



Evaluation der Einsatzmöglichkeiten ausgewählter Machine-Learning-Verfahren im Kontext von IT-Governance-Prozessen, insbesondere im Application-Portfolio-Management

Master-Thesis

zur Erlangung des akademischen Grades Master of Science (M. Sc.)
Studiengang Wirtschaftsinformatik
Fachbereich 2: Informatik und Ingenieurwissenschaften

Von: Felix Gündling
Feldbergstraße 7
60323 Frankfurt am Main

E-Mail: fguendli@stud.fra-uas.de
Matrikelnummer: 1256976

Gutachter: Prof. Dr. Jürgen Jung
Betreuer: Dr. Wolf Pfannenstiel
Abgabe am: 28.12.2020

Vorwort

Die vorliegende Master-Thesis wurde in Kooperation mit der Dangelmayer & Seemann GmbH (Abk.: DS) erstellt. Die Problemstellung der Arbeit stammte von einem Kunden von DS. Es handelte sich bei dem Kunden um ein multinationales Großunternehmen. DS betreute das Unternehmen in beratender Funktion. Auf die Nennung von Details zum Unternehmen wird verzichtet. Es werden in dieser Arbeit keine vertraulichen Daten veröffentlicht, die einen Rückschluss auf das Unternehmen zulassen würden. Das Vorwort soll neben der Kurzbeschreibung des Umfelds auch für eine Danksagung genutzt werden: Ich bedanke mich herzlich bei allen Mitarbeitern der Dangelmayer & Seemann GmbH, insbesondere bei Olaf Seemann, Dr. Wolf Pfannenstiel, Dr. Nedialka Bubner, sowie bei Andrew Smart für die Unterstützung während der Bearbeitungszeit.

Inhaltsverzeichnis

Vorwort	I
Abbildungsverzeichnis	IV
Tabellenverzeichnis	VI
1 Einleitung	1
1.1 Forschungsfrage- und Methode	2
1.2 Ziele und Aufbau der Arbeit	2
1.3 Themeneingrenzung	3
2 Grundlagen	4
2.1 Künstliche Intelligenz, Machine Learning & Deep Learning	4
2.2 Natural Language Processing	7
2.2.1 Vorverarbeitung von Texten	7
2.2.2 Repräsentation von Texten	8
2.2.3 Regelbasierte Ansätze	9
2.2.4 Vektorbasierte Ansätze	10
2.2.5 Transformer-Ansätze	13
2.3 Das Application-Portfolio-Management als Teil der IT-Governance	13
2.3.1 COBIT als mögliche Grundlage für das APM?	14
2.3.2 Die Rolle von Textdokumenten für die IT-Governance	15
2.3.3 Das APM und verwandte Begriffe	16
2.3.4 Handlungsfelder des APM	19
2.3.5 Unterstützung von APM durch Tools	21
3 Relevante Verfahren und Kriterien für die Evaluation	22
3.1 Verfahren zur Ermittlung von Ähnlichkeiten	22
3.2 Verfahren zur Bestimmung von inhaltlicher Integrität	23
3.3 Verfahren zur automatisierten Erzeugung von Zusammenfassungen	24
3.4 Verfahren zur automatisierten Schnittstellen-Erkennung	27
3.5 Zu untersuchende Kriterien	28
4 Ergebnisse der Evaluation	29
4.1 Ergebnisse: Ermittlung von Ähnlichkeiten	29
4.1.1 Ansatz 1: Ähnlichkeitsvergleich mit einem Word2Vec-Modell . .	30
4.1.2 Ansatz 2: Ähnlichkeitsvergleich unter Verwendung des Paragraph- Vector-Modells (Doc2Vec)	34
4.2 Ergebnisse: Bestimmung von inhaltlicher Integrität	37
4.2.1 Ansatz 1: Vergleich von Durchschnitts-Wort-Vektoren	37

4.2.2	Ansatz 2: Topic Modelling mit Non-negative matrix factorization	40
4.3	Ergebnisse: Automatisierte Erzeugung von Text-Zusammenfassungen	44
4.3.1	Ansatz 1: Nutzung eines extraktiven Verfahrens auf Basis von TF-IDF	44
4.3.2	Ansatz 2: Nutzung eines abstraktiven Verfahrens auf Transformer-Basis	47
4.4	Ergebnisse: Automatisierte Schnittstellen-Erkennung	50
4.4.1	Ansatz 1: Klassifizierung auf Grundlage eines BERT-Modells	50
4.4.2	Ansatz 2: Klassifizierung mit logistischer Regression auf Basis von N-Grammen	53
4.5	Zusammenfassung der Ergebnisse	56
5	Fazit und Ausblick	57
Literatur		i
Eidesstattliche Versicherung		vi
Anhang		vii

Abbildungsverzeichnis

1.1	Bewertung der Relevanz von künstlicher Intelligenz. Quelle: Statista [63]	1
2.1	Zusammenhang von KI, ML und Deep Learning. Eigene Erstellung in Anlehnung an [34]	4
2.2	Ein neuronales Netz mit je einem Input-, Hidden- und Output-Layer, aus [44]	6
2.3	Funktionsweise von Word2Vec (links) und von Paragraph Vector (rechts), aus [35]	12
2.4	Vektoren der Begriffe “the“ & “and“. Eigene Bildschirmaufnahme.	12
2.5	COBIT-Kernmodell, aus [30, S. 21]	15
2.6	EA-Überblick, aus [7, S. 13]	17
3.1	Potenziell mögliche Befragung, eigener Entwurf in Anlehnung an [10]	26
3.2	Schnittstellen-Visualisierung (Interface Circle Map) aus LeanIX, aus [29]	27
4.1	Darstellung verschiedener Wörter in 2 Dimensionen , eigene Erstellung.	30
4.2	Similarity Berechnung mit spaCy. Die Texte wurden gekürzt.	31
4.3	Heatmap zur Dokumentenähnlichkeit, eigene Erstellung	32
4.4	Heatmap zur Dokumentenähnlichkeit, mit Stopwortentfernung, eigene Erstellung	32
4.5	Darstellung aller verfügbaren Dokumente, gruppiert nach Dokumententypen, eigene Erstellung	35
4.6	Dokumenten-Cluster, erzeugt über Doc2Vec, aus [9]	36
4.7	Initiale Vorbereitung von durchschnittlichen Wort-Vektoren, eigene Erstellung	37
4.8	Inferenz-Phase. Vergleich von Kapitel-Vektoren mit Durchschnitts-Vektoren, eigene Erstellung	38
4.9	Tag Clouds zu 4 erkannten Topics, erstellt mit NMF (eigene Erstellung)	41
4.10	Verwendung eines Abschnitts aus einem Testdokument für die Vorhersage des Topics, eigene Erstellung	42
4.11	Struktur der Daten, die für Evaluation in diesem Abschnitt genutzt wurden (geschwärzt und gekürzt)	43
4.12	Evaluation der Güte von Zusammenfassungen, unter Veränderung des Ausgangs-Text-Korpus. Eigene Erstellung	46
4.13	Veranschaulichung eines abstraktiven Verfahrens. Eigene Erstellung in Anlehnung an [49]	47
4.14	Klassifizierungsergebnisse unter Verwendung von BERT nach der Feinabstimmung. Eigene Bildschirmaufnahme	51
4.15	Klasse 0 (“kein TI“) gewinnt und bestimmt damit die Vorhersage. Eigene Bildschirmaufnahme	52

4.16 Beispiele, bei denen Klasse 1 als Vorhersage bestimmt wurde. Eigene Bildschirmaufnahme	52
4.17 Klassifizierung mit logistischer Regression. Eigene Erstellung	54
4.18 Ergebnisse der Klassifizierung von Sätzen aus der Testmenge. Eigene Bildschirmaufnahme	54
4.19 Einige Vorhersagen für ein Evaluations-Dokument. Eigene Bildschirmaufnahme	55

Tabellenverzeichnis

2.1	Beispiel für eine TDM. Fiktives Beispiel in Anlehnung an [3, S. 62]	8
3.1	Ausgangssituation bei der Themenerkennung	24
4.1	Vergleich der Durchschnitts-Vektoren miteinander. Der Vergleich erfolgte mit der Similarity-Funktion von spaCy.	39
4.2	Konfusionsmatrix zur Darstellung der NMF-Schätzungen	42
4.3	Zusammenfassung der Evaluations-Ergebnisse	56

1 Einleitung

Wenn Menschen von künstlicher Intelligenz (KI) hören, dann denken nicht wenige an Roboter, die sich intelligent verhalten [20, S. 1]. Mit Robotern wird schnell auch die Automatisierung von Abläufen in Verbindung gebracht. Aus Automatisierung folgen Effizienz und Produktivitätssteigerungen. In Folge dessen ist es nicht verwunderlich, dass mehr als die Hälfte der in einer Studie befragten Entscheider künstliche Intelligenz für das eigene Unternehmen als ein Thema mit großer oder sehr großer Relevanz bewerten, wie eine Statista-Studie (vgl. Abbildung 1.1) zeigt.

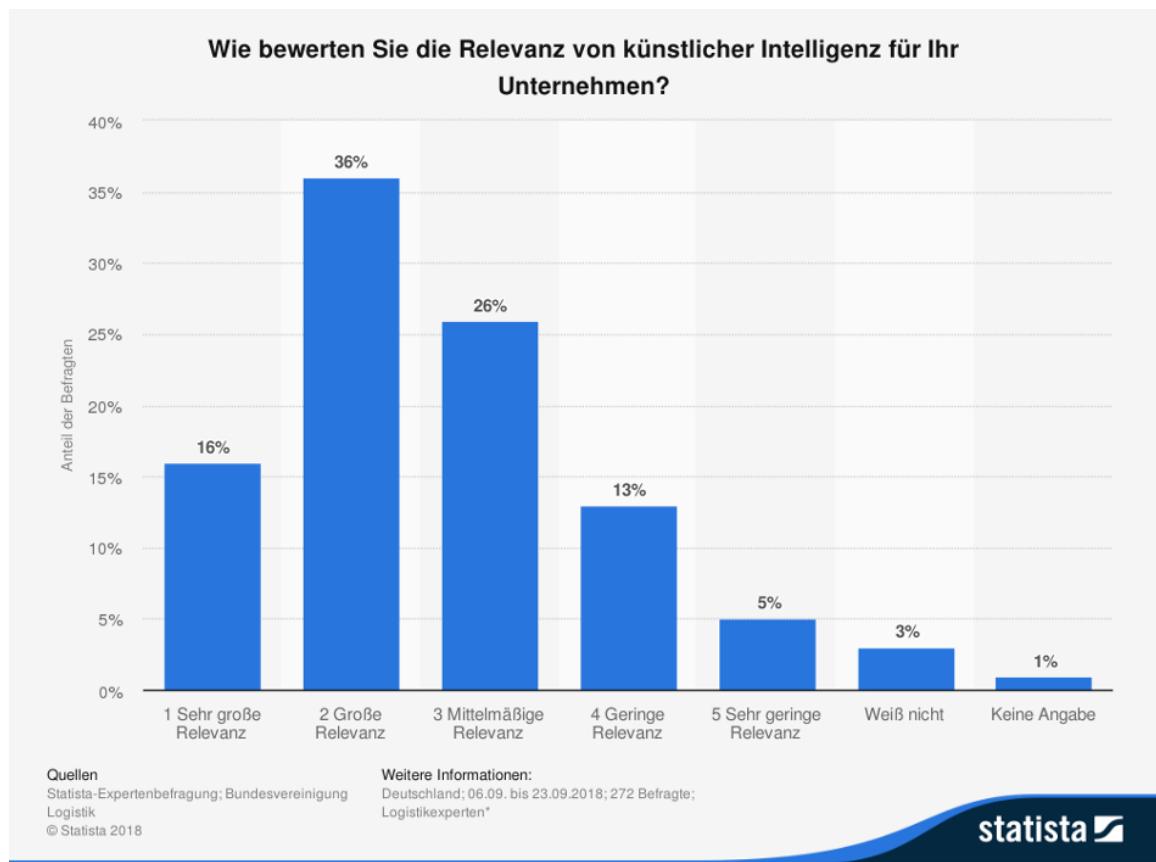


Abbildung 1.1: Bewertung der Relevanz von künstlicher Intelligenz. Quelle: Statista [63]

Da in der Statista-Studie Experten der Logistikbranche befragt wurden, haben möglicherweise viele Teilnehmer tatsächlich an die Optimierung der Betriebskosten durch den Einsatz von Robotern in den Verteilzentren oder an selbstfahrende Lieferwägen gedacht.

1.1 Forschungsfrage- und Methode

Der Gedanke liegt allerdings nahe, dass auch an Ansätze aus dem Gebiet des maschinellen Lernens (Engl.: Machine-Learning, Abk.: ML) gedacht wurde, um bestimmte organisatorische oder prozessuale Abläufe zu unterstützen. Gemeint sind hier Abläufe in der Verwaltung, im Vertrieb oder im Marketing, oder auch in der IT-Governance.

1.1 Forschungsfrage- und Methode

Der Fokus dieser Arbeit liegt speziell auf dem Application-Portfolio-Management (APM) als Teilbereich der IT-Governance. Es sollen in dieser Arbeit Maßnahmen zur Erreichung bestimmter Ziele aus dem Bereich des APM formuliert werden. Dabei soll ganz konkret evaluiert werden welche Maßnahmen zur Zielerreichung geeignet sind, und welche nicht. Insbesondere die Chancen, die durch den Einsatz bestimmter Machine-Learning-Verfahren entstehen, sind im Kontext des APM zu bewerten. Die zu beantwortende Frage lautet demnach:

Können ausgewählte Machine-Learning-Verfahren sinnvoll im Kontext des APM eingesetzt werden und einen echten Mehrwert erbringen?

Die Forschungsfrage wird durch eigene Untersuchungen und Tests beantwortet. Da in einem Unternehmenskontext gearbeitet wird, sollen verschiedene Verfahren auf reale Problemstellungen angewendet werden. Es wird explorativ vorgegangen. Ansätze werden in Python implementiert und damit ausprobiert, im Anschluss erfolgt eine Bewertung der Ergebnisse. Für die Bewertung verwendet werden objektive Kriterien, die in Abschnitt 3.5 beschrieben werden. Der Forschungsansatz orientiert sich an der Methode Action Research [5].

1.2 Ziele und Aufbau der Arbeit

Das eigentliche Ziel der Arbeit ist eine Untersuchung von ML-Verfahren, die für die Unterstützung des APM in Frage kommen. Dabei wird zunächst beschrieben, wie die jeweiligen Verfahren einen Mehrwert für das APM erbringen können. Die Verfahren sollen dann evaluiert und miteinander verglichen werden, was letztendlich zu einer Bewertung

1.3 Themeneingrenzung

von verschiedenen Verfahren führt. Es sollen damit Lösungsansätze für den Umgang mit Applikationsportfolios aufgezeigt werden. Wie später gezeigt wird, spielen bestimmte Text-Dokumente eine wesentliche Rolle im APM. Daher kommen insbesondere Verfahren zur Auswertung von Text-Dokumenten in Frage.

Nach der Einführung folgt ein Grundlagenkapitel, das die notwendigen Begriffe definiert und das Thema der Arbeit detailliert beschreibt. Es wird so gut wie möglich auf aktuelle Fortschritte aus der Forschung eingegangen, wobei zu berücksichtigen ist, dass sich das ML-Gebiet in sehr hoher Geschwindigkeit weiterentwickelt [68, S. 146] und aus diversen Richtungen beeinflusst wird. Es soll außerdem beschrieben werden, mit welchen Fragestellungen sich das APM beschäftigt. Eine Abgrenzung zur IT-Governance wird dabei vorgenommen. Im Anschluss an den Theorienteil wird das Umfeld beschrieben, in dem die Evaluation stattfindet. Es wird außerdem beschrieben, auf welche Art und Weise eine Evaluation von ML-Verfahren im gegebenen Umfeld durchgeführt werden kann. Es werden Kriterien für die Evaluation aufgestellt. Die Ergebnisse der Evaluation werden danach beschrieben. Das Fazit beendet die Arbeit.

1.3 Themeneingrenzung

Es wird in der vorliegenden Arbeit nicht vollumfänglich die historische Entwicklung des Themenbereichs KI diskutiert. Ebenfalls nicht behandelt werden philosophische, moralische oder ethische Fragestellungen, sofern diese nicht unmittelbare Auswirkungen haben für das Thema IT-Governance oder das Application-Portfolio-Management. Es soll aber ausdrücklich betont werden, dass die Nutzung von KI-Systemen das Potenzial hat, die Arbeits- und Lebensrealität von Menschen unmittelbar zu beeinträchtigen. Für eine ausführliche Diskussion dazu wird allerdings auf die einschlägige Literatur verwiesen, z. B. auf [14, S. 11-16]. Die Master-Thesis soll sich hingegen inhaltlich auf aktuelle technische und für den Anwendungsfall relevante betriebswirtschaftliche Aspekte konzentrieren. Aufgrund der begrenzten Bearbeitungszeit kann nur eine gewisse Auswahl an ML-Verfahren berücksichtigt werden.

2 Grundlagen

Ein Verständnis der Möglichkeiten ist der Schlüssel [...] zur Gestaltung und Durchführung von Text-Analysen [3, S. vii].

2.1 Künstliche Intelligenz, Machine Learning & Deep Learning

Abbildung 2.1 gibt einen groben Überblick darüber, wie die in dieser Überschrift genannten Begriffe zu verstehen sind und in welcher Beziehung sie zueinander stehen.

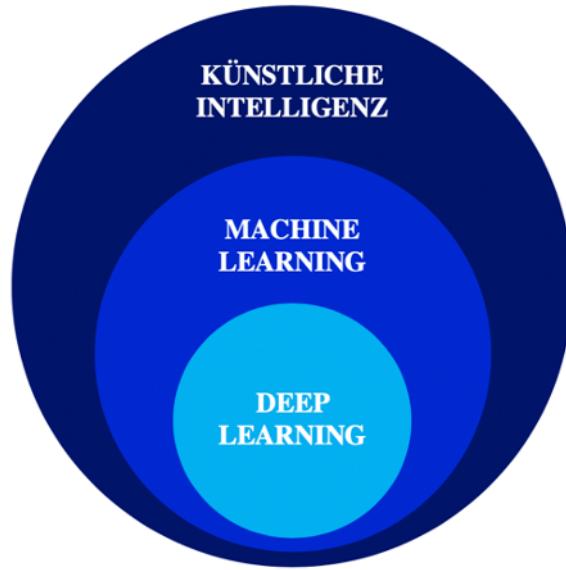


Abbildung 2.1: Zusammenhang von KI, ML und Deep Learning. Eigene Erstellung in Anlehnung an [34]

Das Ziel der künstlichen Intelligenz ist es Systeme aufzubauen, die Aufgaben lösen können, die Intelligenz auf Niveau des Menschen voraussetzen. Machine Learning wird dabei als Teilmenge bzw. Teildisziplin der künstlichen Intelligenz eingeordnet [68, S. 14].

Machine Learning ermöglicht es Computern aus Erfahrungen beziehungsweise aus Aufzeichnungen Regeln zu lernen, ohne explizit programmiert zu werden [54]. Gängig ist im Bereich Machine Learning eine Unterscheidung zwischen drei Arten des Lernens: dem überwachten Lernen, dem unüberwachten Lernen, und dem sogenannten Reinforcement Learning. Beim überwachten Lernen existiert in Trainingsdaten für jedes Beispiel eine Beschriftung. Mit Hilfe der Beschriftungen (oder auch Klassen) kann nun beispielsweise ein Klassifikationsmodell gelernt werden kann. Das so erzeugte Modell wird anschließend genutzt, um eine Vorhersage der Klassen von neuen, unbekannten Datensätzen zu realisieren. Auch die Erstellung von Regressionsanalysen zählt zur Kategorie des überwachten Lernens. Das unüberwachte Lernen verzichtet auf beschriftete Beispiele. Entsprechende Verfahren können etwa Cluster bilden, die ähnliche Datensätze zusammenfassen (Clustering). Auch die Ausreißererkennung, sowie das Gebiet des Association Rule Learning fallen in die Kategorie des unüberwachten Lernens. Beim Reinforcement Learning werden sogenannte Belohnungen (Rewards) und Strafen (Penalties) verwendet, um beispielsweise Bots das Spielen von Computerspielen beizubringen [20, S. 7-15].

Deep Learning (DL) beschreibt eine Gruppe von Verfahren innerhalb der Domäne des maschinellen Lernens, wobei neuronale Netzwerke eingesetzt werden. Bei künstlichen neuronalen Netzen handelt es sich um eine Gruppe von Machine-Learning-Algorithmen, die inspiriert sind von natürlichen neuronalen Netzen aus der Natur [71, S. 2]. Das Wort deep in Deep Learning bezieht sich auf die Anzahl der Schichten des Netzwerks, durch das Eingaben verarbeitet werden. Abbildung 2.2 zeigt exemplarisch ein einfaches Artificial Neural Network (ANN). Deep Neural Networks (DNNs) sind stets mit mehr als einem sogenannten Hidden Layer ausgestattet [44].

Es gibt neben ANNs und DNNs noch weitere Architekturansätze zur Realisierung von neuronalen Netzen. Einige davon, wie Recurrent Neural Networks (RNNs), sind für bestimmte Aufgaben besser geeignet als andere. RNNs sind insbesondere für die Verarbeitung von natürlicher Sprache von Bedeutung. Es können verschiedene Formen von RNNs unterschieden werden, wobei für den Anwendungsfall der Sprachverarbeitung insbesondere Encoder-Decoder-RNNs hervorzuheben sind, die auch als seq2seq-Modelle bezeichnet werden und besonders gut mit Sequenzen (z. B. Abfolgen von Wörtern) umgehen können. Gemeint ist hier etwa eine Übersetzung, die eine Sequenz als Eingabe benötigt und eine Übersetzung als Sequenz wieder ausgibt. [71, S. 23–28] Convolutional Neural Networks (CNNs) sind hingegen unter anderem für den Bereich Computer Vision (Bildverarbeitung)

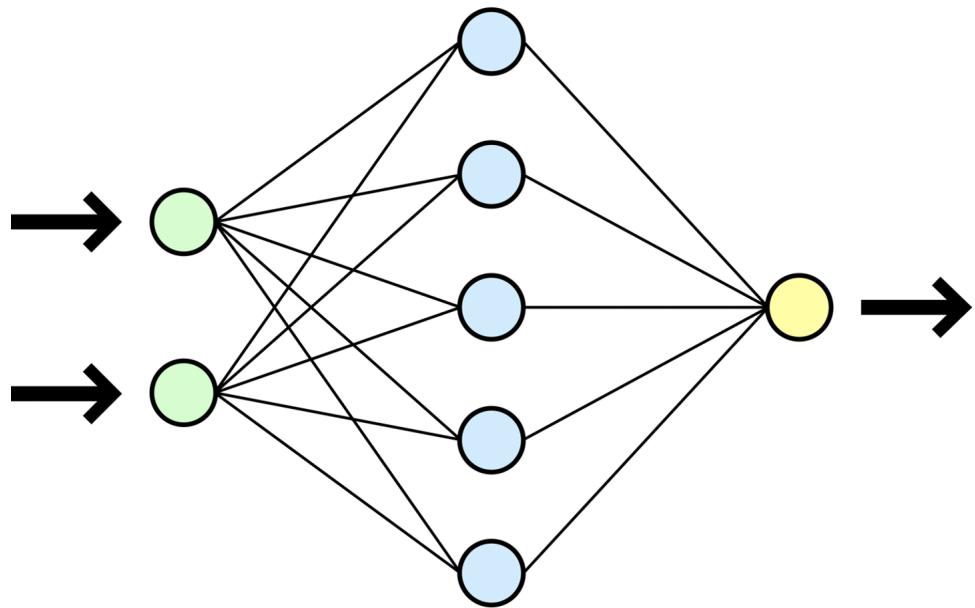


Abbildung 2.2: Ein neuronales Netz mit je einem Input-, Hidden- und Output-Layer, aus [44]

Bilderkennung) von großer Bedeutung. Sie wurden explizit für diese Aufgabe entwickelt [36].

