AND data) AND Title('sequential pattern') |
| <i>arXiv</i> | Abstract('sequential pattern' AND data) AND Title('sequential pattern') ANDNOT {} |
| <i>arXiv</i> | Abstract('sequential pattern') AND Title('sequential pattern') ANDNOT {} |
| <i>arXiv</i> | Abstract('pattern mining' AND data) AND Title('pattern') |
| <i>arXiv</i> | Abstract('pattern mining' AND data) AND Title('pattern') ANDNOT {} |
| <i>arXiv</i> | Abstract('pattern mining') AND Title('pattern mining') ANDNOT {} |
| <i>Medium</i> | Clustering |
| <i>Medium</i> | Classification |

Datenbank Anfrage

| | |
|---------------|------------------------|
| <i>Medium</i> | Regression |
| <i>Medium</i> | Frequent Pattern |
| <i>Medium</i> | Association rules |
| <i>Medium</i> | Market-Basket Analysis |
| <i>Medium</i> | Sequential Pattern |
| <i>Medium</i> | Sequence Mining |

Evaluation der Keyword-Extraction-Methoden auf augmentierten Daten

In der folgenden Tabelle werden analog zu Kapitel 7.2.2 die Ergebnisse der Keyword-Extraction-Evaluation auf dem augmentierten Datensatz gezeigt. Hierbei ist festzustellen, dass auf diesen Daten TFIGM als alleinig applizierter Algorithmus die besten Ergebnisse erzielt. Gleichzeitig zeigt sich ebenfalls, dass auch an zweiter Stelle ein einzelner Algorithmus (TF-IDF) aufzufinden ist. Insgesamt ist zudem zu erkennen, dass sich die Top Modelle auf den augmentierten Daten aus den Algorithmen TF-IDF sowie TF-IGM bestehen.

Tabelle A 2: Kohärenzwerte der Keyword-Modelle auf augmentierten Daten

| Kombination | Topic | FT | LC | WP | Σ |
|-----------------|----------------|--------------|----------|----------|--------------|
| TF-IGM | clustering | 0,997 | 1 | 1 | 2,997 |
| TF-IGM | prediction | 0,997 | 0,917 | 0,953 | 2,868 |
| TF-IGM | pattern_mining | 0,997 | 0,900 | 0,938 | 2,835 |
| TF-IDF | prediction | 0,985 | 0,898 | 0,939 | 2,822 |
| TF-IDF | pattern_mining | 0,986 | 0,887 | 0,928 | 2,801 |
| TF-IDF | clustering | 0,987 | 0,860 | 0,891 | 2,738 |
| TF-IDF + TF-IGM | pattern_mining | 0,986 | 0,853 | 0,901 | 2,740 |
| TF-IDF + TF-IGM | clustering | 0,987 | 0,860 | 0,891 | 2,738 |
| TF-IDF + TF-IGM | prediction | 0,985 | 0,847 | 0,899 | 2,731 |

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

Derselbe Sachverhalt ist weiterhin in nachfolgender Abbildung zu erkennen. Additional gibt zeigt diese allerdings eine insgesamt hohe Güte der Algorithmen auf den augmentierten Daten an.