Die verschiedenen neuronalen Netze haben die Gemeinsamkeit, dass sie Neuronen (Knoten) und gewichtete Verbindungen (Kanten) zwischen den Neuronen beinhalten. Neuronen werden aktiviert durch Eingaben und verarbeiten diese mit bestimmten Funktionen weiter, um wiederum eine Ausgabe zu generieren. Das Ermitteln der Gewichtungen dieser Verbindungen wird als Training bezeichnet. Das Training des Netzes besteht aus dem Lösen eines hoch-dimensionalen, nichtlinearen, nicht-konvexen und globalen Optimierungsproblems. Es kann erfolgen unter Zuhilfenahme der Algorithmen Gradient Descent und Backpropagation [71, S. 1–13].

Während bei herkömmlichen Machine-Learning-Ansätzen Features, die während der Verarbeitung benötigt werden, manuell von Menschen gebildet werden müssen, werden diese Features bei Deep-Learning-Ansätzen durch das neuronale Netz erzeugt [2, S. 3-4]. Durch diesen Vorteil sind bessere Ergebnisse möglich [2, S. 11].

2.2 Natural Language Processing

Es geht nachfolgend um Techniken zur Verarbeitung und Auswertung von Texten und natürlicher Sprache (Engl.: Natural Language Processing, Abk.: NLP). NLP kann dabei als Teil des umfangreichen Gebiets Text Mining verstanden werden [48, S. 17]. Das Verständnis von natürlicher Sprache (Engl.: Natural Language Understanding, Abk.: NLU) gilt als Voraussetzung für die Schaffung eines vollständigen KI-Systems, wodurch der Bezug von Sprachverständnis bzw. der Sprachverarbeitung zur KI gegeben ist [72, S. 8].

2.2.1 Vorverarbeitung von Texten

Die Vorverarbeitung (Engl.: Preprocessing) von Texten gilt als Voraussetzung für den Erfolg der späteren Auswertungen [3, S. 46]. Daher werden hier zunächst einige der gängigen Schritte zur Vorverarbeitung von Texten skizziert und grundlegende Methoden erläutert.

Bei der Token-Erstellung (Engl.: Tokenization) wird Text in sogenannte Tokens unterteilt. Im einfachsten Fall bestehen Tokens aus einem Wort – dann handelt es sich um Unigramme (Engl.: unigram). N-Gramme (Engl.: n-grams) stellen eine Alternative zu Unigrammen dar. N-Gramme sind Wort-Sequenzen mit der Länge n [3, S. 48]. Stoppwörter dienen zwar einem grammatischen Zweck, in Bezug auf den Inhalt von Texten erbringen sie aber nur einen geringen Mehrwert. Aus diesem Grund werden Stopwörter im Rahmen der Vorverarbeitung in der Regel aus Texten entfernt. Dazu kann z. B. ein Wörterbuch verwendet werden, das bekannte Stopwörter wie z. B. [und, er, ein, hat, ...] enthält [3, S. 50-53]. Die Reduktion von Begriffen auf die Stammform (Engl.: Stemming) verfolgt das Ziel, verschiedene grammatische Varianten des gleichen Wortes zu vereinheitlichen, um die Gesamtanzahl der unterschiedlichen Wörter innerhalb eines Textkörpers zu reduzieren. Die Erzeugung der Stammformen erfolgt z. B. durch den Porter-Stemmer, der Stammformen auf Basis von Regeln bilden kann. Die Lemmatisierung (Engl.: Lemmatization) führt Begriffe auf eine Art “Wörterbuchform“ zurück und kann dadurch besser interpretierbare Ergebnisse liefern als das Stemming [40, S. 30-32] Worttypen wie Adjektive, Verben, Substantive, etc. werden erkannt und markiert durch Part-of-Speech-Tags (Abk.: POS-Tags). Zur Realisierung dieser Markierung können Text-Korpora verwendet werden, bei denen manuell Wörter mit den entsprechenden Wortarten markiert wurden [3, S. 58].

2.2.2 Repräsentation von Texten

Nach den beschriebenen Schritten zur Vorverarbeitung von Textdokumenten kann die Darstellung von Begriffen im Verhältnis zu Dokumenten über eine Term-Document-Matrix (TDM) erfolgen (vgl. Tabelle 2.1). Die transponierte Variante dieser Matrix heißt Document-Term-Matrix (DTM) [3, S. 61ff]. Eine TDM könnte so etwa nach der initialen Vorverarbeitung aussehen.

	Dokument_1	Dokument_2	Dokument_3	...
dog	0	0	2	...
cat	1	1	0	...
bird	3	0	1	...
...

Tabelle 2.1: Beispiel für eine TDM. Fiktives Beispiel in Anlehnung an [3, S. 62]

In dieser Art der Darstellung wird die Menge der Wörter eines Dokuments als Bag-of-Words bezeichnet [3, S. 46]. Die Ausprägungen in den Zellen der Tabelle 2.1 beschreiben in diesem Beispiel die Häufigkeit des Vorkommens der Begriffe in den jeweiligen Dokumenten. Um zur Term Frequency (Abk.: tf) zu gelangen, ist die Anzahl des Begriffs durch die Gesamtanzahl der Begriffe im jeweiligen Dokument zu dividieren. Für den Begriff cat ergibt sich in der ersten Spalte eine tf von $\frac{1}{4}$, unter Vernachlässigung der Spalten und Zeilen ohne kardinale Ausprägungen.

Eine weitere, für die spätere Auswertung relevante Kennzahl heißt Document Frequency (Abk. df). Dabei wird zunächst die Häufigkeit der Begriffe reduziert auf eine binäre Angabe 1 oder 0, die Häufigkeit wird also vernachlässigt und es verbleibt lediglich eine Information darüber, ob ein Wort in den Dokumenten vorkommt oder nicht. Die nun erzeugten Werte werden zeilenweise addiert, um die Kennzahl df zu erhalten. Das Beispiel "cat" aus Tabelle 2.1 würde zu einer df von $1+1+0=2$ führen. Df gibt damit an, in wie vielen Dokumenten ein Begriff vorkommt. Um seltenen, und damit aussagekräftigen Begriffen ein höheres Gewicht zu geben als Begriffen die zwar häufig vorkommen, dafür aber auch weniger Relevanz und Wirkung haben, existiert mit der inversen Dokumenten-Häufigkeit (Engl.: inverse document frequency, Abk.: idf) eine weitere Metrik. Sie wird berechnet als:

$$idf_i = \log_2\left(\frac{n}{df_i}\right) + 1$$

wobei n die Anzahl der Dokumente ist, und df wie oben beschrieben berechnet wird. Die hier beschriebenen Metriken können kombiniert werden zur tf-idf-Metrik. Dabei wird tf mit idf lediglich multipliziert. Für den Beispielbegriff *cat* ergibt sich unter Vernachlässigung der Spalten und Zeilen ohne kardinale Ausprägungen ein Wert von $\frac{1}{4} * (\log_2(\frac{3}{2}) + 1) = 0,39$ für das erste Dokument. Der TF-IDF-Wert von Begriffen kann nun als Ausprägung in den Zellen einer TDM an Stelle der schlichten Anzahl der Begriffe verwendet werden [3, S. 61-73].

Die TF-IDF-Metrik ist zusammengefasst eine Grundlage für die Erkennung von wichtigen Begriffen, unter Betrachtung von mehreren Dokumenten. Dabei sollten die Dokumente möglichst einen Bezug zueinander haben. Ein Beispiel wäre eine Menge von Dokumenten über Software-Entwicklungsprojekte. In jenen Dokumenten würde der Begriff "Software" häufig vorkommen. Die Bedeutung des Begriffs ist allerdings eher gering, da er wahrscheinlich in allen Dokumenten vorhanden sein wird. Beim TF-IDF-Maß erhalten seltene Begriffe einen hohen Wert, wodurch solche Begriffe als interessant identifiziert werden können. [40, S. 107-110]. Auf Basis der einfachen TDM oder auch auf Basis der erweiterten Variante unter Verwendung von TF-IDF-Werten können im Anschluss Auswertungen durchgeführt werden.

2.2.3 Regelbasierte Ansätze

Um ein ML-Modell trainieren zu können, sind Trainingsdaten notwendig (vgl. Kapitel 2.1). Wenn diese nicht in ausreichender Menge vorhanden sind, dann kann ein regelbasierter Ansatz eine Alternative sein. Regelbasierte Systeme zählen zu den ältesten Ansätzen im Bereich NLP und haben sich in der Praxis bewährt. Aus technischer Sicht können für die Realisierung eines regelbasierten Ansatzes etwa String-Vergleiche (z. B. mit regulären Ausdrücken) verwendet werden. Auch mit kontextfreien Grammatiken können Regeln ausgedrückt werden [41]. Ein simpler endlicher Automat kann verwendet werden, um einfache Regeln zu beschreiben [40, S. 219]. Zu den regelbasierten Ansätzen werden hier explizit auch Wenn-Dann-Regeln gezählt [14, S. 23ff.]. Beispiele für Entitäten, die gut über einen regelbasierten Ansatz erkannt werden können, sind etwa IP-Adressen, URLs, oder Telefonnummern [61].

Über Ontologien oder Taxonomien können ebenfalls Entitäten erkannt werden. Solche Wissensbasen galten früher als grundlegender Baustein von KI-Systemen. Ein solcher Bestand

kann zu Beginn eines Projekts erstellt werden und dann im Laufe der Zeit weiterentwickelt werden. Ontologien führen in diesem Kontext zu wiederverwendbaren, erweiterbaren Wissensbeständen. Sie können sowohl von Menschen als auch von Computern verstanden werden und bilden Mengen von explizit definierten Terminologien ab. Die grundlegenden Begriffe, also das Vokabular einer bestimmten Domäne und die Beziehungen der Begriffe untereinander werden in einer Ontologie modelliert [43, S. 37-55]. Das Vokabular könnte in den erwähnten Wenn-Dann-Regeln verwendet werden. Die Begriffe Ontologie und Topic werden in der Literatur und daher auch hier als Synonyme verwendet.

Taxonomien können ebenfalls für solche Klassifizierungszwecke verwendet werden. Eine Taxonomie hat einen Namen und besteht aus Beispielen. Es kann flache bzw. einfache Taxonomien geben, aber auch hierarchische, oder gar vernetzte Taxonomien [28]

Da im Titel dieser Arbeit die Evaluation von ausgewählten ML-Verfahren festgelegt ist, werden regelbasierte Ansätze im praktischen Teil der Arbeit weniger ausführlich betrachtet als „echte“ ML-Ansätze. Sie gelten aber in realen Szenarien als reale Option für bestimmte Anwendungsfälle und wurden daher hier beschrieben.

2.2.4 Vektorbasierte Ansätze

Die Begriffe Wort-Vektor und Wort-Einbettung (Engl.: Word Embedding) werden hier als Synonyme verwendet, wie in [71, S. 38]. Als moderner Algorithmus zur Erzeugung von Wort- Einbettungen gilt Word2Vec. Entwickelt wurde das Verfahren von Google-Forschern [42], allerdings sind auch andere Unternehmen an der Weiterentwicklung dieser Ansätze beteiligt. So gibt es etwa mit fastText von Facebook oder mit GloVe der Stanford University ebenfalls Möglichkeiten zur Nutzung von vorbereiteten Word-Vektoren sowie für das Training eigener Wort-Vektoren [15] [47]. Word2Vec versucht, die Probleme des Bag-of-Word-Ansatzes beziehungsweise der One-Hot-Encodings zu vermeiden. Durch diese Repräsentation – bei der das Vorhandensein eines Wortes in einem Textdokument etwa mit einer 1 angezeigt wird und das Fehlen mit einer 0 – entstehen hoch-dimensionale Tabellen mit vielen Zeilen. Dabei kommt häufig der Wert 0 als Ausprägung vor. Damit einher geht ein gewisser Informationsverlust, da Kontextinformationen (z. B. die Position der Wörter innerhalb der Dokumente) verloren gehen und weil viele ML-Verfahren mit Daten, die in dieser Art und Weise repräsentiert sind, schlecht umgehen können. Zu viele Spalten können zu Overfitting und damit zu einem schlechten Modell führen [57, S. 145]. Als

2.2 Natural Language Processing

Architektur für das Erzeugen von Wort-Vektoren kommt bei Word2Vec ein voll-verknüpftes neuronales Feed-Forward-Netz zum Einsatz. Eine Eigenschaft des Verfahrens ist, dass bei jedem Wort der Kontext des Worts in gewissem Maße erhalten bleibt. Die Anwendung von Word2Vec führt zu den sogenannten Einbettungen, was letztendlich einer deutlich kompakteren Darstellung der Informationen entspricht, als das etwa bei der Darstellung von Text in Form einer Term-Document-Matrix der Fall ist. Eine Einbettung entspricht einer Darstellung von Begriffen in Vektorform. Die mit Word2Vec erzeugten Wort-Vektoren können als Eingabe für weitere Machine-Learning-Verfahren verwendet werden, wobei diese Repräsentation deutlich bessere Resultate verspricht, als in der unverarbeiteten Form. Um die Vektoren zu erzeugen wird ein neuronales Netz trainiert, dass das Ziel hat, die Wahrscheinlichkeit von bestimmten Kontextwörtern einzelner Wörter vorherzusagen (im sogenannten Skip-Gram-Ansatz). Die internen Gewichtungen des neuronalen Netzes werden letztendlich als Wort-Vektoren verwendet. [57, S. 148-160].

Neben Wort-Einbettungen kommen an dieser Stelle noch Dokumenten-Einbettungen in Betracht. Der Algorithmus dazu heißt Paragraph Vector. Im Vergleich zu Word2Vec werden für das Erstellen der Einbettungen zusätzlich zu den Wörtern auch jeweils Paragraphen in das Training der Vektoren mit einbezogen. Die Paragraphen liefern also zusätzlichen Kontext. Vergleichbar ist der Paragraph mit einer Angabe darüber, wo im Dokument bzw. auch in welchem Dokument sich ein Begriff in den Trainingsdaten befunden hat [35]. Eine Implementierung von Dokument-Einbettungen findet sich unter dem Namen doc2vec [51].

In Abbildung 2.3 dargestellt sind einzelne Wörter, die als Vektoren abgebildet werden. Die Abkürzung W steht aus mathematischer Sicht für eine Matrix, die einzelnen Wort-Vektoren sind als Spalten in dieser Matrix enthalten. Nach Bildung der initialen Vektoren werden diese konkateniert (“aufsummiert”), um den sogenannten Kontext abzubilden. In diesem Beispiel wird ein neuronales Netz verwendet, dass die Wahrscheinlichkeit des Wortes “on” im Kontext der Begriffe “the cat sat” ermittelt (Hier dargestellt ist das Continuous Bag of Word-Modell (CBOW), das genaue Gegenteil von Skip-Gram, das oben bereits erwähnt wurde). Bei der Verwendung des Paragraph Vector Ansatzes wird jeder Paragraph als Spalte in einer Matrix D abgebildet. Diese zusätzliche Information ergänzt den beschriebenen Kontext. Was genau als Paragraph definiert wird, kann frei gewählt werden. Es kann sich um längere Abschnitte, ganze Dokumente, oder auch eine Auswahl von Sätzen handeln [35].

2.2 Natural Language Processing

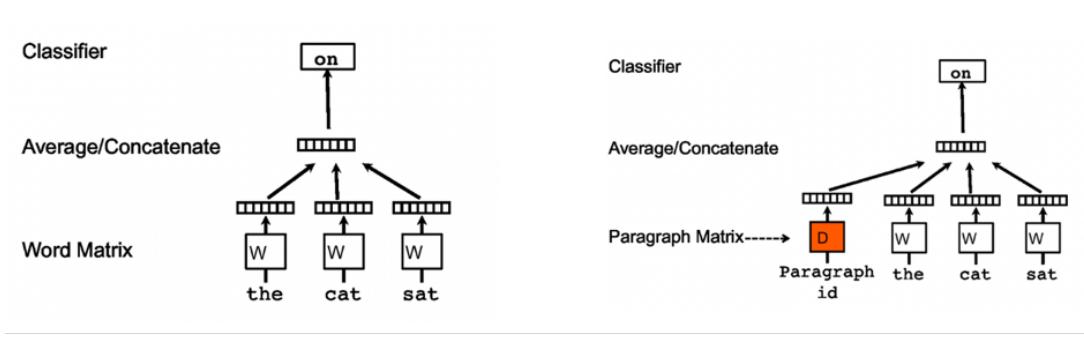


Abbildung 2.3: Funktionsweise von Word2Vec (links) und von Paragraph Vector (rechts), aus [35]

Einbettungen müssen nicht für jede Anwendung neu trainiert werden. Stattdessen können auch vor-trainierte (pre-trained) Einbettungen wiederverwendet werden, wie am Anfang dieses Abschnitts bereits angedeutet wurde. Unten dargestellt (Abbildung 2.4) ist ein Ausschnitt aus dem bereits trainierten GloVe-Modell, der die Vektor-Repräsentation verdeutlichen soll. Es wurde trainiert auf Basis des Korpus Wikipedia 2014 [47].

```

the 0.418 0.24968 -0.41242 0.1217 0.34527 -0.044457 -0.49686 -0.00066023 -0.6566 0.27843 -0.14767 -0.55677 0.14658 -0.0095095 0.011658 0.10204
-0.12792 -0.8443 -0.12181 -0.016801 -0.33279 -0.1552 -0.23131 -0.19181 -0.76746 0.099851 -0.42125 -0.19526 4.0071 -0.18594 -0.52287 -0.31681
0.00059213 0.0074449 0.17778 -0.15897 0.012041 -0.054223 -0.29871 -0.15749 -0.34758 -0.045637 -0.44251 0.18785 0.0027849 -0.18411 -0.11514 -0.78581
, 0.0131441 0.23682 -0.16899 0.40951 0.63812 0.47709 -0.42852 -0.55641 -0.364 -0.23938 0.13001 -0.063734 -0.39575 -0.48162 0.23291 0.090201 -0.13324
0.078639 -0.41634 -0.15428 0.10068 0.48891 0.31226 -0.12612 -0.037512 -1.5179 -0.02442 -0.042961 -0.28351 3.5416 -0.11956 -0.14533 -0.1499
0.21864 -0.33412 -0.13872 0.31806 0.70358 0.44858 -0.080262 0.63083 0.32111 -0.46765 0.22786 0.36834 -0.37818 -0.56657 0.044691 0.30392
. 0.15161 0.30177 -0.16763 0.17684 0.31719 0.33973 -0.43478 -0.31086 -0.44999 -0.29498 0.16608 0.11963 -0.41328 -0.42353 0.59866 0.28285 -0.11547
-0.04184 -0.67989 -0.25063 0.18472 0.086878 0.46582 0.015033 0.043474 -1.4671 -0.30384 -0.023441 0.30589 -0.21785 3.746 0.0042284 -0.18436 -0.46209
0.098329 -0.11987 0.23919 0.1161 0.41703 0.056763 -0.3681e-05 0.668998 0.087939 -0.10285 -0.13934 0.22314 -0.0808083 -0.35659 0.016413 0.10216
of 0.70853 0.57088 -0.16846 0.54443 0.72683 0.1815 -0.52393 0.16381 -0.17566 0.078851 0.36216 -0.11842 0.083386 0.1191 -0.16605 0.061555
-0.01084 -0.01666 0.113656 0.22851 0.367549 -0.37056 -0.23698 1.7137 0.07004 0.26704 0.25186 0.1767 -0.89161 -0.1613 -0.13273 0.09881
0.01444 0.0052464 -0.33874 -0.010956 0.24185 0.36576 -0.24721 0.28481 0.075603 -0.21718 -0.38988 0.22992 -0.21617 -0.72552 0.093918 -0.86375
, 0.68847 -0.039763 0.30186 -0.17792 0.42962 0.032246 -0.1376 0.13728 -0.29847 -0.085253 0.17118 0.22149 -0.1046 0.43653 0.33418 0.67846 0.072704
-0.34448 -0.42785 -0.43275 0.55963 0.10032 0.18677 -0.26854 0.037334 -2.0932 0.22171 0.39868 0.20912 -0.55725 3.8826 0.47466 -0.95658 -0.37788 0.20869
-0.32752 0.12751 0.088359 0.16351 -0.21634 -0.094375 0.018324 0.21048 -0.19272 0.082279 -0.09434 -0.073207 -0.064699 -0.26044
and 0.26818 0.14346 -0.27877 0.016257 0.11384 0.69923 -0.51332 -0.47768 -0.33075 -0.1383 0.2702 0.38938 -0.45812 -0.89932 0.038885 0.029749
0.19076 -0.25958 -0.51818 0.34558 0.44922 0.48791 -0.08866 -0.102121 -1.3777 -0.10866 -0.23201 0.012839 -0.46508 3.8463 0.31362 0.13643 -0.52244 0.3382
0.33707 -0.35601 0.32431 0.128041 0.3512 -0.069043 0.36885 0.25168 -0.24517 0.25381 0.1367 -0.31178 -0.6321 -0.25028 -0.38097
in 0.33842 0.24995 -0.60874 0.18923 0.036372 0.151 -0.55883 -0.074239 -0.092307 -0.32821 0.09598 -0.82269 -0.36717 -0.67809 0.42909 0.016496 -0.23573
0.12864 -1.0953 0.43334 0.57067 -0.1036 0.20422 0.078386 -0.42795 -1.7984 -0.27865 0.11954 -0.12689 0.031744 3.8631 -0.17786 -0.082434 -0.62698 0.26497
-0.057185 -0.073521 0.46183 0.30862 0.12498 -0.48689 -0.0880272 0.031184 -0.36576 -0.42699 0.42164 -0.11666 -0.50703 0.027273 -0.53285
a 0.21705 0.46515 -0.46757 0.10082 1.0135 0.74845 -0.53184 -0.26256 0.16812 0.13182 -0.24909 0.51004 0.13448 -0.43141 -0.03123
0.20674 -0.78138 -0.20148 -0.097401 0.16088 0.61833 -0.18504 -0.12461 -0.2232 0.05403 0.32257 0.15133 0.9363 -0.71365 -0.67012 0.28388 0.21738
0.14433 0.25928 0.23434 0.4274 -0.44451 0.13813 0.36973 -0.64289 0.024142 -0.039315 -0.26863 0.12817 -0.04378 0.41813 0.1796
" 0.25765 0.45629 -0.76974 -0.37679 0.59272 -0.063527 0.20545 -0.57385 -0.29088 -0.13662 0.32728 1.4719 -0.73681 -0.13154 -0.46098 0.65248
0.48887 -0.51558 0.039951 -0.34387 -0.014887 0.86488 0.3546 0.7999 -1.4995 -1.8153 0.41128 0.23921 -0.43139 0.36623 -0.79834 -0.54538 0.16943 -0.82617
-0.3461 0.69495 -0.12256 0.17992 -0.057474 0.038498 -0.39543 -0.38915 -1.0002 0.087599 -0.31089 -0.34677 -0.31438 0.75004 0.97065
" 0.23727 0.40478 -0.28547 0.58805 0.65533 0.32867 -0.81964 -0.23268 0.27428 0.24265 0.054992 0.16296 -0.2555 -0.086437 0.44536 0.096561 -0.16519
0.058378 -0.38598 0.086577 0.0833869 0.55895 -0.77697 -0.62096 0.092948 -0.5685 -0.67739 0.10151 -0.46463 -0.07805 3.1859 -0.107554 -0.16138 0.055486
0.25885 -0.33938 -0.19928 0.268049 0.10478 -0.55934 -0.12342 0.65961 -0.51802 -0.82995 -0.082739 0.28155 -0.423 -0.27378 -0.087901 -0.030231

```

Abbildung 2.4: Vektoren der Begriffe "the" & "and". Eigene Bildschirmaufnahme.

Die größeren NLP-Modelle, etwa von spaCy, beinhalten standardmäßig nutzbare Wortvektoren. Eine eigene Untersuchung ergibt, dass diese Modelle auch gewisse fachspezifische Wörter berücksichtigen, etwa die Begriffe Shipments oder containerization. Allerdings existieren für echte domänen spezifische Begriffe wie Projekt- oder Systemnamen keine Wortvektoren. Solche Begriffe nennt man out-of-vocabulary [62]. Das erschwert die Verwendung von Modellen, die "gebrauchsfertig" zur Verfügung stehen.

2.2.5 Transformer-Ansätze

Transformer werden erstmals eingeführt Ende 2017. Laut den Autoren der Veröffentlichung ist der Ansatz in der Lage bessere Ergebnisse in bestimmten Bereichen zu erzielen als bisherige Ansätze auf Basis von RNNs oder CNNs; zudem ist das Training parallelisierbar und damit schneller als bei früheren Ansätzen [69]. Transformer-Ansätze können für viele NLP-Teilbereiche verwendet werden, etwa für Named-Entity-Recognition, Übersetzungen, das Generieren von Zusammenfassungen, und weitere Anwendungsfälle. [68, S. 25-26]. Die Nutzung von vortrainierten Modellen wird als Transfer-Learning bezeichnet. Transformer können dies ermöglichen. Gewisse Gewichtungen innerhalb von neuronalen Netzen können dabei übernommen werden und müssen nicht neu trainiert werden (z. B. grundlegende Sprach-Eigenschaften). Vortrainierte Modelle sind eine Option, wenn nur wenige Daten für das Training zur Verfügung stehen. Die vor-trainierten Modell erfahren eine Fein-Abstimmung (Engl.: Fine-Tuning), wodurch die Modelle an den jeweiligen Anwendungsfall angepasst werden [2, S. 45-56]. Ein konkreter Transformer-Ansatz ist BERT (Bidirectional Encoder Representations from Transformers). Es gibt eine Pre-Training-Phase und eine Fine-Tuning-Phase, durch die das finale Modell erzeugt wird. Es wurden Ergebnisse erreicht, die als state-of-the-art bezeichnet werden können [12]. GPT (Generative Pre-trained Transformer) und seine Nachfolger GPT-2 und GPT-3 sind weitere Beispiele für Transformer-Modelle, die insbesondere für die Text-Erzeugung eingesetzt werden können [8]

2.3 Das Application-Portfolio-Management als Teil der IT-Governance

IT-Governance soll sicherstellen, dass Informationstechnologie (IT) die Strategien und Ziele von Unternehmen unterstützt. Die IT soll so angepasst und transformiert werden, dass unterschiedliche Anforderungen erfüllt werden. Dazu zählen insbesondere regulatorische und gesetzliche Anforderungen [53, S. 5]. Gemeint sind damit etwa Gesetze und Rechtsverordnungen, wie die Datenschutzgrundverordnung (DSGVO), der Sarbanes-Oxley-Act (SOA), oder auch die bankaufsichtlichen Anforderungen an die IT (BAIT) im Bankenumfeld [18, S. 274-275].

Außerdem kann davon ausgegangen werden, dass Manager stets auch die eigenen Interessen im Sinn haben, wenn die Notwendigkeit von IT-Governance begründet werden soll. Risiken, die mit dem Betrieb von IT-Systemen einhergehen, sollten dem Management bekannt sein, damit entsprechende Maßnahmen getroffen werden können. Außerdem ist von Interesse, welchen Mehrwert einzelne Systeme tatsächlich bringen und welche Geschäftsbereiche durch die Systeme unterstützt werden. Der große Vorteil von IT-Governance ist erhöhte Transparenz. Die Transparenz macht Kosten sichtbar und ermöglicht sinnvolle Entscheidungen im Zusammenhang mit der Unternehmens-IT [27, S. 5-7]. Mit den angeprochenen Risiken können (u. a.) wirtschaftliche Verluste im Falle von Systemausfällen verbunden sein, aber auch der langfristige Verlust der eigenen Wettbewerbsfähigkeit, wenn zu lange mit der Erneuerung von Systemen gewartet wird [27, S. 20]. IT-Governance ist ein Bestandteil eines übergeordneten Corporate-Governance-Ansatzes, bei dem es um die Steuerung und Überwachung eines gesamten Unternehmens geht [18, S. 2].

2.3.1 COBIT als mögliche Grundlage für das APM?

Das Rahmenwerk (Engl.: Framework) COBIT kann als methodische Grundlage für IT-Governance dienen. Die Abkürzung steht für Control Objectives for Information and Related Technology. Die aktuelle Version (während der Bearbeitungszeit dieser Arbeit) ist COBIT 2019 [18, S. 9-12].

COBIT definiert in seinem Kern-Modell (vgl. Abbildung 2.5) sogenannte Governance- und Management-Ziele [18, S. 67]. Ziel APO05 heißt Managed Portfolio. Anhand dieses Beispiels soll verdeutlicht werden, dass der Umgang mit bestimmten Portfolios ein wesentlicher Bestandteil von Governance-Tätigkeiten sein kann. Die Bezeichnung Application Portfolio wird hier allerdings nicht verwendet, es geht stattdessen um Projekte und Programme. Dabei müssen einzelne Projekte oder Programme priorisiert werden, damit strategische Ziele erreicht werden können [31, S. 87-92].

Statt einem eigenen Management-Ziel zum Umgang mit Applikationen existiert in COBIT die Empfehlung zu Managed Assets (“Betriebsmitteln”, vgl. Ziel BAI09) [31, S. 209-214], wozu auch der Umgang mit Software-Lizenzen zählt. Unter dem Ziel Managed Configuration (BAI10) wird empfohlen, Daten zu “wichtigen Ressourcen” zu erheben und zu verwalten [18, S. 73]. Unter Ziel APO03 wird das Managen der Unternehmensarchitektur empfohlen, was die Einrichtung einer Referenzarchitektur beinhaltet, in der u. a. auch

2.3 Das Application-Portfolio-Management als Teil der IT-Governance

Anwendungen und Technologien berücksichtigt werden können [18, S. 70]. In welchen Variationen diese Empfehlungen in Unternehmen ganz konkret umgesetzt werden, ist diesen stets selbst überlassen [18, S. 65].

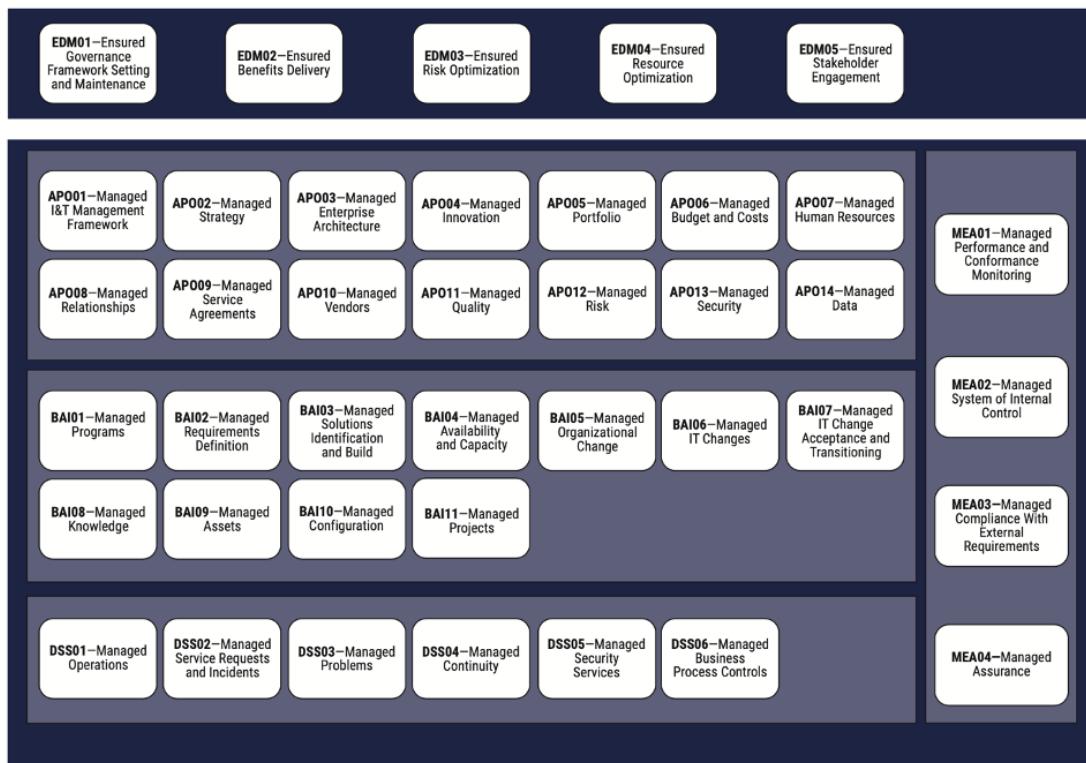


Abbildung 2.5: COBIT-Kernmodell, aus [30, S. 21]

Neben COBIT gibt es weitere Frameworks, bzw. Modelle zum Umgang mit einzelnen Governance-Themen. In [18, S. 127-133] werden beispielsweise das CMMI - Capability Maturity Model (Integration) sowie die IT Infrastructure Library (ITIL) erwähnt.

Außerdem gibt es spezielle (Enterprise-)Architektur-Frameworks. Auf einige davon wird in einem späteren Abschnitt kurz eingegangen.

2.3.2 Die Rolle von Textdokumenten für die IT-Governance

Verschiedene IT-Projekte eines Unternehmens und damit verbunden auch Softwareentwicklungsprojekte stehen grundsätzlich in Konkurrenz zueinander. Mehrere Stakeholder möchten, dass ihr eigenes Projekt mit Priorität behandelt wird und erfolgreich umgesetzt

wird. Dazu kommt, dass es Abhängigkeiten zwischen den einzelnen Vorhaben geben kann (Der Abschluss von Projekt A ist eine Voraussetzung für die erfolgreiche Umsetzung von Projekt B). Aus diesem Grund ist es wichtig, dass das IT-Management die Gesamtheit der Projekte im Blick hat, und sich weniger nur auf einzelne Projekte fokussiert [16, S. 1]. Ein Portfolio muss also verwaltet und gesteuert werden.

Neben dem voraussichtlichen Return on Investment (ROI) der sich durch die Umsetzung eines Projekts ergibt, müssen weitere Faktoren vor der Umsetzung von Projekten geschätzt werden. Dazu zählt das Risiko, dass mit einem Projekt einher geht [17, S. 120ff.]. Zunächst können für diese Schätzung die bereits erstellten Dokumentationen in Betracht gezogen werden. Der Begriff der Dokumentation bezeichnet in der Regel nachträglich erstellte Beschreibungen von Projekten. Unter dem Begriff können allerdings auch die zu einem noch laufenden oder geplanten Projekt gehörenden Dokumente verstanden werden. Dokumentationen erfüllen neben der Aufzeichnung von Ideen in Software-Entwicklungsprojekten einen weiteren wichtigen Zweck: Sie gelten als wesentliches Werkzeug zur Umsetzung von Governance und zur Sicherstellung von Compliance [53, S. 2]. Diesen Zweck erfüllen sie, wenn Außenstehende die bereits abgeschlossenen oder geplanten Projekte oder Anwendungen bewerten müssen, um letztendlich Entscheidungen zum Umgang mit diesen treffen zu können. Deshalb wird in dieser Arbeit angenommen, dass Dokumentationen in Text-Form von hoher Bedeutung für die Gestaltung der Projekt- und Applikationslandschaft von Unternehmen sind. Die Güte von Dokumentationen ist dabei von entscheidender Bedeutung. Anforderungsgerechte Dokumente werden im Idealfall durch ein Dokumentationsmanagementsystem gewährleistet. Dabei können verschiedene Vorgaben an die Struktur der Dokumente gemacht werden [53, S. 29]. Dazu zählt beispielsweise eine bestimmte Kapitelstruktur. Diese kann von einer Einteilung der Dokumente zu bestimmten Dokumententypen abhängig gemacht werden [53, S. 34-39].

2.3.3 Das APM und verwandte Begriffe

Es gibt im Zusammenhang mit der Idee des Application Portfolio Management in der Literatur diverse ähnliche Begriffe, die im Kern das beschreiben, was in dieser Arbeit unter APM verstanden wird. Ein solcher Begriff aus einer frühen Veröffentlichung zu APM ist etwa das IT Portfolio Management. Hier wird von stetigen “Kaufs-, Verkaufs- und Beibehaltungs-Entscheidungen“ gesprochen, um sogenannte IT-Portfolios zu gestalten

2.3 Das Application-Portfolio-Management als Teil der IT-Governance

bzw. um sie stets zu optimieren. APM bildet laut den Autoren gemeinsam mit dem Project Portfolio Management sowie dem IT Asset und Infrastructure Management die Disziplin des IT Portfolio Managements [21, S. 2-14].

Das oben bereits erwähnte Enterprise Architecture Management (EAM) verfolgt einen umfangreicheren, ganzheitlichen Ansatz zur Darstellung der gesamten Systemlandschaft von Unternehmen. Es werden sowohl die Prozess- als auch die Systemwelt strukturiert betrachtet. Dabei geht es stets um die Schaffung von Transparenz. Durch eine solche Transparenz können etwa unnötig redundante Systeme erkannt bzw. vermieden werden [7, S. 7-10]. Mit unnötig redundanten Systemen sind solche Systeme gemeint, welche Geschäftsfunktionen abdecken, die bereits durch andere Systeme oder Anwendungen unterstützt werden. Eine Business Support Matrix kann solche Beziehungen zwischen Anwendungen und Geschäftsfunktionen (auch: Capabilities) visualisieren [33]. Der Umfang von EAM wird dargestellt in Abbildung 2.6. Die Abbildung soll verdeutlichen, dass der Fokus nicht wie im APM nur auf Applikationen (hier repräsentiert am ehesten durch den Kasten Anwendungsarchitektur) liegt, sondern dass sich EAM mit weiteren Ebenen beschäftigt (etwa mit der Geschäftsarchitektur oder auch mit der Strategie).

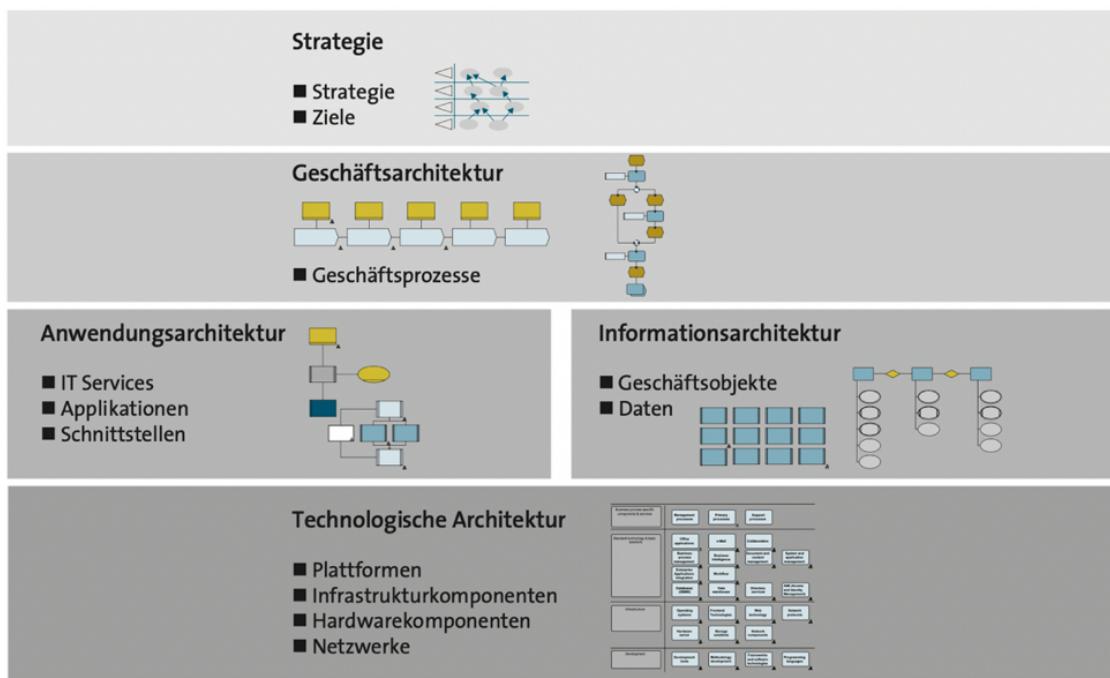


Abbildung 2.6: EA-Überblick, aus [7, S. 13]

Der enge Bezug von APM und EAM zeigt sich außerdem in Rahmenwerken, wie dem Architekturframework der Open Group (TOGAF). Dort zählt z. B. der Application Portfolio

Catalog zu den grundlegenden Artefakten, die bei der Gestaltung einer Architektur nach TOGAF zu berücksichtigen sind. Schnittstellen zwischen Applikationen werden in dem Framework in einem Interface Catalog dokumentiert. Damit sind u. a. Abhängigkeiten visualisierbar und zentrale Anwendungen identifizierbar [67, Kap. 31]. Im Department of Defense Architecture Framework (DoDAF) gibt es verschiedene sogenannte Viewpoints, die wiederum aus mehreren Views bestehen können. Eine Viewpoint in diesem Framework ist die Systems-Viewpoint. Eine konkrete View ist z. B. die “System Interface Description“ (SV-1). In ihr werden alle Systeme, die von Interesse sind, inklusive Schnittstellen zu anderen Systemen dargestellt. Die View ist vergleichbar mit dem TOGAF Interface Catalog. Es kann in anderen Views auch z. B. um die Unterstützung “operationaler Aktivitäten“ durch Software-/ oder System-Funktionen gehen (SV-5). Ein Beispiel wäre die Dokumentation darüber, dass der operationale Prozess “Einchecken von Passagieren“, durch ein System mit der Bezeichnung Airline Reservation and Ticketing System abgewickelt wird [50, S. 311-344].

Darüber hinaus zeigt eine Befragung von Enterprise-Architekten, dass diese sich in ihrer täglichen Arbeit mit Fragen zum Umgang mit Anwendungslandschaften beschäftigen [32]. Dazu zählen zum Beispiel Fragen wie

- Welche Anwendungen sind in Verwendung?
- Welchen Zweck erfüllen einzelne Anwendungen?
- Welche Anwendungen sollen eingeführt oder abgeschaltet werden?
- Welche Redundanzen in der Anwendungslandschaft gibt es?
- Welche Abhängigkeiten oder Schnittstellen gibt es zwischen Anwendungen?
- In welchen Anwendungen werden kritische oder schützenswerte Daten gehalten?

Als Kernelement von IT-Strategien wird der IT-Bebauungsplan in einer Veröffentlichung zum Thema IT-Controlling bezeichnet. Er gibt Aufschluss über die Informationssysteme, die derzeit in einem Unternehmen im Einsatz sind. Dazu zählen (u. a.) Informationen zu Release-Ständen, zu Schnittstellen der Systeme untereinander, sowie zu Geschäftsprozessen, die von den Systemen unterstützt werden. [17, S. 36] Die Autoren dieses Beitrags nennen das IT-Portfoliomangement als Werkzeug für IT-Controller (die dem Chief Information Officer gleichgestellt oder dieser Position direkt unterstellt sein können). Zu deren Aufgaben können die Auswahl und Steuerung von neuen Projekten oder von Wartungsprojekten gehören. [17, S. 41-45]

2.3.4 Handlungsfelder des APM

Das grundlegende Ziel von APM ist stets die Reduzierung der Komplexität von Anwendungslandschaften. Anwendungen werden dabei aus technischer (etwa Aktualität der Software) und betriebswirtschaftlicher Sicht (Betriebs- oder Lizenzkosten) untersucht. Es handelt sich um einen kontinuierlichen, andauernden Vorgang [58]. Nachfolgend aufgelistet sind Beispiele für Fragestellungen, mit denen sich das APM befasst. Sie werden im Folgenden als Handlungsfelder des Application Portfolio Management zusammengefasst. Die Fragen wurden abgeleitet aus [4]. Hervorgehoben sind Fragen, auf die im weiteren Verlauf der Arbeit noch stärker eingegangen werden soll. Es gibt auch jeweils einen Verweis auf das jeweilige Kapitel, in denen entsprechende Verfahren zum Umgang mit den Fragen diskutiert werden.