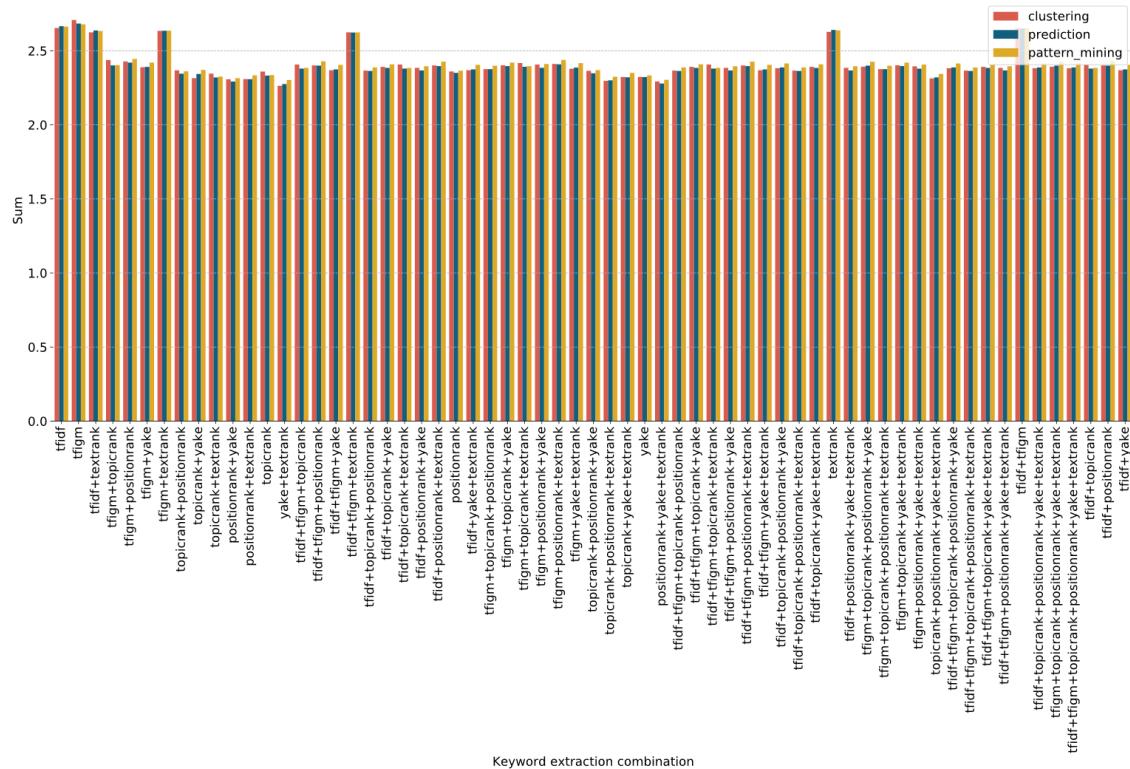


Abbildung A 1: Evaluationswerte von Keyword-Modellen auf augmentierten Daten (Eigene Darstellung)

Gesamtübersicht des Klassendiagramms

In folgender Abbildung wird das Klassendiagramm des programmierten Prototyps visualisiert. Konträr zu Abbildungen im Verlauf der Arbeit wird hier eine Gesamtübersicht bereitgestellt, die alle Klassen und Methoden umfasst.

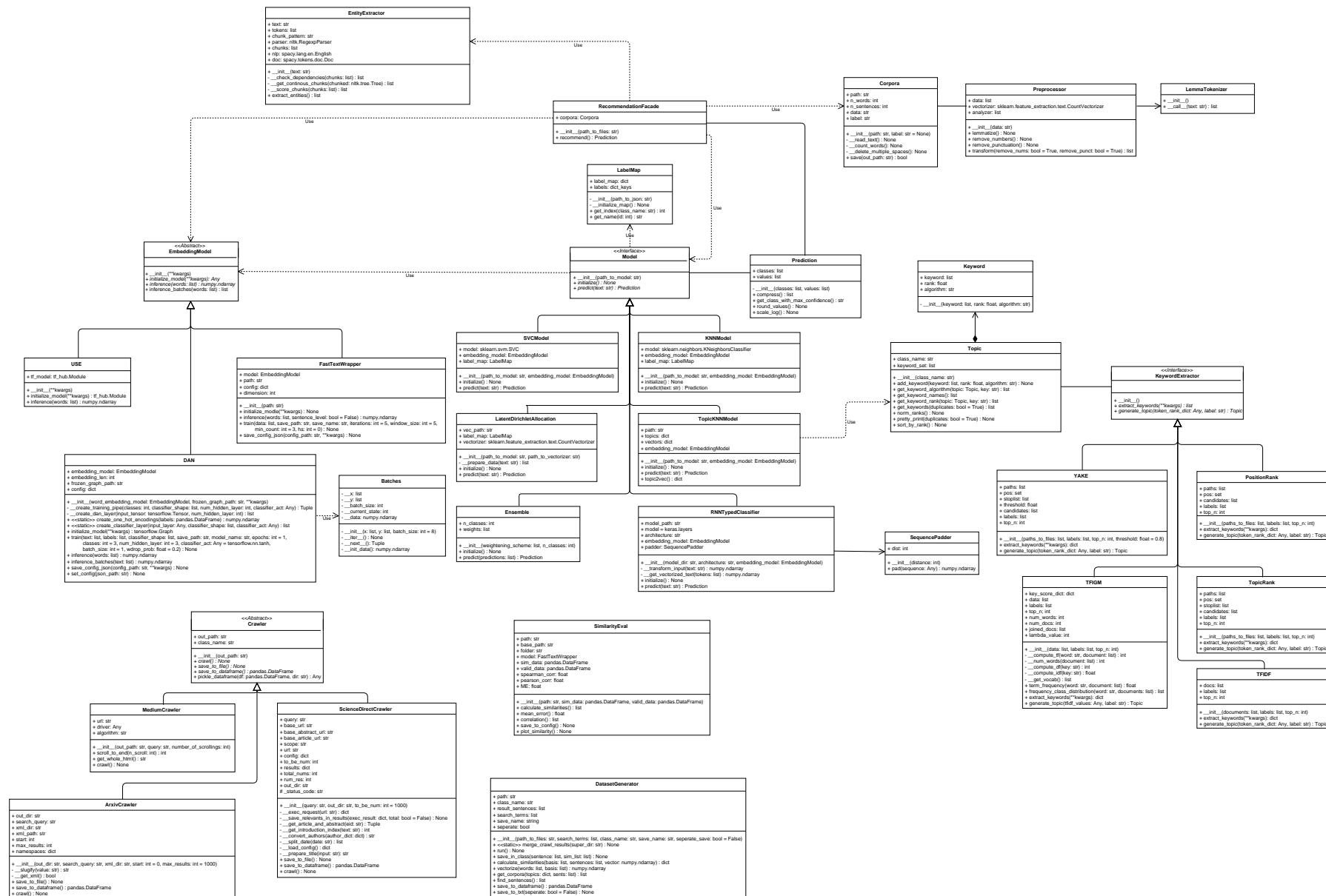


Abbildung A 2: Klassendiagramm des Prototyps (Eigene Darstellung)

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, 31.03.2019