- Werden bei der Implementierung von neuer Software Legacy-Systeme als Grundlage verwendet?
- Werden die Anforderungen aus dem operativen Geschäft durch neue Anwendungen abgedeckt?
- Sind Anwendungen aus technischer und fachlicher Sicht “fit“ für die Zukunft?
- Werden durch Anwendungen bestimmte rechtliche Anforderungen umgesetzt?
- **Was ist für einzelne Anwendungen “in-scope“ und was ist “out-of-scope“? (vgl. Abschnitt 4.3)**
- Gibt es in den Dokumentationen der geplanten Softwareprojekte einen Projektplan oder ein Gantt-Diagramm?
- **Wurde schon einmal versucht, ein bestimmtes Vorhaben umzusetzen und wie war das Resultat? (vgl. Abschnitt 4.1)**
- Was sind die fachlichen Anforderungen, die durch ein bestimmtes Projektvorhaben abgedeckt werden sollen?
- Wie verhalten sich die Kosten, die durch das Ersetzen einer Software anfallen im Vergleich zu den Kosten die bei einer Modernisierung/ Weiterentwicklung entstehen?
- Ist es möglich, einzelne Anwendung zu verbessern/ weiterzuentwickeln?
- Was spricht für eine Wartung einer bestehenden Software, was für eine Neuentwicklung?
- Welche Anwendungen sind Kandidaten für eine Restrukturierung/ Optimierung
- **Welche Abhängigkeiten von einzelnen Anwendungen gibt es zu anderen Anwendungen im Application Portfolio? (vgl. Abschnitt 4.4)**

- Werden bei Neueinführung von Software Abhängigkeiten geschaffen?
- Welcher Aufwand für die Einführung einer Anwendung ist realistisch?
- Welche Auswirkungen hat eine Nicht-Verfügbarkeit von einzelnen Anwendungen?
- In welcher Phase im Lebenszyklus befinden sich bestimmte Anwendungen?
- Sind Performance-Verbesserungen durch die Einführung von neuer Software identifizierbar?
- Werden bestehende Prozesse angemessen durch neue/ bestehende Anwendungen unterstützt?
- Mit welchen Risiken wird die Organisation konfrontiert durch den Betrieb der Anwendungen und wie können die Risiken minimiert werden?
- Was sind Dinge, die schief gehen könnten bei Veränderungen am Application Portfolio
- Ist ein Produkt auf Basis der eigenen Bedürfnisse neu zu entwickeln oder kann ein Standardprodukt (z. B. für die Zeiterfassung) gewählt werden?
- In welchen Fällen muss über den Abbruch eines laufenden Projekts nachgedacht werden?
- Geht es bei Projekten eher um das “Bekämpfen von Bränden“ (Engl.: “Firefighting“) oder um echte Verbesserungen der Geschäftsprozesse?
- **Wie gut ist die Dokumentation einer Anwendung? (vgl. Abschnitt 4.2)**
- Passen Implementierungsplanungen und strategische Planungen des Unternehmens zusammen?
- Wie stark werden Standards (z. B. Standard-Entwicklungsmethoden) eingehalten?
- Wie ist die Aktualität der Anwendungen im Portfolio – welche Anwendungen sind veraltet?
- Wie lange werden Anwendungen vom Vendor noch unterstützt im Hinblick auf “Support“?
- Wie ist die Kritikalität von Anwendungen? (mission-critical, critical, high, medium, low).
- Wer sind die Verantwortlichen (Engl.: “Owner“) der Anwendungen?
- Ist die Funktionalität einer Anwendung einzigartig und nicht redundant?
- Was sind “Lessons Learned“ die sich ergeben haben durch Betrieb von Software?
- Wie einfach ist eine Anwendung nutzbar?
- Welche sind die am häufigsten/ seltensten genutzten Anwendungen im Portfolio
- Wie viele Nutzer hat eine Anwendung?
- Ist die Anwendungslandschaft mit jener von Wettbewerbern vergleichbar?

Ein Großteil dieser Fragestellung setzt zur Beantwortung Expertenwissen voraus. Als Beispiel wird der Begriff der strategischen Planung herausgegriffen. Ein solcher Begriff wäre in der Realität zunächst durch Experten zu definieren.

2.3.5 Unterstützung von APM durch Tools

Im vorangegangen Abschnitt wurden einzelne Fragen genannt. Bei der Bearbeitung der Fragen können Experten durch Technologie unterstützt werden. Zum einen gibt es dazu die bereits erwähnten Frameworks wie COBIT, TOGAF, und weitere – zum anderen gibt es auch spezialisierte Software-Produkte, die an dieser Stelle unterstützen können, wie:

- ServiceNow Application Portfolio Management. Die Use Cases umfassen u. a. das Darstellen der gesamten Applikationslandschaft, die Verlinkung von Anwendungen mit Projekten, sowie das Zuordnen von Anwendungen zu Lebenszyklus-Phasen [56].
- LeanIX Enterprise Architecture Suite. Die Unterstützung des APM ist einer der grundlegenden Anwendungsfälle des Produkts. Mit dem Werkzeug können (u. a.) Anwendungen mit Business Capabilities verknüpft, Lebenszyklus-Phasen erfasst, und Anwendungslandschaften visualisiert werden [65].
- Alfabet (Software AG). Sowohl Anwendungsfälle im Bereich EAM als auch im sogenannten Integrated Portfolio Management (ITPM) werden durch die Software abgedeckt. Dabei sind bestimmte Features auch explizit für das APM von Relevanz, wie das Zusammenstellen eines Anwendungs-Inventars, das Zuordnen von Anwendungen zu Lebenszyklus-Phasen, sowie das Hinzufügen von Attributen zu Anwendungen, wie beispielsweise Kosten, Risiken, Nutzungsgrad, und Performance [46].

All diese Tools sind laut den Herstellern geeignet, um das APM grundlegend zu unterstützen, wobei Inhalte manuell erfasst werden müssen. Die in Word-Dokumenten verborgenen Informationen werten sie jedoch nicht aus. Sie wenden darüber hinaus keine oder kaum Möglichkeiten aus dem Bereich des maschinellen Lernens auf die eigentlichen Inhalte an. Sie nutzen damit das bestehende Potenzial dieser Ansätze noch nicht. Hier sollen die Verfahren ansetzen, die in den nachfolgenden Abschnitten untersucht werden.

3 Relevante Verfahren und Kriterien für die Evaluation

Die Evaluation der Einsatzmöglichkeiten von ML-Verfahren für das Application Portfolio Management erfolgt in einem festgelegten Kontext. Zunächst soll eine reale Ist-Situation skizziert werden, die während der Bearbeitungszeit vorgefunden wurde. Auf der Ist-Situation soll die eigentliche Evaluation von Verbesserungsansätzen aufbauen.

Die ursprüngliche Aufgabenstellung stammte von einem multinationalen Großunternehmen. Wie im Theorienteil beschrieben wurde, werden im Laufe von Softwareentwicklungsprojekten verschiedene Dokumente erstellt (beispielsweise “System Requirements Specifications” (SRS), “Business Requirement Specifications” (BRS), “Solution Architecture Statements” (SAS), “Detail(ed) Design Specifications” (DDS)). Diese Dokumente werden normalerweise auf Basis von Standard-Templates angefertigt. Die Dokumente müssen im vorliegenden Szenario manuell ausgewertet werden, um Informationen aus den Dokumenten zu extrahieren oder um Dokumenten-übergreifende Erkenntnisse zu gewinnen. Es wird die These vertreten, dass durch KI- und ML-Ansätze diese manuellen Prozesse unterstützt werden können. Nachfolgend werden zunächst Anwendungsfälle beschrieben und es wird erläutert, mit welchen ML-Verfahren die Anwendungsfälle realisiert werden können. Das Kapitel erläutert außerdem wie und anhand von welchen Kriterien die einzelnen Verfahren evaluiert werden.

3.1 Verfahren zur Ermittlung von Ähnlichkeiten

Es wäre für das Application-Portfolio-Management sehr hilfreich, wenn anhand von Text-Dokumenten auf in der Vergangenheit durchgeföhrte, ähnliche Projekte geschlossen werden kann, um Überschneidungen von Projekten festzuhalten. Der Frage “Wurde schon einmal versucht, ein bestimmtes Vorhaben umzusetzen“ könnte sich über ein Ähnlichkeitsmaß angenähert werden. Ob das möglich und sinnvoll mit Methoden aus dem Bereich des maschinellen Lernens ist, wird gezeigt in Kapitel 4.1. Es werden zwei Ansätze untersucht.

3.2 Verfahren zur Bestimmung von inhaltlicher Integrität

Die Ermittlung von Ähnlichkeiten ergibt sich durch den Abgleich eines Dokuments mit anderen Dokumenten einer bestehenden Dokumenten-Datenbank. Genutzt werden könnten dazu zum einen regelbasierte Ansätze, aber auch unterschiedliche Vektor-basierte Ansätze, wie sie in Kapitel 2 beschrieben wurden. Damit kann ein Ähnlichkeitsmaß zwischen einzelnen Dokumenten erzeugt werden. Das Ziel ist hier also die Ermittlung der Überschneidung von Projekten, basierend auf dem Inhalt von Dokumenten. So könnten zum Beispiel zwei Projekte identifiziert werden, in denen es um die Entwicklung von Anwendungen mit den gleichen Absichten geht. Vektor-basierte Ansätze, wie sie in Kapitel 2 beschrieben wurden, nutzen Verfahren des maschinellen Lernens und sind daher hier für eine Evaluation von Bedeutung. Regelbasierte Ansätze können in der Praxis sinnvoll sein, sie werden aufgrund des Titels der Arbeit aber nicht näher betrachtet. Weitere Möglichkeiten könnten Nearest-Neighbor-Verfahren bieten, sie wurden jedoch nicht näher evaluiert und werden nur als mögliche Perspektive erwähnt.

3.2 Verfahren zur Bestimmung von inhaltlicher Integrität

Text-Klassifizierungsverfahren (auch: Verfahren zur Topic-Classification) verfolgen das Ziel, einem gegebenen Text eine oder mehrere Klassen zuzuordnen. Ein bekanntes Beispiel ist die Klassifizierung von Emails in die beiden Klassen Spam oder kein Spam. [68] Es wird bei den vorliegenden Dokumenten davon ausgegangen, dass sie viel Text, aber nur wenige echte Informationen enthalten. Es werden Ansätze untersucht, mit denen überprüft werden soll, ob Dokumente das beinhalten, was erwartet wird. Ein Kapitel über “Anforderungen“ sollte idealerweise auch tatsächlich Anforderungen beschreiben. Dazu kann ein Dokumenten-Template definiert werden, mit erwarteten “Topics“. Neben einer Klasse “Requirements“ könnte eine weitere Klasse “Architectural Language“ heißen. Die Spalte “Gefundenes Thema“ aus Tabelle 3.1 ist bei diesem Ansatz das durch einen Klassifizierungsansatz zu schätzende Attribut. Voraussetzung für diese Schätzung ist lediglich, dass die Kapitelstruktur der Dokumente sich an die Templates hält. Davon wird in dieser Arbeit ausgegangen. In der Realität kann es durchaus Abweichungen zwischen Template-Kapitelstruktur und Kapitelstruktur der Dokumente geben. Dieses Problem wird hier bewusst ausgeklammert. Nicht gezeigt werden in der Tabelle die eigentlichen Texte, die als Grundlage für die Schätzung verwendet werden.

3.3 Verfahren zur automatisierten Erzeugung von Zusammenfassungen

Kapitel	Kapitelüberschrift	Erwartetes Thema	Gefundenes Thema
1	Introduction	No special expectation	?
1.1	Background	No special expectation	?
1.2	Purpose of the Document	No special expectation	?
2	System Requirements	Requirements	?
2.1	User Interface and Functional Requirements	Requirements	?
2.2	Technical Requirements	Requirements	?
3	System Scope	Architectural Language	?
3.1	Context	Architectural Language	?
3.2	System Interfaces	Architectural Language	?
3.3	System Impacts	Architectural Language	?
4	Business Process and Data Definition	Architectural Language	?
4.1	Business Process Maps	Architectural Language	?
4.2	High-level Data Model	Architectural Language	?

Tabelle 3.1: Ausgangssituation bei der Themenerkennung

Es geht hier letztendlich darum, die inhaltliche Integrität von Dokumenten zu beurteilen, in denen es um die Spezifikationen von Applikationen geht. Dadurch kann die Frage nach der Güte der Dokumentation einer Anwendung bzw. eines Projekts beantwortet werden. Eine Dokumentation mit Inhalten, die besonders stark von ähnlichen historischen Dokumentationen abweicht, könnte identifiziert werden. In Abschnitt 4.2 werden zwei Ansätze zur Schätzung der inhaltlichen Integrität von Texten untersucht.

3.3 Verfahren zur automatisierten Erzeugung von Zusammenfassungen

Es wird hier die These vertreten, dass es ab einer bestimmten Menge von Dokumenten nicht mehr praktikabel ist, manuell alle vorhandenen Dokumentationen zu lesen und zu versuchen, eine Einordnung der Dokumente und damit der Anwendungen vorzunehmen. Eine automatisch generierte Zusammenfassung der Dokumente könnte sinnvoll sein, damit ein verantwortliches Team die Dokumente schneller und effizienter zu- bzw. einordnen kann.

3.3 Verfahren zur automatisierten Erzeugung von Zusammenfassungen

Bei der Erzeugung von Text-Zusammenfassungen kann grundsätzlich unterschieden werden zwischen extraktiven und abstraktiven Algorithmen. Extraktive Algorithmen zur Erzeugung von Textzusammenfassungen entnehmen wichtige Sätze aus Texten, ohne dabei einzelne Wörter oder Formulierungen zu modifizieren [1, S. 2]. Derartige Ansätze gibt es mindestens seit dem Jahr 1958, als ursprünglich versucht wurde Literaturabstrakte automatisiert zu erzeugen [38]. Von den extractiven Verfahren soll in dieser Arbeit ein Ansatz auf Basis von TF-IDF näher untersucht werden. Nicht untersucht, aber für die Praxis relevant könnte ein Ontologie-basierter Ansatz sein, der die Wichtigkeit von Sätzen berechnet auf Basis der Anzahl bestimmter Signalwörter aus der entsprechenden Ontologie. Signalwörter können dabei zum Beispiel angeben, ob es sich um interessante Sätze, wie Anforderungsbeschreibungen, handelt. Weitere Algorithmen aus der Kategorie der extractiven Verfahren sind der WordFrequency- sowie der TextRank-Algorithmus. Sie werden nicht näher betrachtet, da davon ausgegangen wird, dass die Güte dieser Verfahren in etwa der des TF-IDF-Ansatzes entspricht. Abstraktive Verfahren hingegen generieren Zusammenfassungen, die neu zusammengestellte Satzkonstellationen enthalten können, die es so in den ursprünglichen Texten gar nicht gegeben hat [68, S. 258]. Zu den abstraktiven Verfahren die untersucht werden sollen zählt ein vortrainiertes Modell mit dem Namen T5: Text-To-Text Transfer Transformer. Die Neuartigkeit und die Geschwindigkeit der Weiterentwicklung dieser Ansätze erlaubt jedoch hier keine ausführliche Untersuchung aller möglichen Verfahren. Indem je ein extractives und ein abstraktives Verfahren untersucht wird, kann ein grober Vergleich der beiden Ansätze ermöglicht werden.

Eine Metrik bzw. ein Ansatz zur Bewertung der Qualität von Zusammenfassungen ist ROUGE: A Package for Automatic Evaluation of Summaries, wobei die Abkürzung für Recall Oriented Understudy for Gisting Evaluation steht. ROUGE ist gedacht für den Abgleich von automatisch generierten und von Menschen erstellten Zusammenfassungen. Es gibt von diesem Package verschiedene Implementierungen. Für die Evaluation der Verfahren kann insbesondere Rouge-N in Betracht gezogen werden. Rouge-N erzeugt N-gram Co-Occurrence Statistics, beantwortet also etwa die Frage “wie viele Bi-Gramme aus einer von Menschen erzeugten Zusammenfassung in einer automatisch erzeugten Zusammenfassung vorkommen“ [37]. Das mögliche Vorgehen sieht folgendermaßen aus:

1. Automatisierte Erzeugung von Zusammenfassungen mit einem der beiden Ansätze
2. Erzeugung einer menschlichen Referenz-Zusammenfassung mit entsprechender Länge
3. Anwendung von Rouge zur Ermittlung der Rouge-Metriken

3.3 Verfahren zur automatisierten Erzeugung von Zusammenfassungen

Die Güte von Zusammenfassungen wird in dieser Arbeit also mit einer Metrik beurteilt. ROUGE gleicht zwar den Inhalt der Zusammenfassungen ab, kann aber nicht die “Lesbarkeit“ bewerten, oder ob ein Text aus “grammatikalischer Sicht sinnvoll ist“. Daher ist die Einschätzung eines Menschen eigentlich immer am aussagekräftigsten. Eine solche Einschätzung ist jedoch nur dann nützlich, wenn die Verfahren überhaupt sinnvolle Ergebnisse produzieren. Wenn in der Realität ausreichend gute Zusammenfassungen mit einem Verfahren erzeugt werden könnten, dann wäre eine Befragung von Projektbeteiligten denkbar, um die Verfahren zu evaluieren. Die Auswertung der Experten-Befragung kann vereinfacht werden durch Verwendung einer standardisierten 5-Punkte LIKERT-Skala (vgl. das Beispiel in Abbildung 3.1). Eine solche Befragung wurde hier nicht durchgeführt, stattdessen wird ein Vorschlag für eine zukünftige mögliche Befragung gemacht. Die linguistischen Eigenschaften, die untersucht werden, orientieren sich an [10]. Sie sind in Abbildung 3.1 in Spalte 1 abgebildet. Es geht um Grammatik, Redundanz-Vermeidung, korrekte Referenzen, sowie darum ob der Fokus auf dem Wesentlichen liegt, und ob Struktur und Zusammenhänge zwischen Sätzen sinnvoll sind. Die Skala reicht von “sehr unzufrieden“ bis “sehr zufrieden“.

Question	Very Unsatisfied	Unsatisfied	Neutral	Satisfied	Very Satisfied
Grammaticality	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Non-Redundancy	<input type="radio"/>				
Referential Clarity	<input type="radio"/>				
Focus	<input type="radio"/>				
Structure and coherence	<input type="radio"/>				

Abbildung 3.1: Potenziell mögliche Befragung, eigener Entwurf in Anlehnung an [10]

In Abschnitt 4.3 werden zwei Verfahren zur Erzeugung von Zusammenfassungen evaluiert.

3.4 Verfahren zur automatisierten Schnittstellen-Erkennung

Dieser Anwendungsfall bezieht sich auf die Erkennung von Schnittstellen (Engl.: Interfaces). Gesucht sind technische Schnittstellen, die Abhangigkeiten zwischen einzelnen Anwendungen darstellen. Das Problem bei der Erkennung einer Schnittstelle anhand von Text in Englischer Sprache ist, dass das Vorkommen des Worts Interface alleine noch nicht unbedingt bedeutet, dass eine technische Schnittstelle identifiziert wurde. Es muss dabei mindestens unterschieden werden zwischen Technischen Schnittstellen (Technical Interfaces, Abk.: TI) und Benutzerschnittstellen (User Interfaces, Abk.: UI). Wenn Schnittstellen automatisch erkannt werden, dann konnen Sie anschlieend visualisiert, etwa wie im Beispiel in Abbildung 3.2 werden und entsprechend bei Entscheidungen im APM berucksichtigt werden. Das ist der groe Mehrwert dieses Ansatzes. Zwei verschiedene Verfahren zur Losung dieses Anwendungsfalls werden evaluiert in Kapitel 4.4.



Abbildung 3.2: Schnittstellen-Visualisierung (Interface Circle Map) aus LeanIX, aus [29]

Alternativ zu Schnittstellen könnten mit derartigen Verfahren auch beispielsweise Anforderungen als solche klassifiziert werden, d.h. einzelne Sätze würden der Klasse Anforderungen zugeordnet. Sinnvoll im Kontext der IT-Governance ist das Erkennen einer Anforderung innerhalb eines Dokuments dann, wenn es z. B. einen dokumentierten Test für jede Anforderung geben muss. Dann könnte diese Überprüfung auf den durch ML

3.5 Zu untersuchende Kriterien

erkannten Anforderungen aufbauen [70]. Hier könnte z. B. zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden werden.

Statt den Klassen zur Schnittstellenerkennung könnten also auch andere Klassen verwendet werden. In Kapitel 4.4 erfolgt jedoch eine Begrenzung auf die Schnittstellen-Erkennung.

3.5 Zu untersuchende Kriterien

Einzelne Kennzahlen zur Bewertung von Ergebnissen sind wichtig für die Kommunikation von Ergebnissen an Projektbeteiligte. Hier werden nun Kriterien aufgestellt, anhand derer die Evaluation in dieser Arbeit durchgeführt werden soll. Die Kriterien sind:

- Kriterium 1: Einsetzbarkeit der Methode unter den gegebenen Randbedingungen. Für die Evaluation von Verfahren standen bis zu 40 Text-Dokumente (im .docx-Format) zur Verfügung. Es soll überprüft werden, ob die jeweiligen Verfahren mit dieser recht geringen Datenmenge sinnvoll nutzbar sind und einen Mehrwert für das APM erbringen können. Wo es sinnvoll ist, können Metriken wie Accuracy, Precision, Recall, und die F1-Metrik eingesetzt werden.
- Kriterium 2: Generelle Einsetzbarkeit im APM-Kontext, unter der Annahme einer beliebig großen Datenmenge. Auch hier können entsprechende Metriken eingesetzt werden.
- Kriterium 3: Vertretbarer Rechenaufwand der Methode. Untersucht werden hier z. B. die Dauer für das Training von Modellen, die Dauer zur Ausführung von Schätzungen, etc.
- Kriterium 4: Anfallende Kosten, welche zum Beispiel durch notwendige Lizenzierung von Modellen entstehen. Kosten können in der Praxis auch entstehen für das Zusammenstellen und Auszeichnen von Trainingsdaten, was durch Menschen geschehen muss. Es könnte auch sein, dass bei Einsatz gewisser Methoden oder bei Nutzung von großen Modellen eine bestimmte Hardware vorausgesetzt wird.

4 Ergebnisse der Evaluation

Hier sollen Implementierungsansätze diskutiert werden. Zusammengefasst wurden vier Anwendungsfälle identifiziert (vgl. Kapitel 3.1-3.4), für deren Implementierung in diesem Kapitel bestimmte Verfahren näher evaluiert werden. Die Anwendungsfälle sind:

1. Ermittlung von Ähnlichkeiten
2. Bestimmung von inhaltlicher Integrität
3. Automatisierte Erzeugung von Text-Zusammenfassungen
4. Automatisierte Schnittstellen-Erkennung

Es werden für jeden Anwendungsfall nachfolgend jeweils zwei verschiedene Ansätze näher untersucht. Alle Ansätze werden anhand der in Kapitel 3.5 definierten Kriterien untersucht.

4.1 Ergebnisse: Ermittlung von Ähnlichkeiten

Lösungsansatz 1 für diesen Anwendungsfall verwendet einen Ansatz auf Basis von Wort-Vektoren. Ansatz 2 basiert ebenfalls auf Wort-Vektoren, es wird jedoch der Paragraph-Vector-Ansatz untersucht. Eine Visualisierung der Vektoren im zweidimensionalen Raum in Abbildung 4.1 zeigt die Intuition hinter Wort-Vektoren. Die hochdimensionalen Wortvektoren wurden per Hauptkomponentenanalyse (Engl.: Principal Component Analysis, Abk.: PCA) auf eine zweidimensionale Darstellung reduziert. Die Qualität der Vektoren kann etwa über Analogien evaluiert werden, wie “Frau zu König ist wie Mann zu . . .” [52], wobei man sich eigene Analogien ausdenken müsste, die im jeweiligen Kontext sinnvoll sind.

Abbildung 4.1 basiert auf einem eigenen Experiment, in dem die verfügbaren Kunden-Dokumente in Kleinschreibung (etwa 330 000 Wörter; entspricht einer Größe von 2.1 Megabyte) für das Training eines eigenen Wort-Vektor-Modells verwendet wurden. Das so erzeugte Modell kennt zwar spezielle Begriffe (etwa die Namen von einzelnen Projekten) aus den Dokumenten, es hat aber z. B. keine Idee davon, wie beliebige Begriffe

4.1 Ergebnisse: Ermittlung von Ähnlichkeiten

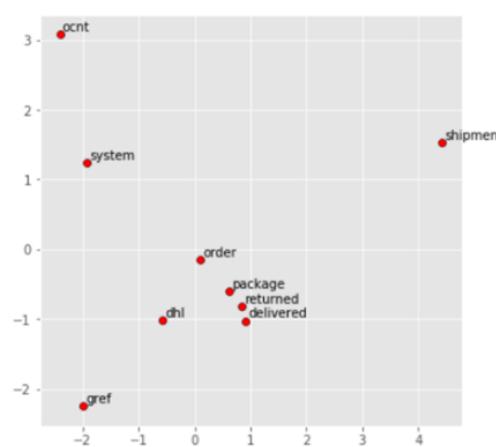


Abbildung 4.1: Darstellung verschiedener Wörter in 2 Dimensionen , eigene Erstellung.

wie “London“ oder “Paris“ einzuordnen sind. Diese Begriffe sind nicht einmal Teil des Vokabulars . Das zeigt, dass ein selbst trainiertes Modell nur bedingt für den Vergleich von Dokumenten geeignet ist. Es kann geschlussfolgert werden, dass das Training eines geeigneten Modells domänenspezifische Dokumente und zusätzlich einen “allgemeinen“ Textkorpus voraussetzen würde, damit sinnvoll mit dem Modell gearbeitet werden könnte.

4.1.1 Ansatz 1: Ähnlichkeitsvergleich mit einem Word2Vec-Modell

Im nachfolgenden, einleitenden Beispiel wurde die Ähnlichkeit zwischen ganzen Dokumenten berechnet. Aufgrund der beschriebenen Schwierigkeiten mit einem selbst trainierten Modell wird hier für die Evaluation das vortrainierte, gebrauchsfertige Wort-Vektor-Modell von spaCy verwendet. Es wurde das größte verfügbare Modell *en_core_web_lg* gewählt. Das Modell deckt von den Begriffen aus den Kundendokumenten lediglich 91,9 Prozent der Begriffe durch Vektoren ab. Das heißt, es gibt für 8,1 Prozent der Begriffe in den Kundendokumenten keine Vektoren in dem gebrauchsfertigen spaCy-Modell¹. Aus diesem Grund steht bereits an dieser Stelle fest, dass ein gebrauchsfertiges Modell für die Berechnung von Ähnlichkeiten recht viele wichtige Begriffe nicht berücksichtigen kann. Abbildung 4.2 zeigt exemplarisch, wie die Ähnlichkeit von Texten in Python auf Basis von Wort-Vektoren ermittelt werden kann. Das dargestellte Skript liefert ein Ähnlichkeitsmaß von 0.98. Es handelt sich in dem Beispiel um Dokumente, die unterschiedliche Versionen der gleichen Software beschrieben. Der gleiche Test mit doc1 aus der Abbildung und einem

¹Berechnung: Wörter die keinen spaCy-Vektoren besitzen/ Gesamtanzahl der Wörter aus den Dokumenten

4.1 Ergebnisse: Ermittlung von Ähnlichkeiten

Dokument aus einem anderen Projekt lieferte einen Score von 0.951064. Grundsätzlich sind die Ausgaben also interpretier- und nachvollziehbar. Zu beachten bei den Ergebnissen oben ist stets, dass einige domänenspezifische Wörter bei diesem verwendeten Modell keine Vektoren besitzen und deshalb gar nicht berücksichtigt werden können, wie bereits angedeutet. Eine Option zur Lösung des Problems bestünde darin, Vektoren komplett neu zu trainieren.

```
import spacy

nlp = spacy.load("en_core_web_lg")

doc1 = nlp("Business requirements for xxxv1.2 as below : No Business
Requirement in summary BR1 Offline containerization and support tools... ")

doc2 = nlp("Business requirements for xxxv1.3 as below : No Business
Requirement in summary BR1 Enhance HU Profile - Code in ... ")

print(doc1.similarity(doc2))
```

Abbildung 4.2: Similarity Berechnung mit spaCy. Die Texte wurden gekürzt.

Kriterium 1: Einsetzbarkeit des Ansatzes unter den gegebenen Rahmenbedingungen

Unten dargestellt (Abbildung 4.3) ist eine Heatmap, welche die Ähnlichkeit von Dokumenten zueinander visualisiert. Aus Datenschutzgründen werden hier nicht die Namen der Dokumente, sondern interne IDs präsentiert. In diesem Versuch wurden keine Stopworte entfernt. Das erklärt, weshalb die Skala lediglich von 1 (zu interpretieren als “Dokumente sind vollständig identisch“) bis 0,95 (zu interpretieren als “Dokumente sind sich sehr ähnlich“) verläuft. In Abbildung 4.4) ist das Ergebnis beim gleichen Vorgehen mit Stopwortentfernung dargestellt. Die Skala verändert sich, d. h. Dokumente sind sich weniger Ähnlich als vorher. Insgesamt sind aber kaum Unterschiede zu oben (Abbildung 4.3) erkennbar.

Es sind weitere Verfeinerungen denkbar, wie die Entfernung aller Verben oder Adjektive, oder die manuelle Gewichtung von bestimmten Substantiven. Ein solches Vorgehen geht in Richtung regelbasierte Sprachverarbeitung und wird daher hier nicht weiterverfolgt.

4.1 Ergebnisse: Ermittlung von Ähnlichkeiten

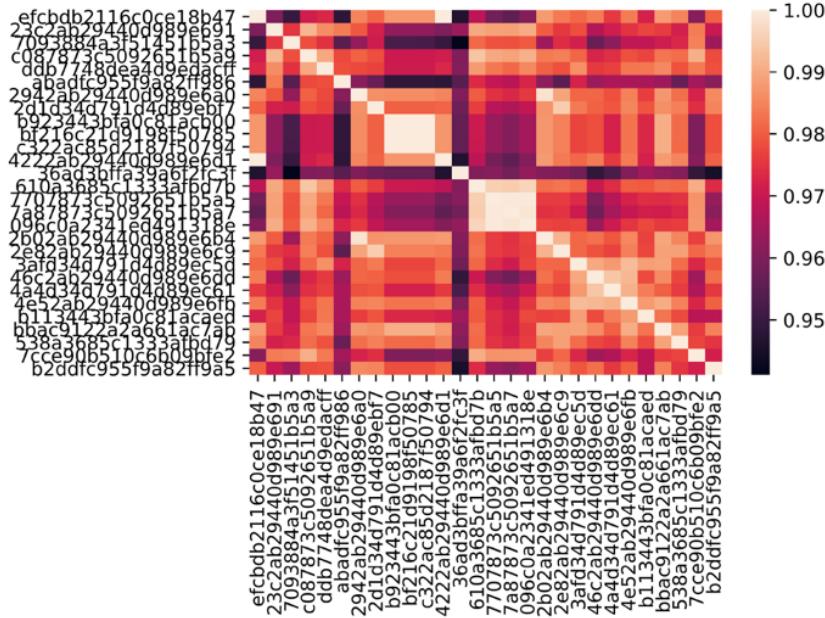


Abbildung 4.3: Heatmap zur Dokumentenähnlichkeit, eigene Erstellung

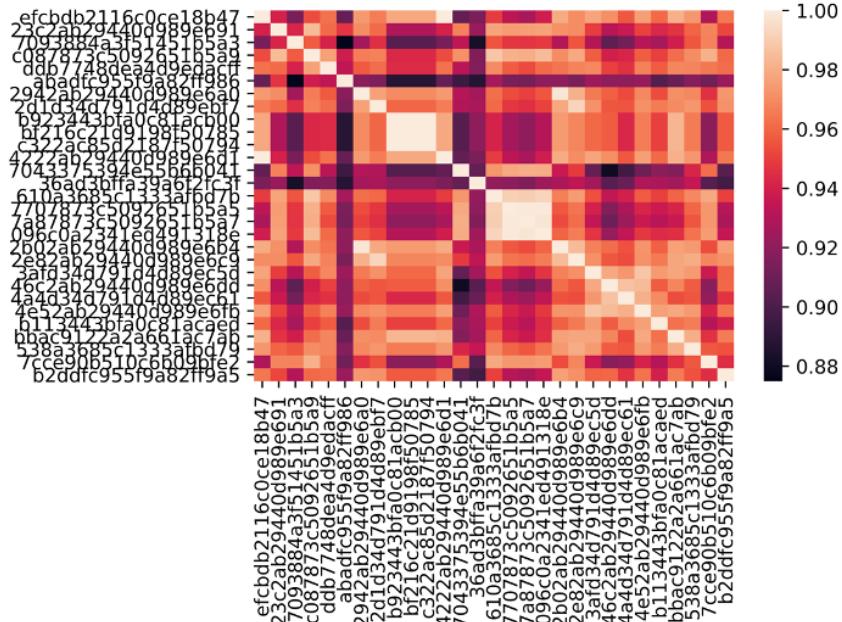


Abbildung 4.4: Heatmap zur Dokumentenähnlichkeit, mit Stopwortentfernung, eigene Erstellung

4.1 Ergebnisse: Ermittlung von Ähnlichkeiten

Der Ansatz ist also mit der gegebenen Datenmenge (20-40 Dokumente) sinnvoll einsetzbar, auch wenn durch ein Training eigener Vektoren noch Verbesserungen im Hinblick auf die Ergebnisse zu erwarten sind. Er liefert zumindest Hinweise darauf, in welchen Dokumenten bzw. Projekten mit ähnlicher Sprache gearbeitet wird. Die Metrik (etwa “Ähnlichkeit von 0,99“) kann zum Beispiel genutzt werden, um zu überprüfen, ob ein Dokument anderen existierenden Dokumenten ähnelt. Ein Entscheider könnte sich dieses ähnliche Dokument dann noch einmal genauer ansehen um zu prüfen, ob beispielsweise Systeme mit ähnlichen Anforderungen entdeckt wurden.

Kriterium 2: Generelle Einsetzbarkeit des Ansatzes im APM-Kontext, unter der Annahme einer beliebig großen Datenmenge

Für Kriterium 2 gilt das gleiche wie bei Kriterium 1. Eine größere Datenmenge würde das Training eines eigenen, auf den IT-Governance-Kontext zugeschnittenen Vektor-Modells ermöglichen, wodurch noch genauere Ergebnisse zu erwarten wären.

Kriterium 3: Rechenaufwand

Das Vergleichen von nur zwei Dokumenten mit durchschnittlich 5000 Wörtern dauert auf gewöhnlicher Hardware bereits einige Sekunden. Bei 30 Dokumenten würden sich 900 durchzuführende Vergleiche ergeben. Es ist also mit einer gewissen Dauer zu kalkulieren, wenn die Ähnlichkeit dynamisch berechnet werden sollte. Die Ähnlichkeit von einzelnen Dokumenten zueinander wird berechnet auf Basis der Kosinus-Ähnlichkeit, wobei zunächst von allen Vektoren der jeweiligen Dokumente der Durchschnitt berechnet wird [59]. Die Kosinus-Ähnlichkeit ermittelt einen Wert zwischen -1 und 1 für die Ähnlichkeit von zwei Vektoren A und B [68, S. 84]:

$$\cos(\theta) = \frac{\mathbf{A} \times \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^n (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^n (\mathbf{B}_i)^2}} \quad (4.1)$$

Der Rechenaufwand zur Erzeugung der fachspezifischen Vektoren ist abhängig von der Größe des jeweiligen Trainingskorpus.

Kriterium 4: Kosten

Es sind hier keine Kosten durch Lizensierung oder manuelle Tätigkeiten erkennbar, zumindest solange, wie mit den voreingestellten Wort-Vektoren gearbeitet wird. spaCy-Modelle verwenden die MIT-Lizenz [60], welche die kostenfreie Nutzung ermöglicht [66]. Wenn ein Textkorpus zusammengestellt werden sollte, um eigene Vektoren zu erzeugen, dann müssten entsprechende Dokumente aus unterschiedlichen Quellen gesammelt werden, was Aufwand bedeuten würde.

4.1.2 Ansatz 2: Ähnlichkeitsvergleich unter Verwendung des Paragraph-Vector-Modells (Doc2Vec)

Das Paragraph-Vector-Modell kann für die Ermittlung von Ähnlichkeiten eingesetzt werden [9]. Nachfolgend wird dieses Verfahren im Kontext des APM zur Ermittlung von Ähnlichkeiten untersucht. Die Ähnlichkeit zwischen Dokumenten kann abgesehen von einer Heatmap (wie im vorangegangenen Abschnitt) auch über Cluster in einem Streudiagramm visualisiert werden. Diese Form der Visualisierung wird hier gewählt. Dokumente die sich ähneln, würden im Streudiagramm nahe beieinander liegen.

Kriterium 1: Einsetzbarkeit des Ansatzes unter den gegebenen Rahmenbedingungen

Es wird hier ein eigenes Doc2Vec-Modell trainiert. Das Verfahren Doc2Vec von gensim erzeugt für jedes Dokument einen Vektor. Diese Vektoren können je nach Parameterwahl verschiedene Dimensionen haben. Um die Vektoren in einer zweidimensionalen Ebene darstellen zu können, wurde t-SNE (t-Distributed Stochastic Neighbor Embedding, nach [39]) zur Dimensionsreduktion verwendet. Es wurde davon ausgegangen, dass ähnliche Dokumente auch ähnliche Vektoren haben. Dadurch sollten in Abbildung 4.5 Cluster erkennbar sein – wie nachfolgend unter Kriterium 2 beschrieben – dies ist jedoch hier zunächst nicht der Fall. Das Verfahren eignet sich also in der Variante die hier evaluiert wurde nicht für den Einsatz bei einer derart kleinen Datenmenge. Ein Punkt stellt ein Dokument dar, die Farbe zeigt den Dokumententyp des Dokuments. Auf eine Beschriftung der einzelnen Datenpunkte wird aus optischen Gründen sowie aus Datenschutzgründen verzichtet. Es wurde erwartet, dass Dokumente des gleichen Dokumententyps Cluster bilden würden.

4.1 Ergebnisse: Ermittlung von Ähnlichkeiten

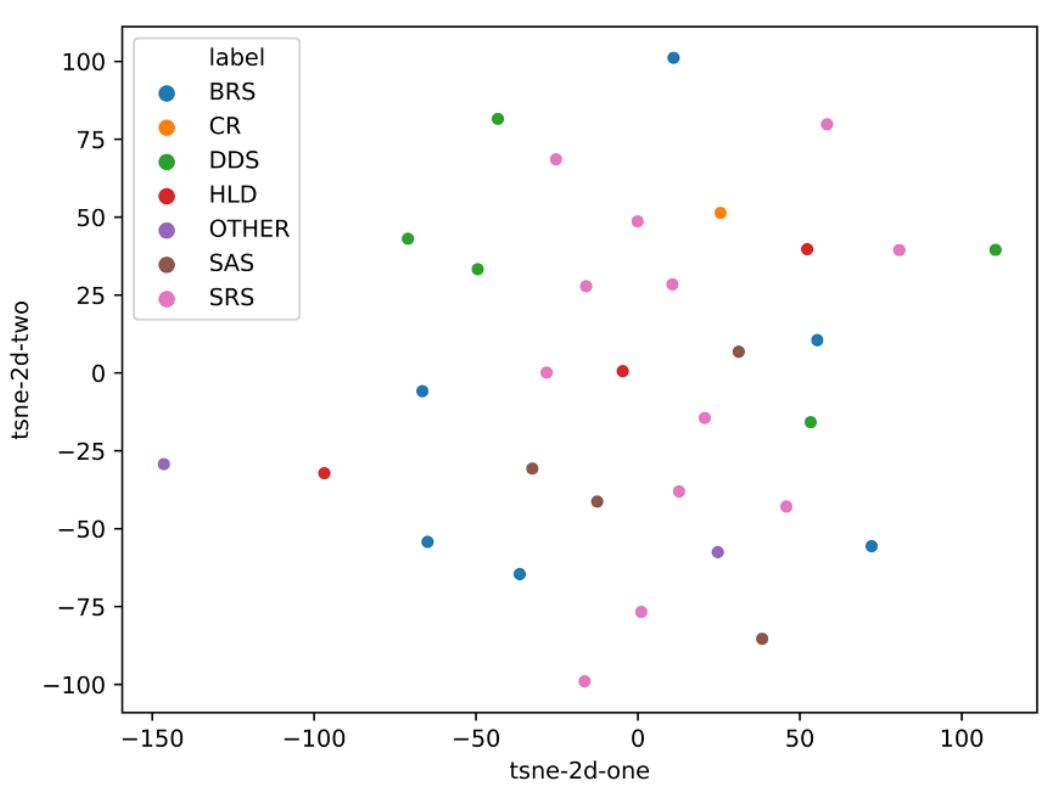


Abbildung 4.5: Darstellung aller verfügbaren Dokumente, gruppiert nach Dokumententypen, eigene Erstellung

Kriterium 2: Generelle Einsetzbarkeit des Ansatzes im APM-Kontext, unter der Annahme einer beliebig großen Datenmenge

Man kann mit dem Verfahren grundsätzlich Cluster bilden. Im Falle eines neuen Dokuments könnte die Position in der Visualisierung Hinweise darauf geben, welche ähnlichen Dokumente es bereits gibt. Die Dimensionsreduktion erfolgte ebenfalls mit t-SNE (vgl. Abbildung 4.6) [9].

Es werden in dem Beispiel ähnliche Wikipedia-Artikeln zu Dokumenten-Clustern zusammengefasst wodurch die generelle Einsetzbarkeit unter der Annahme einer beliebig großen Menge von Dokumenten bestätigt werden kann. Es kann allerdings nicht ausgeschlossen werden, dass die APM-Dokumente sich nicht genug voneinander unterscheiden, womit das Wikipedia-Ergebnis nicht erzielt werden könnte im Kontext des APM. In einer realen Anwendung könnte sich z. B. durch das Auswählen einzelner Datenpunkte ein Fenster mit Details zum jeweiligen Dokument öffnen.

4.1 Ergebnisse: Ermittlung von Ähnlichkeiten



Abbildung 4.6: Dokumenten-Cluster, erzeugt über Doc2Vec, aus [9]

Kriterium 3: Rechenaufwand

Das Erzeugen eines Scatter-Plots wie in Abbildung 4.6 erfordert zunächst das Training eines Modells. Die Trainingszeit ist abhängig von der Anzahl der Dokumente. Ein solches Modell kann auch nachts trainiert werden und dann gespeichert werden. Neue Dokumente können dann durch das Modell in Vektorform gebracht und visualisiert werden, z. B. in einer interaktiven Darstellung etwa auf einer Webseite. Die Inferenz erfolgt ohne Verzögerungen.

Kriterium 4: Kosten

Der Paragraph Vector-Ansatz benötigt keine beschrifteten Beispiele und kann daher auch in Szenarien eingesetzt werden, in denen es keine beschrifteten Daten gibt [35, S. 4]. Daher ist hier kein manueller Aufwand nötig, der Kosten verursachen würde. Lizenzkosten gibt es bei der Verwendung der Bibliothek gensim ebenfalls nicht [19].

4.2 Ergebnisse: Bestimmung von inhaltlicher Integrität

4.2.1 Ansatz 1: Vergleich von Durchschnitts-Wort-Vektoren

Dieser Ansatz basiert auf durchschnittlichen Wort-Vektoren. Auf Basis einer Trainingsdatenmenge werden zunächst durchschnittliche Wort-Vektoren berechnet. Die Trainingsdatenmenge setzt sich zusammen aus mehreren Textkörpern. Die unterschiedlichen Textkörper bestehen hier aus frei zugänglichen PDF-Dokumenten aus dem Internet. Unterschieden werden dabei die folgenden Dokumententypen:

- Texte in denen Anforderungen beschrieben werden (Thema Requirements)
- Texte zu Lösungs-Architekturen (Thema Architecture bzw. Architectural Language)
- Texte zu Software-Design (Thema Design)
- Texte zu Logistik-Themen (Thema Logistics)

Für jeden dieser Typen wird ein durchschnittlicher Wort-Vektor berechnet. Visualisiert wird das Vorgehen in Abbildung 4.7. Nicht gezeigt wird hier der Logistik-Textkorpus.

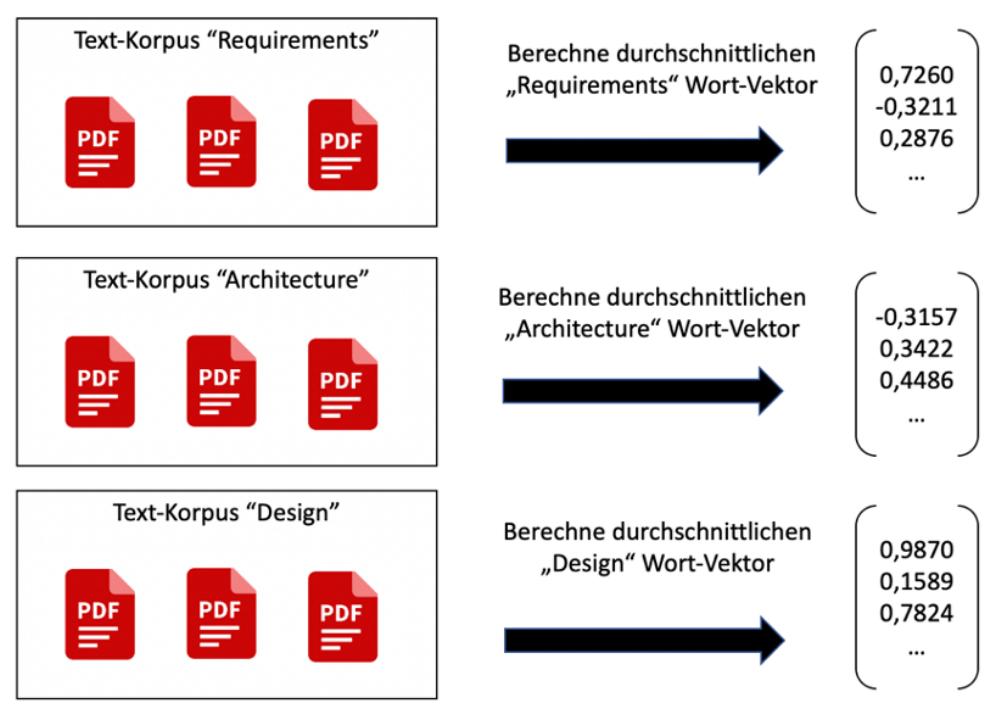


Abbildung 4.7: Initiale Vorbereitung von durchschnittlichen Wort-Vektoren, eigene Erstellung

4.2 Ergebnisse: Bestimmung von inhaltlicher Integrität

Ziel ist es nun auf Kapitelebene zu schätzen, ob es in einzelnen Kapiteln von Test-Dokumenten um eines der festgelegten Themen (Requirements, Architectural Language, Design oder Logistics) geht, wie in Kapitel 3.2 beschrieben. Das Thema Logistik dient dabei als Referenz- bzw. als Vergleichskorpus. Hintergrund ist stets die Prüfung, ob ein Kapitel enthält, was es laut Vorgabe enthalten sollte. Die Zielvariable heißt "erkanntes Thema" – sie kann verwendet werden, um die Wahrscheinlichkeit zu messen, mit der ein Kapitel die erwarteten Inhalte enthält. Die Wahrscheinlichkeiten ergeben sich, indem für jedes Kapitel in einem zu untersuchenden Beispiel-Dokument ebenfalls Vektoren erzeugt werden. Diese werden dann mit den Durchschnitts-Vektoren verglichen (vgl. Abbildung 4.8).

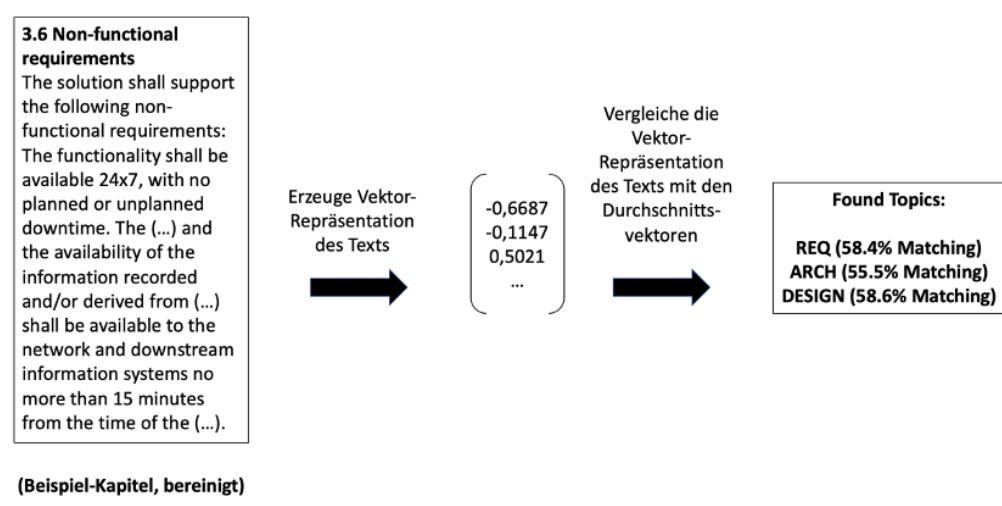


Abbildung 4.8: Inferenz-Phase. Vergleich von Kapitel-Vektoren mit Durchschnitts-Vektoren, eigene Erstellung

Kriterium 1: Einsetzbarkeit des Ansatzes unter den gegebenen Rahmenbedingungen

Als Trainingsmenge können beliebig viele Dokumente verwendet werden. Hier wurden pro Text-Korpus etwa 20 PDF-Dokumente aus dem Internet verwendet. Auf Basis dieser Dokumente wurden die Durchschnittsvektoren ermittelt. In Abbildung 4.8 werden mögliche Resultate schon angedeutet. Die Schätzungen haben meist mit mehreren Durchschnittsvektoren Ähnlichkeiten. Daher kann hier von keiner zuverlässigen Vorhersage der Topics gesprochen werden. Es werden hier auch keine Metriken im Sinne einer Konfusionsmatrix präsentiert, weil Abschnitte stets mit mehreren der Topics eine Ähnlichkeit hatten. Das zeigt auch der nachfolgende direkte Vergleich von den durchschnittlichen Vektoren (vgl. Tabelle 4.1).

4.2 Ergebnisse: Bestimmung von inhaltlicher Integrität

	Logistics	Architecture	Design	Requirements
Logistics	1	0.93	0.91	0.92
Architecture	0.93	1	0.99	0.98
Design	0.91	0.99	1	0.99
Requirements	0.92	0.98	0.99	1

Tabelle 4.1: Vergleich der Durchschnitts-Vektoren miteinander. Der Vergleich erfolgte mit der Similarity-Funktion von spaCy.

Als Ursache für die Ähnlichkeit der Vektoren (mit Ausnahme des Referenzkorpus “Logistics”, wo durchaus eine Unterscheidung möglich ist) wird von der inhaltlichen Nähe der Text-Körper zueinander (vgl. etwa “Design” und “Architecture”) ausgegangen. Die gewählten Topics sind sich inhaltlich sehr ähnlich, weshalb hier die sorgfältige Auswahl der Trainingsdokumente in den jeweiligen Kategorien eine Rolle spielen kann. Die einfache Durchschnittsbildung (averaging) führt also im Kontext des APM eher weniger zu sinnvollen Vektoren, weshalb mit einer derart geringen Datenmenge der Ansatz nicht sinnvoll genutzt werden kann. Aus den Ergebnissen folgt jedoch, dass man mit dem Verfahren zumindest ein Gefühl dafür bekommen kann, ob die Inhalte einzelner Kapitel grob in die richtige Richtung gehen. Texte wie etwa *lorem ipsum*, könnten als unsinnige Inhalte identifiziert werden.

Kriterium 2: Generelle Einsetzbarkeit des Ansatzes im APM-Kontext, unter der Annahme einer beliebig großen Datenmenge

Wie oben beschrieben (vgl. Tabelle 4.1), sind die gewählten Themen, mit Ausnahme des Themas Logistics, sich inhaltlich sehr ähnlich. Das ändert sich auch bei größeren Trainingsdatenmengen nicht, da die Dokumente sich inhaltlich stets sehr ähnlich sein werden. Daher kann hier die generelle Einsetzbarkeit dieses Ansatzes nicht bestätigt werden, sofern zwischen sehr ähnlichen Themen (wie im vorliegenden Kontext) unterschieden werden soll.

Die Autoren dieser Studie [13] evaluierten ebenfalls verschiedene Ansätze zur Kombination von Wort-Vektoren, so wie es bis hierhin beschrieben wurde. Allerdings unterscheidet sich der Ansatz der Autoren darin, dass kein schlichter Vektor-Vergleich mit Durchschnittsvektoren durchgeführt wird. Stattdessen werden Vektor-Repräsentationen des Texts noch als Eingabe für ein Klassifizierungsverfahren verwendet, wobei ein neuronales Netz zum Einsatz kommt. Dabei handelt es sich im Prinzip um einen eigenen Ansatz. Da es in diesem Abschnitt um den Vergleich von Durchschnitts-Vektoren ging, wird der Klassifizie-

4.2 Ergebnisse: Bestimmung von inhaltlicher Integrität

rungsansatz hier nicht näher betrachtet, er soll aber als Möglichkeit für weitere Versuche dokumentiert werden.

Kriterium 3: Rechenaufwand

Ein gebräuchliches Maß zur Bewertung der Ähnlichkeit ist die Kosinus-Ähnlichkeit. Der eigentliche Abgleich der Vektoren erfolgt mit der dieser Metrik, unter Verwendung der spaCy-Similarity-Funktion. Details zur Kosinus-Ähnlichkeit werden bereits in Abschnitt 4.1.1 unter Kriterium 3 beschrieben. Als Wort-Vektoren wurden hier die vortrainierten Vektoren von spaCy genutzt. Daher ist kein Aufwand für das Training notwendig, sofern das vortrainierte Modell gewählt werden soll. Alternativ können Wort-Vektoren auch selbst trainiert werden, dann wäre ein entsprechender Rechen- und Zeitaufwand zu berücksichtigen.

Kriterium 4: Kosten

Es können frei verfügbare Software-Bibliotheken (u. a. spaCy [60], NumPy [45]) verwendet werden. Kosten entstehen durch das Sammeln und das Verwalten von Dokumenten für das Training. Im vorliegenden Beispiel wurden zunächst im Internet Dokumente für das Training von Wort-Vektoren gesucht, was aufwändig wird, wenn eine hohe Anzahl an Dokumenten mit ausreichender Qualität gesucht werden soll.

4.2.2 Ansatz 2: Topic Modelling mit Non-negative matrix factorization

Wie oben bereits angedeutet, gäbe es für die Lösung des Anwendungsfalls “Bestimmung von inhaltlicher Integrität“ weitere Möglichkeiten aus den Bereichen des überwachten, unüberwachten oder auch semi-überwachten Lernens. Mit verschiedenen Klassifizierungsverfahren können die Klassen beziehungsweise die Themen von einzelnen Kapiteln aus Text-Dokumenten geschätzt werden. In Frage kämen etwa Ansätze wie Naive Bayes, logistische Regression, neuronale Netze, und andere. Hier vorgestellt wird nun ein Ansatz auf Basis von Non-negative matrix factorization (Abk.: NMF). Zu beachten ist, dass es sich dabei nicht um ein Klassifizierungsverfahren, sondern um eine Methode aus der Kategorie des Topic Modelling handelt. Statt mit NMF könnten ähnliche Resultate auch unter Verwendung des Verfahrens Latent Dirichlet Allocation (Abk.: LDA) realisiert und evaluiert werden [22].

4.2 Ergebnisse: Bestimmung von inhaltlicher Integrität

Als Trainingskorpus wird auch hier eine Menge von Dokumenten aus dem Internet verwendet. Verwendet wurden wiederum die Textkorpora Requirements, Architectural Language, Design sowie Logistics. Bei näherer Betrachtung des Trainingskorpus fällt auf, dass die Trainingsdokumente in den einzelnen Kategorien sich durchaus durch bestimmte, typische Begriffe auszeichnen. Diese Begriffe sind die bedeutsamsten für ein bestimmtes Thema. Zur Visualisierung dieser Begriffe von Topics können Ansätze wie NMF oder LDA genutzt werden. Diese Ansätze verwenden intern als Vorverarbeitungsschritt die TF-IDF-Metrik. Zur Schätzung des Topics eines Testdokuments kann dann das erstellte Modell genutzt werden. Zunächst folgt allerdings in Abbildung 4.9 eine Visualisierung der für die Themen typischen Begriffe. Die Begriffe die von Bedeutung sind für ein Thema wurden automatisch durch das Verfahren extrahiert und sind unten dargestellt.

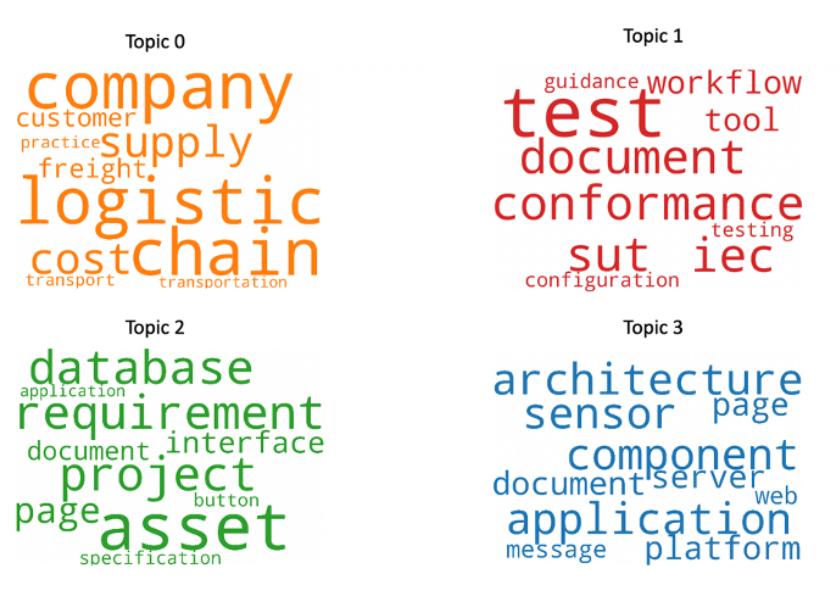


Abbildung 4.9: Tag Clouds zu 4 erkannten Topics, erstellt mit NMF (eigene Erstellung)

Mit bloßem Auge erkennbar sind in Abbildung 4.9 die Themen Requirements (Topic 2), Architectural Language (Topic 3) und Logistics (Topic 0). Für die Vorhersage des Topics eines neuen Dokuments (Testdokumente) werden noch weitere als die je 10 hier dargestellten Begriffe verwendet. Es handelt sich jedoch bei den Begriffen in Abbildung 4.9 um jene Begriffe, die laut Modell am aussagekräftigsten für die Topics sind. Die Vorhersage des Themas eines Testdokuments liefert als Ergebnis eine Zuordnung zu einem der vier Themen. Ein Beispiel ist abgebildet in Abbildung 4.10.

4.2 Ergebnisse: Bestimmung von inhaltlicher Integrität

2. → Application Architecture¶

This [REDACTED] application architecture is remain as [REDACTED] v1.1. Refer to architecture diagram as shown below for overall overview of [REDACTED] architecture.¶

Prediction → Topic 3

Abbildung 4.10: Verwendung eines Abschnitts aus einem Testdokument für die Vorhersage des Topics, eigene Erstellung

Dieses Verfahren sagt dem Nutzer lediglich, dass zum Beispiel Topic 3 als Ergebnis einer Vorhersage bestimmt wurde. Das Verfahren sagt nicht, wie das Topic zu interpretieren ist, also um welche “Klasse“ es sich bei dem Topic handelt. Die “Klassen“ müssen deshalb manuell auf Basis der Tag Clouds (Abbildung 4.9) bestimmt werden.

Kriterium 1: Einsetzbarkeit des Ansatzes unter den gegebenen Rahmenbedingungen

Es wurden per Hand Text-Beispiele für Tests ausgewählt und entsprechende Beschriftungen erzeugt. Es wurden manuell 4 Texte dem Thema Software Design zugeordnet, 13 der Kategorie Requirements, und 3 wurden als Architectural Language eingeordnet. Für jedes Beispiel wird eine Schätzung mit dem NMF-Modell durchgeführt. Die Ergebnisse der Schätzungen sind unten (Tabelle 4.2) dargestellt. Topic 0 (“Logistics“) kann als Kontroll-Klasse verstanden werden, die eigentlich nicht vorhergesagt werden sollte. Die Klasse “Requirements“ könnte aus fachlicher Sicht auch in zwei Klassen functional und non-functional aufgeteilt werden. Diese Unterscheidung wurde hier nicht gemacht.

		Tatsächliche Klasse		
Schätzung		Software Design	Requirements	Architectural Language
	Topic 0	0	0	0
	Topic 1	0	0	0
	Topic 2	0	10	0
	Topic 3	4	3	3

Tabelle 4.2: Konfusionsmatrix zur Darstellung der NMF-Schätzungen

Es wurde also für 20 Test-Abschnitte eine Vorhersage mit dem Modell erstellt werden. Da hier lediglich 20 Schätzungen erstellt wurden, liegt an den wenigen qualitativ hochwertigen Dokumenten, die für die Evaluation genutzt werden konnten. Ein Ausschnitt aus den für die Evaluation verwendeten Daten ist in Abbildung 4.11 abgebildet. Für die Interpretation der Konfusionsmatrix wird auf Abbildung 4.9 verwiesen. Zu erkennen ist, dass etwa 77% der Texte, die manuell als Requirements markiert wurden, auch durch das Modell in die korrekte Klasse eingeteilt wurden (10/13). Bei Architectural Language werden alle drei

4.2 Ergebnisse: Bestimmung von inhaltlicher Integrität

Beispiele in die richtige Klasse eingeteilt. Die Klasse Software Design hingegen wird in die gleiche Kategorie wie die Texte aus Architectural Language eingeteilt. Das kann daran liegen, dass sich diese beiden Kategorien inhaltlich sehr stark überschneiden. Abschnitte für “Software Design“ stammen hauptsächlich aus DDS-Dokumenten (Detail(ed) Design Specification), Abschnitte für “Requirements“ stammen hauptsächlich aus BRS-Dokumenten (Business Requirement Specification), Abschnitte für “Architectural Language“ stammen hauptsächlich aus SAS-Dokumenten (Solution Architecture Statement).

source_file	source_chapter	actual_label_manually_identified	prediction	text
2017-12-29_DOKXX9XXXXXX9XXXXXX9XXXXXX10.docx	2.3 Architecture Overview	Software Design	3	"Architecture Overview Th
GXXXXXGXXXXX9XXXXXX1XXXXXX2.1.docx	3. High-level Business Change Statement	Requirements	3	"3. High-level Business Cha
EXXXXXX9XXXXXX9XXXXXX9XXXXXX	Functional Requirements	Requirements	2	"2. Functional Requirements
BXXXXXX9XXXXXX9XXXXXX9XXXXXX	4 Detailed Business Requirements	Requirements	2	"4 Detailed Business Requi
AXXXXXX9XXXXXX9XXXXXX9XXXXXX	5 User Interface and Functional Requirements	Requirements	2	"5 User Interface and Func
2018-06-06-DOV/XXXXXX9XXXXXX9XXXXXX_DDS_v1.12	2 Design Approach	Software Design	3	"2 Design Approach and En
2018-06-06-DOV/XXXXXX9XXXXXX9XXXXXX_DDS_v1.11.docx	2 Design Approach and Environment	Software Design	3	"2 Design Approach and En
OBXXXXXX9XXXXXX9XXXXXX_v0.2.docx	2. Application Architecture	Software Design	3	"2. Application Architectur
DXXXXXX9XXXXXX9XXXXXX9XXXXXX	5 Solution Architecture	Architectural Language	3	"5 Solution Architecture Th
DXXXXXX9XXXXXX9XXXXXX9XXXXXX	5 Solution Architecture	Architectural Language	3	"5 Solution Architecture Th
DXXXXXX9XXXXXX9XXXXXX9XXXXXX7.docx	5 Solution Architecture	Architectural Language	3	"5 Solution Architecture Th
SRS_XXXXXXX9XXXXXX9XXXXXX9XXXXXX	3. Functional Requirements	Requirements	2	"3. Functional requirement
SRS_XXXXXXX9XXXXXX9XXXXXX9XXXXXX	4. Non-Functional Requirements	Requirements	2	"4. Non-functional requir
SRS_XXXXXXX9XXXXXX9XXXXXX9XXXXXX	3. Functional Requirements	Requirements	2	"3. Functional requirement
SRS_XXXXXXX9XXXXXX9XXXXXX9XXXXXX	4. Non-Functional Requirements	Requirements	2	"4. Non-functional requir
XXXXXX9XXXXXX9XXXXXX9XXXXXX	Scope of Enhancement	Requirements	2	"2. Scope of Enhancement
XXXXXX9XXXXXX9XXXXXX9XXXXXX	Scope of Enhancement	Requirements	2	"2. Scope of Enhancement
2018-11-29_RXXXXXX9XXXXXX9XXXXXX_SRS_v1.16.docx	System Requirements	Requirements	3	"2 System Requirements 2
InXXXXXX9XXXXXX9XXXXXX9XXXXXX180606_v2.docx	Functional Requirements	Requirements	2	"2. Functional requirement
Intercept_SXXXXXX9XXXXXX9XXXXXX9XXXXXX2.docx	Non-functional Requirements	Requirements	3	"3 Non-functional requir

Abbildung 4.11: Struktur der Daten, die für Evaluation in diesem Abschnitt genutzt wurden (geschwärzt und gekürzt)

Kriterium 2: Generelle Einsetzbarkeit des Ansatzes im APM-Kontext, unter der Annahme einer beliebig großen Datenmenge

Wie der Code im Anhang dieser Arbeit zeigt, arbeitet der Ansatz unüberwacht. Das bedeutet, dass auch größere Textmengen für die Erstellung des Modells genutzt werden könnten. Da bereits unter Kriterium 1 gezeigt wurde, dass der Ansatz durchaus Themen schätzen kann, wird das auch bei einer beliebig großen Datenmenge möglich sein. Zu beachten ist, dass die Themen sich inhaltlich stark genug voneinander unterscheiden sollten.

Kriterium 3: Rechenaufwand

Eine Implementierung des Verfahrens ist gekapselt in der scikit-learn-Funktion NMF, weshalb auf eine detaillierte Algorithmus-Beschreibung verzichtet wird. Das Verfahren zeichnet sich jedoch durch eine polynomiale Komplexität aus. Das bedeutet, dass der Rechenaufwand je nach Parameterwahl bei der Initialisierung des Modells entsprechend stark steigen kann, je nachdem welche Parameter ausgewählt werden. Zu den Parametern

zählt sowohl die Anzahl der Themen, sowie die Textmenge, die für die Modellerstellung verwendet wird [22].

Kriterium 4: Kosten

Es wurden für das Trainings des Modells aufgrund der geringen verwendbaren Datenmenge PDF-Dokumente aus dem Internet verwendet. In der Realität müsste ebenfalls eine ausreichende Trainingsdatenmenge bereitgestellt werden. Dafür sind Texte zusammenzutragen, und es ist insbesondere auf die Qualität der Texte zu achten. Dadurch entstünden Kosten. Kosten für die Nutzung des Verfahrens, z. B. unter Verwendung von Bibliotheken wie scikit-learn [55], ergeben sich zunächst nicht.

4.3 Ergebnisse: Automatisierte Erzeugung von Text-Zusammenfassungen

In Abschnitt 3.3 wird begründet, was unter den Ansätzen zur Erzeugung von Text-Zusammenfassungen verstanden wird und wofür sie sinnvoll sein können. Nachfolgend evaluiert werden zunächst ein Ansatz auf Basis von TF-IDF (Abschnitt 4.3.1), gefolgt von einem Ansatz auf Transformer-Basis (Abschnitt 4.3.2).

4.3.1 Ansatz 1: Nutzung eines extraktiven Verfahrens auf Basis von TF-IDF

Vor der Evaluation soll kurz die Funktionsweise dieses Ansatzes skizziert werden. Die Implementierung besteht aus mehreren Schritten. Eine detaillierte Implementierung mit Kommentaren befindet sich in einem Zip-Archiv, dass dieser Arbeit beigelegt wird.

Der gesamte Textkorpus wird in einzelne Sätze aufgeteilt. Das Vorkommen von Wörtern in einzelnen Sätzen wird anschließend gezählt. Daraus kann nun die TF-Metrik berechnet werden, wie im Theorieteil beschrieben. Im Anschluss wird für jeden Begriff berechnet, in wie vielen Dokumenten dieser vorkommt.

Der nächste Schritt besteht in der Multiplikation der beiden Metriken. Sätze aus dem Text, der zusammengefasst werden soll, werden nun ausgewählt. Das geschieht, indem die TF-IDF-Werte der Begriffe innerhalb eines Satzes addiert werden. Dieser Wert wird geteilt durch die Länge des jeweiligen Satzes. Die Sätze mit den höchsten Werten gelten damit als die wichtigsten Sätze des Dokuments. Ein Grenzwert bestimmt letztendlich, welche Sätze in die finale Zusammenfassung übernommen werden.

Kriterium 1: Einsetzbarkeit des Ansatzes unter den gegebenen Rahmenbedingungen

Es wurde für diese Evaluation eins der zur Verfügung stehenden Dokumente ausgewählt. Um die Güte der durch den Ansatz erzeugten Zusammenfassungen objektiv bewerten zu können, werden von Menschen erzeugte Referenz-Zusammenfassungen benötigt. Diese haben in etwa die gleiche Länge wie die Zusammenfassungen, die automatisch generiert wurden. Bei der Erstellung dieser Referenz-Zusammenfassungen wurde darauf geachtet, dass möglichst viele Sätze so wie sie waren übernommen wurden, ohne manuelles Umschreiben. Es stellte sich heraus, dass die Auswahl von aussagekräftigen Sätzen auch für Menschen, die wenige Hintergrundinformationen zu den eigentlichen Inhalten der Dokumente besitzen, bereits eine Herausforderung ist.

Eine ideale durch ein Modell erzeugte Zusammenfassung hätte einen F-Score von 1. In diesem Fall entspräche die Zusammenfassung der durch Menschen erzeugten Referenz-Zusammenfassung. Der F-Score kann interpretiert werden als gewichteter Mittelwert von Precision und Recall [11]. Abbildung 4.12 stellt sehr geringe F-Scores dar. Das zeigt, dass die automatisch generierten Zusammenfassungen recht weit entfernt von Referenz-Zusammenfassungen liegen. Die Entwicklung der Metrik hängt auch mit der Länge von erzeugten Zusammenfassungen zusammen. Die Länge wird stets kleiner bzw. sie geht gegen Null, wenn der verwendete Korpus vergrößert wird. Referenz-Zusammenfassungen sind von Menschen mit entsprechender Sorgfalt erstellt worden. Diese Sorgfalt kann dem extraktiven Ansatz nicht bescheinigt werden, d. h. die Qualität der extraktiven Zusammenfassungen ist gering.

Kriterium 2: Generelle Einsetzbarkeit des Ansatzes im APM-Kontext, unter der Annahme einer beliebig großen Datenmenge

Erwartet wurde zu Beginn, dass bei steigender Datenmenge eine Verbesserung der Rouge-Metrik erreicht werden kann. Wie in Abbildung 21 gezeigt wurde, ist eine solche Tendenz

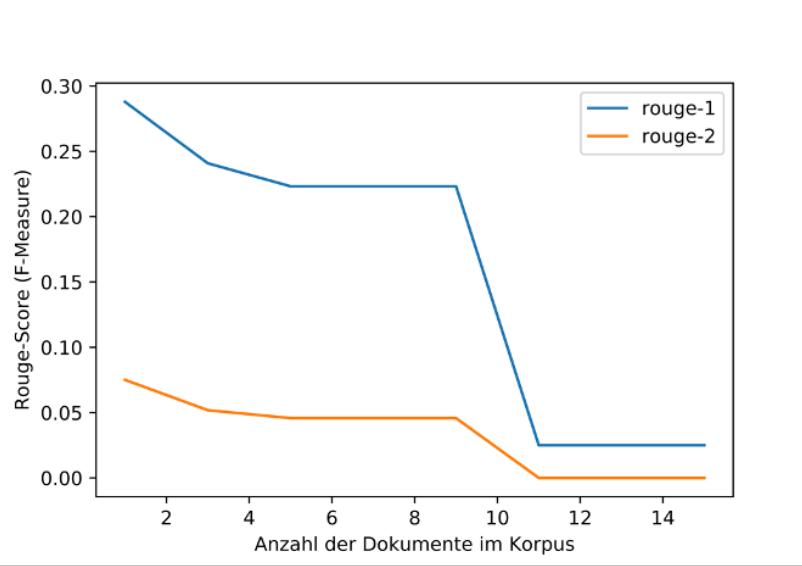


Abbildung 4.12: Evaluation der Güte von Zusammenfassungen, unter Veränderung des Ausgangs-Text-Korpus. Eigene Erstellung

nicht erkennbar. Eine solche Entwicklung konnte nicht beobachtet werden, weshalb das Verfahren auch bei größeren Datenmengen keine besseren Ergebnisse erbringen wird. Wie gezeigt wurde, wird die Güte der Zusammenfassungen von der Größe des Korpus beeinflusst. Die maximale Güte ergibt sich jedoch unter Verwendung von nur einem Text-Dokument als Korpus.

Kriterium 3: Rechenaufwand

Dieses Verfahren verwendet eine statistische Herangehensweise und benötigt keine Iterationen oder Ähnliches für die Berechnung der Metriken, die letztendlich bestimmen, welche Sätze in die finale Zusammenfassung aufgenommen werden. Daher kann der notwendige Rechenaufwand vernachlässigt werden, wenn der Textkorpus ausreichend klein gehalten wird.

Kriterium 4: Kosten Wenn davon ausgegangen wird, dass die durch das Verfahren erzeugten Zusammenfassungen sinnvoll genutzt werden könnten, dann würde das Verfahren keine Kosten verursachen, da keine Beschriftungen zu erzeugen sind und Lizenzkosten nicht zu erwarten sind.

4.3.2 Ansatz 2: Nutzung eines abstraktiven Verfahrens auf Transformer-Basis

Im Vergleich zu den extraktiven Verfahren ist das Gebiet der abstraktiven Verfahren zur Erzeugung von Text-Zusammenfassungen eher Gegenstand aktueller Forschung, als eine Klasse von in der Praxis nutzbaren Verfahren [68, S. 261]. Abbildung 4.13 zeigt die erzielten Ergebnisse in einem eigenen Versuch. Evaluiert wird hier das von Google-Forschern entwickelte Modell T5: Text-To-Text-Transfer-Transformer [49].

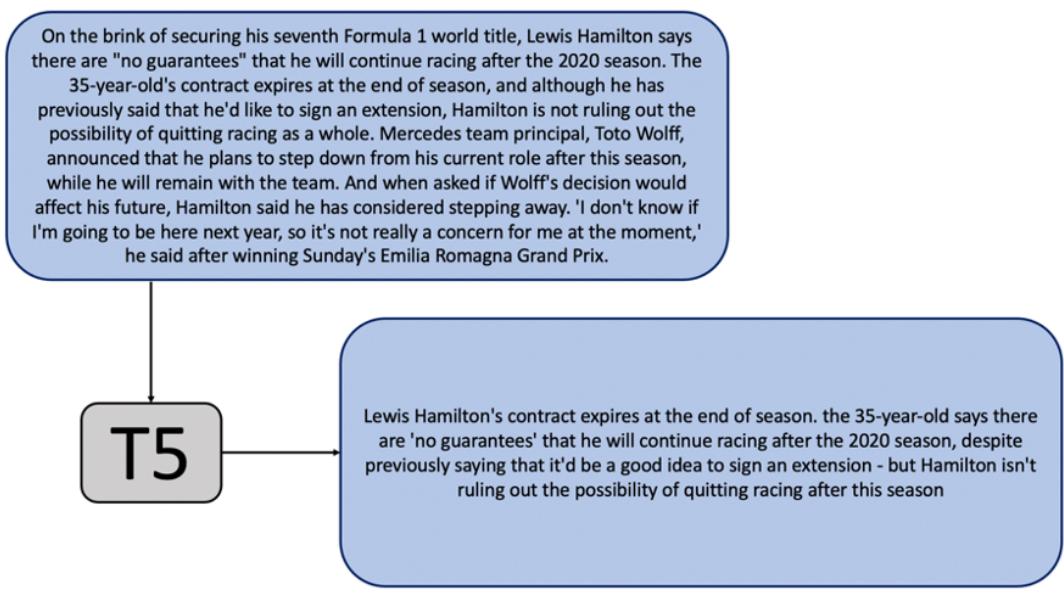


Abbildung 4.13: Veranschaulichung eines abstraktiven Verfahrens. Eigene Erstellung in Anlehnung an [49]

Kriterium 1: Einsetzbarkeit des Ansatzes unter den gegebenen Rahmenbedingungen

Es ist möglich, ein vorbereitetes Modell zu verwenden. Auf dieser Grundlage kann eine Fein-Abstimmung durchgeführt werden. Damit könnte erreicht werden, dass das Modell speziell auf Governance-Themen abgestimmt wird. Die Implementierung der Methode wurde im Rahmen der vorliegenden Evaluation in Python realisiert. Dafür konnte das Package Transformers (v3.0.2) verwendet werden. Es ermöglicht die Verwendung von vortrainierten Modellen unter dem Slogan “state-of-the-art NLP for everyone“ [26].

Es wurden in einer Veröffentlichung CNN/ Daily Mail-Nachrichtentexte unter Verwendung des Modells zusammengefasst und diese Zusammenfassungen wurden mit der Rouge-

Metrik evaluiert. Die Ergebnisse dieses Ansatzes setzen neue Maßstäbe bei der Qualität von automatisch generierten Zusammenfassungen [49, S. 39]. Daher kann gesagt werden, dass die Qualität der Zusammenfassungen als gut eingestuft werden kann, sofern eine Fein-Abstimmung auf den jeweiligen Kontext stattgefunden hat. Die maximale Länge der Eingabe-Sequenz des T5-Modells beträgt nach aktuellem Stand 512 Tokens. Damit können als Eingabe für Zusammenfassungen bislang nur Texte mit maximal 512 Wörtern verwendet werden [49, S. 39]. Das reicht maximal zur Erzeugung der Zusammenfassung einzelner Abschnitte, die dann kombiniert werden könnten. Das würde jedoch nicht dazu beitragen, dass die wichtigsten Informationen aus einem Text extrahiert werden, was hier eigentlich angestrebt werden sollte.

Longformer (The Long-Document Transformer) ist ein Transformer-Ansatz, der als Alternative zum T5-Modell in Frage käme. Hier wird kontinuierlich versucht die maximale Sequenzlänge der Eingabe zu erhöhen, dennoch gibt es auch hier derzeit noch Beschränkungen auf 4096 Tokens als Eingabe-Sequenz [6] [23], was ebenfalls als nicht ausreichend beurteilt werden muss.

Als Fazit wird festgehalten, dass sich die Ansätze noch zu sehr im Entwicklungsstadium befinden und durch die Beschränkung auf sehr kleine Eingabesequenzen keine sinnvollen Ergebnisse im vorliegenden Kontext erzielt werden können. Die notwendige Datenmenge für eine Feinabstimmung wird nachfolgend erläutert. Vorweggenommen werden soll, dass 20-40 Textdokumente dafür nicht ausreichend sind. Unter den gegebenen Rahmenbedingungen ist dieses abstraktive Verfahren also nicht sinnvoll einsetzbar.

Kriterium 2: Generelle Einsetzbarkeit des Ansatzes im APM-Kontext, unter der Annahme einer beliebig großen Datenmenge

Wie in Abbildung 4.13 verdeutlicht wird, sind grundsätzlich Zusammenfassungen generierbar. Auch im Kontext der IT-Governance geht das, wobei das oben dargestellte Beispiel nur zur Verdeutlichung des Ansatzes dient. Bei der Erstellung des vortrainierten Modells wurde mit dem Colossal Clean Crawled Corpus (C4) eine Datenmenge verwendet, die aus hunderten Gigabytes bereinigten Text aus dem Internet besteht [49, S. 5-7]. Bei Texten aus dem Internet ist davon auszugehen, dass diese anders zusammengesetzt sind als die Software- oder Architektur-Dokumentationen, um die es im vorliegenden Kontext geht. Daher ist ein solches vortrainiertes Modell noch nicht abgestimmt für den Anwendungsfall. Es sind für eine Fein-Abstimmung entsprechend viele Trainingsdaten notwendig. In einer

Veröffentlichung wurde die Fein-Abstimmung auf Basis des CNN-Daily Mail-Datensatzes durchgeführt [49, S. 37]. Die Größe dieses Datensatzes liegt bei 1.27 Gigabytes. Mit einer solchen Größe ließen sich, wie unter Kriterium 1 bereits beschrieben wurde, auch im APM-Kontext hochwertige Zusammenfassungen erzeugen, allerdings mit der Beschränkung auf 512 Tokens als Eingabesequenz.

Die Fein-Abstimmung ist insbesondere von Bedeutung, weil IT-Dokumentationen in unterschiedlichen Regionen der Welt geschrieben worden sein können. Grammatikalische Regeln werden daher nicht immer eingehalten und Formulierungen können durch lokale Besonderheiten geprägt sein. Vortrainierte Sprachmodelle allein sind daher weniger geeignet, um sinnvolle Zusammenfassungen aus den Dokumentationen zu erzeugen. Ein weiteres Hindernis stellen Abbildungen dar, auf die in den Text-Dokumenten nicht ausreichend eingegangen wird – sie enthalten häufig wichtige Informationen, die durch NLP-Verfahren alleine nicht berücksichtigt werden können. Es müssten zusätzlich weitere ML-Ansätze zur Bildverarbeitung angewendet werden, um optimale Ergebnisse erzielen zu können. Zusammengefasst könnte mit entsprechendem Aufwand eine Feinabstimmung erreicht werden, aufgrund der Beschränkung auf 512 Tokens ist der Ansatz der hier untersucht wurde jedoch (noch) nicht für das APM geeignet.

Kriterium 3: Rechenaufwand

Die Zusammenfassungs-Funktionalitäten können unter Verwendung der Bibliotheken direkt genutzt werden, dann allerdings zunächst ohne Fein-Abstimmung. Diese Inferenz dauert auf gängiger Hardware wenige Sekunden, wie beispielhafter Code zeigt, der im Anhang der Arbeit zu finden ist. Versuche der T5-Autoren deuten darauf hin, dass mit mindestens einigen Stunden für die Feinabstimmung zu rechnen ist [49].

Kriterium 4: Kosten

Die Modelle (z. B. t5-small [25]) sowie die Transformer-Bibliothek [26] sind frei verfügbar. Die vortrainierten Modelle stehen zur Verfügung und können verwendet werden. Nicht zu vernachlässigende Kosten können entstehen durch die Zusammenstellung von Trainingsdaten für die Feinabstimmung eines Modells. Das bedeutet, dass für die Feinabstimmung von Menschen erstellte Zusammenfassungen benötigt werden. Wie beschrieben wurde, kann sich am CNN/ Daily Mail-Beispiel orientiert werden, wo mehrere Gigabytes an Text (Original-Texte + Zusammenfassungen) verwendet wurden.

4.4 Ergebnisse: Automatisierte Schnittstellen-Erkennung

Die beiden nachfolgenden Ansätze verwenden beschriftete Beispiele für das Training eines Klassifikationsmodells. Die so trainierten Modelle können anschließend genutzt werden, um einzelne Sätze von Textdokumenten in eine der beiden Klassen “Technical Interface (TI)“ oder “kein technisches Interface“ einzuordnen. Wenn die Vorhersage des Modells einem Satz aus der Testmenge die Klasse TI zuteilt, dann kann diese Vorhersage genutzt werden, um Sätze zu identifizieren, in denen eine technische Schnittstelle beschrieben wird. Die Genauigkeit (accuracy) dieser Vorhersage gilt es zu maximieren. Die Qualität und die Menge der Trainingsdaten haben entscheidenden Einfluss auf die Qualität der Vorhersage, wie nachfolgend beschrieben werden soll.

4.4.1 Ansatz 1: Klassifizierung auf Grundlage eines BERT-Modells

BERT, ein vortrainiertes Transformer-Modell, wird hier für die Klassifizierung verwendet, indem das vortrainierte Modell per Fine-Tuning auf den vorliegenden Kontext abgestimmt wird. Dabei werden weitere Schichten eines neuronalen Netzes trainiert, die hinter denen das BERT-Modells liegen, um eine Klassifizierungsaufgabe lösen zu können, wie beschrieben z. B. in [64, S. 2]. Dieser Ansatz heißt Transfer-Learning. Das BERT-Modell kennt bereits grundlegende Eigenschaften von natürlicher Sprache. Es wurde trainiert unter Verwendung des BookCorpus mit 800 Millionen Wörtern sowie der gesamten Englischsprachigen Wikipedia (2500 Millionen Wörter) [12].

Kriterium 1: Einsetzbarkeit des Ansatzes unter den gegebenen Rahmenbedingungen

Die für die Feinabstimmung verwendeten Daten setzen sich zusammen aus 195 positiven Beispielen (Klasse TI), und 195 negativen Beispielen (zur Hälfte Sätze zu User Interfaces, zur anderen Hälfte Sätze vollständig ohne Interface-Information). Sie wurden manuell beschriftet. Es werden 70% der Daten für das Training verwendet, 15% für das Testen, und 15% für die Validierung. Die Sätze wurden in einem vorausgegangenen Arbeitsschritt mit einem Ansatz, der auf Ontologien basiert, aus den Textdokumenten des Kunden extrahiert. Der erste Versuch ergab einen f1-score von 70% (vgl. Abbildung 4.14). Die interessierende Klasse TI ist erkennbar über die Ausprägung 1. die Ausprägung 0 gibt an, dass die Vorhersage “kein TI“ lautete. Die 70% sind zunächst ein Zeichen dafür, dass Overfitting

4.4 Ergebnisse: Automatisierte Schnittstellen-Erkennung

hier nicht unbedingt ein Problem darstellt. Bei einer verdächtig hohen Genauigkeit müsste davon ausgegangen werden.

	precision	recall	f1-score	support
0	0.67	0.67	0.67	48
1	0.72	0.72	0.72	58
accuracy			0.70	106
macro avg	0.70	0.70	0.70	106
weighted avg	0.70	0.70	0.70	106

Abbildung 4.14: Klassifizierungsergebnisse unter Verwendung von BERT nach der Feinabstimmung. Eigene Bildschirmaufnahme

Die praktische Einsetzbarkeit in einem realen Szenario bei einer Erkennungsgenauigkeit von (etwa) zwei aus drei Fällen ist fraglich. Angemerkt werden muss, dass sowohl Trainings- als auch Testsätze aus dem gleichen Textkorpus stammen. Zur Folge hätte dies, dass in Produktion, d. h. bei Verwendung eines komplett anderen Dokuments, geschrieben in anderer Art- und Weise, das Modell viel schlechtere Leistung erbringen könnte, als bei den hier vorgestellten Ergebnissen vermutet werden kann. Nachfolgend soll diese Vermutung anhand eines Beispiels evaluiert werden. Es wird eine “Requirement Specification“ aus dem Internet verwendet, um darin vorkommende Sätze in eine der beiden Klassen (“TI“ oder “kein TI“) einzuteilen. Interessant ist, dass dieses Verfahren zunächst nicht direkt eine Klasse als Vorhersage ausgibt, sondern zwei Werte. Das sind die direkten Ausgaben des neuronalen Netzes, das für die Klassifizierung verwendet wird. Es wird der absolut größere Wert verwendet für die Bestimmung der Klasse. Eindeutig als “kein technisches Interface“ (Klasse 0) können damit Sätze bestimmt werden, die offensichtlich kein Interface beschreiben (Abbildung 4.15). Bei längeren Sätzen sind in diesem Beispiel allerdings auch viele Sätze dabei, die nichts mit Interfaces zu tun haben, aber als solche klassifiziert wurden (Abbildung 4.16).

Es wird festgehalten, dass die Feinabstimmung mit der gegebenen Datenmenge nicht ausreichend ist, um in der Praxis mit dem Ansatz sinnvolle Ergebnisse zu erreichen. Die Kernaussage ist aber, dass der Ansatz im Sinne eines Proof-of-Concepts funktionsfähig und in der Perspektive vielversprechend ist.

4.4 Ergebnisse: Automatisierte Schnittstellen-Erkennung

Text	▼ 1	▼ 0	▼
The visiting foreign business executive S5 „Äl Sebastian.	-0,26	-1,48	
User selects „Äregister,Ä function 2.	-0,27	-1,46	
10.	-0,27	-1,45	
User can select only subset of questions to submit (he/she doesn,Ät have to fill the whole form).	-0,27	-1,44	
609023 6.3.3 Track a Journey	58 6.3.4 Online feedback to „Äroad conditions and limitati	-0,27	-1,43
9.	-0,29	-1,39	
PERSONAL INFORMATION: Name, Surname Male/Female Email Mobile phone (for booking the services) Home Address Home GPS location PRIVATE VEHICL	-0,29	-1,39	
Figure 5.	-0,29	-1,37	
CONCLUSIONS	69 8.	-0,3	-1,36
REQUIREMENTS FOR THE STANDARDS COMPLIANCE	52 TABLE 20.	-0,3	-1,36
Table 12.	-0,3	-1,35	
Table 18.	-0,3	-1,35	
User selects one journey plan, which will be followed further on 11.	-0,3	-1,35	

Abbildung 4.15: Klasse 0 (“kein TI“) gewinnt und bestimmt damit die Vorhersage. Eigene Bildschirmaufnahme

Text	▼ 1	▼ 0	▼
User selects function „Äedit user profile,Ä –© MyWay Consortium 65 D2.1 Requirement specification and analysis ECGA No.	-1,94	-0,16	
The feedback form contains: –© MyWay Consortium 63 D2.1 Requirement specification and analysis ECGA No.	-1,8	-0,18	
–© MyWay Consortium 68 D2.1 Requirement specification and analysis ECGA No.	-1,53	-0,25	
Actors: Anonymous user Registered user MyWay –© MyWay Consortium 59 D2.1 Requirement specification and analysis ECGA No.	-1,52	-0,25	
Therefore, the resulting requirements specification has to serve as a basis for the definition and design of the MyWay components in the reference architecture.	-1,48	-0,26	
Finally, WP2 (Reference Design & Architecture) also produces specific requirements gathered during the system design process.	-1,46	-0,26	
4.1 Scenarios A Scenario is a narrative short story which describes how the actors could use a system or application.	-1,42	-0,28	
In other words, a user story synthesizes ideally in one sentence a specific functionality that an end user may perform in the system.	-1,39	-0,29	
The resulting requirements specification has to serve as a basis for the definition and design of the MyWay components in the reference architecture.	-1,38	-0,29	
US2-S1 - Mrs. Ginger travels by train, tram, bus.	-1,38	-0,29	
–© MyWay Consortium 67 D2.1 Requirement specification and analysis ECGA No.	-1,36	-0,3	
To gather them, after an introduction about the MyWay architecture, a requirements elicitation methodology has been defined and followed.	-1,35	-0,3	
609023 Planning graph maintainer - Creates and maintains a planning graph from the external city data.	-1,33	-0,31	
REQUIREMENTS In MyWay, the requirements describe the single functionalities to perform in the system.	-1,32	-0,31	

Abbildung 4.16: Beispiele, bei denen Klasse 1 als Vorhersage bestimmt wurde. Eigene Bildschirmaufnahme

Kriterium 2: Generelle Einsetzbarkeit des Ansatzes im APM-Kontext, unter der Annahme einer beliebig großen Datenmenge

Um die Güte des Verfahrens messen zu können, wären weitere beschriftete Daten notwendig. Diese müssen in ausreichender Qualität vorhanden sein, und am besten durch Experten beschriftet worden sein. Es wäre auch denkbar, dass mehrere Experten die Beschriftungen vergeben und letztendlich nur die Beschriftung für das Training verwendet wird, die von mehreren Personen als Beschriftung ausgewählt wurde. Damit würden Zweifel eliminiert werden, darüber welche Klasse die richtige für einzelne Sätze ist. In anderen Veröffentlichungen wird von Genauigkeiten von bis zu 97% bei ausreichend großen Datenmengen berichtet [64]. Daher wird zusammengefasst, dass dieser Ansatz generell einsetzbar sein könnte, um Schnittstellen in Texten zu identifizieren, allerdings erst dann, wenn beschriftete Trainings-Beispiele in ausreichender Menge vorliegen.

Kriterium 3: Rechenaufwand

Es wird lediglich die Fein-Abstimmung eines BERT-Modells durchgeführt. Ein großer Teil der Gewichtungen des Netzes ist also bereits verfügbar und muss nicht erneut berechnet

werden [24]. Das Training kann auf einer CPU oder auf einer GPU durchgeführt werden². Die Dauer der Fein-Abstimmung ist stets abhängig davon welche Prozessorarchitektur verwendet werden kann. Außerdem ist sie von gewählten Hyperparametern (Epochen/ Learning Rate, etc.) abhängig. Auf einer CPU dauert die Feinabstimmung länger als auf einer GPU. Es muss mit einigen Minuten auf gewöhnlicher Hardware kalkuliert werden. Die Inferenz (Vorhersage) nach dem Training ist ohne größere Verzögerungen möglich.

Kriterium 4: Kosten

Das Auszeichnen von Sätzen ist der Kostentreiber bei diesem Ansatz. Dieser Aufwand erfolgt bei Einführung des Ansatzes, im Laufe der Zeit können jedoch Anpassungen notwendig werden. Es wäre denkbar, dass mehrere Menschen über die Beschriftungen entscheiden, etwa nach dem Mehrheitsprinzip: es wird die Beschriftung von dem Modell verwendet, für die sich die Mehrheit entschieden hat. Da Kenntnisse auf Expertenniveau benötigt werden, kann dieser Ansatz entsprechende Kosten verursachen. Lizenzkosten sind nicht zu erwarten, da die Apache-2.0-Lizenz verwendet wird [24].

4.4.2 Ansatz 2: Klassifizierung mit logistischer Regression auf Basis von N-Grammen

Das Ziel ist hier wie bereits schon in Abschnitt 4.4.1 die Erkennung von Interface-Sätzen. In Kontrast zu dem vorangegangenen Ansatz, der auf Deep Learning sowie dem Transformer-Modell basierte, wird hier nun ein anderer Ansatz untersucht. Dadurch können die Ergebnisse der beiden Klassifizierungs-Ansätze besser eingeschätzt und verglichen werden.

Es handelt sich bei diesem Ansatz um die Klassifizierung von einzelnen Sätzen mit logistischer Regression. Es wird auf Wort-Vektoren oder ein vortrainiertes Transformer-Modell verzichtet und stattdessen werden aus Texten nur N-Gramme gebildet, die später als Features für die logistische Regression verwendet werden. Es werden die typischen Schritte [68, S. 124] für eine Klassifizierung verfolgt. Die Implementierung ist überschaubar, weshalb sie anhand des schematischen Programmcodes (Abbildung 4.17) beschrieben werden sollen. Der Code ist gekürzt, unter anderem wurden die Importe der Bibliotheken entfernt.

²vgl. Code im Anhang

4.4 Ergebnisse: Automatisierte Schnittstellen-Erkennung

```
# Einlesen der Daten (Verwendung der Daten aus Kapitel 4.4.1)
df = pd.read_csv("/Users/felix/.../_20202310_Pretraining_test.csv")

# Aufteilung der Datenmenge in Trainings- und Testdatenmenge
texts = df.text.tolist()
labels = df.label.tolist()
X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.25, random_state=42)

# Der CountVectorizer führt diverse Vorverarbeitungsschritte durch. Er wird verwendet zur Erzeugung
# einer angemessenen Repräsentation der Texte (insbesondere Bildung von N-Grammen)
count_vect = CountVectorizer(ngram_range=(1, 2))

# Extraktion von Features aus den Texten ("term-document-matrix")
X_train_counts = count_vect.fit_transform(X_train)

# Training eines Klassifizierungs-Modells ("Logistic Regression").
lr_clf = LogisticRegression(solver="lbfgs").fit(X=X_train_counts, y=y_train)

# Transformation der Testdaten & Nutzung des Modells für Vorhersagen
X_test_counts = count_vect.transform(X_test)
predicted = lr_clf.predict(X_test_counts)

# Evaluation der Ergebnisse, wie im vorangegangenen Abschnitt
print(metrics.classification_report(y_test, predicted))
```

Abbildung 4.17: Klassifizierung mit logistischer Regression. Eigene Erstellung

Kriterium 1: Einsetzbarkeit des Ansatzes unter den gegebenen Rahmenbedingungen

Es werden nachfolgend in 4.18 die Ergebnisse des Versuchs aus 4.17 präsentiert.

	precision	recall	f1-score	support
0	0.88	0.92	0.90	71
1	0.94	0.91	0.93	105
accuracy			0.91	176
macro avg	0.91	0.91	0.91	176
weighted avg	0.92	0.91	0.92	176

Abbildung 4.18: Ergebnisse der Klassifizierung von Sätzen aus der Testmenge. Eigene Bildschirmaufnahme

Diese zunächst gutaussehenden Ergebnisse können wohl durch Overfitting erklärt werden. Dieser Umstand ist der geringen Datenmenge geschuldet. Ein Versuch mit einem komplett anderen Dokument (vgl. 4.19) etwa klassifiziert 153 der insgesamt 494 Sätze aus diesem Dokument in die TI-Klasse (entspricht Klasse 1 in 4.19; es geht um das gleiche Dokument bzw. das gleiche Vorgehen wie in 4.4.1). Zu sehen ist, dass Sätze in Klasse 0 ("kein technisches Interface") tatsächlich häufig korrekt klassifiziert wurden. Genauso häufig werden aber auch Sätze, die offensichtlich nichts mit Interfaces zu tun haben, in Klasse 1 (technisches Interface) klassifiziert, was nicht korrekt ist.

4.4 Ergebnisse: Automatisierte Schnittstellen-Erkennung

Index	sentence	prediction
287	Therefore, alternative journey planners and how to allowing users to implement an alternative front-end to the MyWay services have been analysed.	0
288	Table 16.	0
289	Requirements for the Interoperability ID Requirement Level Catalonia Berlin Trikala Importance NF-IN-001 Architecture allows using a third-party journey pla...	1
290	609023 ID Requirement Level Catalonia Berlin Trikala Importance NF-IN-010 MyWay APP is adapted to tablets and smartphones screen size requirement High Low L...	1
291	Table 17.	0
292	Requirements for Extensibility/Maintainability ID Requirement Level Catalonia Berlin Trikala Importance NF-EM-001 Architecture of MyWay is flexible enough t...	1
293	609023 5.2.4 Privacy This section summarizes the non-functional requirements related to data privacy, security and traceability of actions performed by the ...	1
294	Table 18.	0
295	Requirements related with the Privacy ID Requirement Level Catalonia Berlin Trikala Importance NF-PR-001 Where practical, user data is anonymized requiremen...	1
296	Communications with sensitive data are encrypted requirement High High High Must NF-PR-004 MyWay provides a traceability of the sensible actions performed b...	1
297	609023 5.2.5 Standards Compliance This section summarizes the non-functional requirements related to the compliance of the MyWay to the source data standard...	1
298	Table 19.	0
299	Requirements for the Standards Compliance ID Requirement Level Catalonia Berlin Trikala Importance NF-SC-001 Where practical, data and Service Model should ...	1
300	609023 5.2.6 Data Quality: Availability and Reliability This section summarizes the non-functional requirements related to the quality and consistency of da...	1
301	Table 20.	0
302	Requirements for the Data Quality ID Requirement Level Catalonia Berlin Trikala Importance NF-DQ-001 Minimum level of data quality is ensured in order to ma...	1
303	609023 6.	0
304	SELECTED USE CASES FOR KEY FUNCTIONALITY In this chapter, selected detailed use-cases capturing MyWay key functionality from the public end user's point of ...	0
305	During the detailed system design, relevant details of the functionality will be adjusted further.	0
306	This chapter keeps the same structure as the requirements overview (see chapter 'Requirements').	0
307	For the 'System management' part, the detailed use-cases are not provided in this stage, as this functionality has to be elaborated further in detail during...	1
308	6.1 Notation of the detailed use cases For interaction overview between actors and particular detailed use cases, standard UML Use Case diagram is being use...	1
309	For the description of the particular use cases (depicted as ovals linked to the Actor picture in the Use Case diagrams), in accordance to the common softwa...	1
310	609023 GOTO STEP – go to a particular step in the same use case GOTO USE CASE – go to a particular step in another use case.	0
311	Each step is specified CONSEQUENT USE CASE – definition of the use case, which is logically following the particular use case.	0

Abbildung 4.19: Einige Vorhersagen für ein Evaluations-Dokument. Eigene Bildschirmaufnahme

Deshalb kann die produktive Einsetzbarkeit des Verfahrens unter den gegebenen Rahmenbedingungen mit relativ wenigen Datensätzen für das Training eines Modells nicht bestätigt werden.

Kriterium 2: Generelle Einsetzbarkeit des Ansatzes im APM-Kontext, unter der Annahme einer beliebig großen Datenmenge

Hier [70] wurde eine ähnliche Klassifikation durchgeführt. Die Autoren verwendeten 89 Text-Dokumente für das Training des Klassifikationsmodells. Erreicht wurde, dass 4 von 5 Test-Dokumenten korrekt klassifiziert wurden – ein grundsätzlich gutes Ergebnis, erreicht mit einer kleinen Trainingsmenge. Es kann jedoch gesagt werden, dass bei steigenden Trainingsdatenmengen recht zuverlässige Ergebnisse erzielt werden können, was auch für den vorliegenden Kontext gelten würde. Damit kommt das Verfahren bei ausreichender Trainingsdatenmenge durchaus für den Einsatz in Frage.

Kriterium 3: Rechenaufwand

Die Vorverarbeitung, sowie das Training des Modells erfolgen in wenigen Sekunden. Die Anwendung zur Vorhersage von neuen Datensätzen erfolgt ebenfalls in wenigen Sekunden.

4.5 Zusammenfassung der Ergebnisse

Kriterium 4: Kosten

scikit-learn kann kostenfrei genutzt werden [55]. Zur Verbesserung und Weiterentwicklung des Modells müssen weitere Sätze beschriftet und weitere Dokumente gesammelt werden. Das erzeugt manuellen Aufwand.

4.5 Zusammenfassung der Ergebnisse

Dieser Abschnitt stellt die Ergebnisse der Evaluation zusammenfassend in einer Übersicht dar. Des Weiteren enthält 4.3 eine Handlungsempfehlung zum weiteren Vorgehen mit den Verfahren.

Kategorie	Ansatz	Handlungsempfehlung/ Zusammenfassung
Ermittlung von Ähnlichkeiten	Mit einem (vortrainierten) Word2Vec-Modell	Gute Grundlage. Weiterentwicklung kann sinnvoll sein.
	Unter Verwendung des Paragraph-Vector-Modells (Doc2Vec)	Kann bei ausreichenden Datenmengen sinnvoll sein. Weiterentwickeln.
Bestimmung von inhaltlicher Integrität	Durch Erzeugung und den Vergleich von Durchschnitts-Wort-Vektoren	Ist höchstens sinnvoll, um absolut unpassende Texte ("lorem ipsum") zu entdecken.
	Mit Topic Modeling (NMF)	Gute Grundlage. Weiterentwicklung kann sinnvoll sein.
Automatisierte Erzeugung von Zusammenfassungen	Mit einem extraktiven Ansatz (TF-IDF)	Nicht sinnvoll. Idee verwerfen.
	Mit einem abstraktiven Ansatz (T5)	Noch nicht sinnvoll einsetzbar. Weiter beobachten
Automatisierte Schnittstellen-erkennung	Per Klassifizierung von Sätzen mit BERT und einem neuronalen Netz	Weiter beobachten. Weitere Trainingsdaten anlegen und weiterentwickeln.
	Per Klassifizierung von Sätzen mit logistischer Regression	Weiterentwickeln, insbesondere um eine Referenz zu anderen Ansätzen zu erhalten.

Tabelle 4.3: Zusammenfassung der Evaluations-Ergebnisse

5 Fazit und Ausblick

Ausgangsbasis dieser Arbeit war die Annahme, dass Entscheidungen zum Application-Portfolio-Management meist auf Expertenmeinungen und manuellen Prozessen basieren, also sowohl papiergetrieben als auch unterstützt durch diverse Architektur-Tools getroffen werden. Grundlage für die Entscheidungen können Dokumentationen sein, die in Textform vorliegen. In den Dokumentationen können wertvolle Informationen enthalten sein. Die Extraktion der Informationen bzw. die Auswertung der Dokumente durch Menschen ist aber schwierig und zeitaufwändig.

Die vorliegende Arbeit versuchte daher eine Verbindung herzustellen zwischen den beiden Domänen IT-Governance und Maschinelles Lernen. Untersucht werden sollte welche ML-Verfahren für das APM von Bedeutung sein können. Es wurden datengestützte, praktische Verfahren auf ein dem IT-Management zuzuschreibendes Thema angewendet und evaluiert. IT-Governance wird dabei als Teilgebiet des IT-Managements betrachtet [53, S. 6]. Bevor die Evaluation von ML-Verfahren überhaupt durchgeführt werden konnte, mussten verschiedene Verfahren in einem vorhergehenden Schritt ausgewählt werden und von nicht-relevanten Verfahren getrennt werden. Zusätzlich war zu klären, mit welchen Fragestellungen sich das APM überhaupt beschäftigt.

Die Verfahren wurden alle in Python implementiert, da in der Sprache viele Bibliotheken für maschinelles Lernen zur Verfügung stehen. Diese Bibliotheken werden in vielen Tutorials und Beispielen verwendet. Die sorgfältige Aufbereitung bzw. das ordentliche Einlesen von Texten wurde in der praktischen Tätigkeit als Voraussetzung für den späteren Erfolg des Einsatzes von Machine-Learning-Verfahren identifiziert. Wenn etwa Texte nur zur Hälfte verarbeitet werden, Aufzählungen oder Tabellen ignoriert werden, oder statische Bestandteile wie das Datum der Veröffentlichung oder Seitennummern fälschlicherweise in die Verarbeitung aufgenommen werden, dann kann sich die Vorhersage eines ML-Verfahrens aufgrund dieser Fehler später als nicht zuverlässig herausstellen. Das bekannte Pareto-Prinzip kann hier Anwendung finden, indem von 80% des Aufwands für Vorverarbeitung von Texten und lediglich 20% für den Einsatz bzw. die Evaluation von ML-Verfahren ausgegangen werden kann. Die Trainingsdaten sollten außerdem die Daten, die später verarbeitet werden, möglichst gut repräsentieren.

5 Fazit und Ausblick

Als ungelöstes Problem verbleibt insbesondere die Extraktion von Informationen aus Bildern. Hierbei handelt es sich um eine andere Domäne als NLP. Auch Bilder können Träger von wichtigen Informationen sein, weshalb hier noch Potenzial zu Verbesserungen liegt. Wenn ausschließlich Texte betrachtet würden, dann gingen für das APM wichtige Informationen verloren.

Überwachte Verfahren des maschinellen Lernens sind problematisch für eine sehr spezielle Domäne wie das APM, da zunächst viel Aufwand in das Beschriften von Beispiel-Dokumenten/-Abschnitten/ und/ oder -Sätzen investiert werden muss. Die Beschriftungen sollten von Experten durchgeführt werden. Es müssen Beschriftungen in ausreichender Menge gesammelt werden, da bei unbekannten neuen Dokumenten die Schätzungen sonst ungenaue Ergebnisse liefern. Unüberwachte Ansätze, wie bei der Verwendung von vortrainierten Modellen oder beim Topic-Modeling, benötigen keine Beschriftungen. Sie können daher als Alternative in Betracht gezogen werden, wenn der Aufwand für überwachte Verfahren als zu hoch eingeschätzt wird.

Fast alle der untersuchten Anwendungsfälle können zu Klassifizierungsproblemen reduziert werden. Als Eingabe kommen verschiedene Ebenen in Frage: mehrere Dokumente auf einmal, einzelne Dokumente, einzelne Abschnitte oder auch einzelne Sätze. Die Verfahren, die hier evaluiert wurden, können demnach auch für andere Anwendungsfälle angepasst werden.

TF-IDF wurde als universelles Werkzeug zur Vorverarbeitung von Texten für verschiedene Anwendungsfälle identifiziert. Es ermöglicht die Bestimmung der Wichtigkeit einzelner Begriffe und kann daher für quasi alle Anwendungsfälle, die in dieser Arbeit beschrieben wurden, sinnvoll genutzt werden. Für vektorbasierte Ansätze gilt eine ähnliche Bedeutung. Auch diese Art der Vorverarbeitung ist unabhängig von Anwendungsfällen und sollte in Betracht gezogen, wenn mit Text gearbeitet werden soll. Transformer-Ansätze werden aktiv erforscht und sind daher für die Zukunft im Auge zu behalten. Aktuelle Transformer-Ansätze wie BERT oder GPT-3 sollen hier als vielversprechend im Bereich der künstlichen Intelligenz hervorgehoben werden.

Literatur

- [1] M. Allahyari, S. Pouriyeh, M. Assefi & al. *Text Summarization Techniques: A Brief Survey*. 2017.
- [2] M. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, M. Hasan, B. Van Essen, A. Awwal & V. Asari. *A State-of-the-Art Survey on Deep Learning Theory and Architectures*. Electronics, 2019.
- [3] M. Anandarajan, C. Hill & T. Nolan. *Practical Text Analytics: Maximizing the Value of Text Data*. Springer, 2019.
- [4] *Application Portfolio Management*. The Art of Service - Application Portfolio Management Publishing, 2019.
- [5] D. Avison, F. Lau, M. Myers & P. A. Nielsen. *Action Research*. Communications of the ACM, 1999.
- [6] I. Beltagy, M. E. Peters & A. Cohan. *Longformer: The Long-Document Transformer*. 2020.
- [7] BITKOM. *Enterprise Architecture Management – neue Disziplin für die ganzheitliche Unternehmensentwicklung*. 2011.
- [8] T. B. Brown, B. Mann, N. Ryder & al. *Language Models are Few-Shot Learners*. 2020.
- [9] A. M. Dai, C. Olah & Q. V. Le. *Document Embedding with Paragraph Vectors*. 2015.
- [10] H. T. Dang. *Overview of DUC 2005*. Proceedings of the Document Understanding Conf. Wksp. 2005 (DUC 2005), 2005.
- [11] scikit-learn developers. *sklearn.metrics.f1-score*. Abgerufen am 20. November 2020. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.
- [12] J. Devlin, M.-W. Chang, K. Lee & K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019.
- [13] N. Dilawar, H. Majeed, M. O. Beg & al. *Understanding Citizen Issues through Reviews: A Step towards Data Informed Planning in Smart Cities*. Applied Sciences 8(9):1589, 2018.
- [14] W. Ertel. *Introduction to Artificial Intelligence*. Springer, 2017.
- [15] fastText. *Library for efficient text classification and representation learning*. Facebook Inc. Abgerufen am 15. August 2020. URL: <https://fasttext.cc/>.
- [16] T. Frey. *Governance Arrangements for IT Project Portfolio Management*. Springer Gabler, 2014.

- [17] A. Gadatsch & E. Mayer. *Masterkurs IT-Controlling*. Springer Vieweg, 2014.
- [18] M. Gaulke. *Praxiswissen COBIT. Grundlagen und praktische Anwendung in der Unternehmens-IT*. dpunkt.verlag, 2020.
- [19] gensim. Python Software Foundation. Abgerufen am 16. Dezember 2020. URL: <https://pypi.org/project/gensim/>.
- [20] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow*. O'Reilly, 2019.
- [21] C. Gliedman. *Defining IT Portfolio Management*. Forrester Research, 2004.
- [22] O. Grisel, L. Buitinck & C.-K. Yau. *Topic extraction with Non-negative Matrix Factorization and Latent Dirichlet Allocation*. Abgerufen am 20. November 2020. URL: https://scikit-learn.org/0.18/auto_examples/applications/topics_extraction_with_nmf_lda.html.
- [23] huggingface. *Longformer*. Abgerufen am 28. Oktober 2020. URL: https://huggingface.co/transformers/model_doc/longformer.html.
- [24] huggingface. *Model: bert-base-uncased*. 03. Dezember 2020. URL: <https://huggingface.co/bert-base-uncased>.
- [25] huggingface. *Model: t5-small*. Abgerufen am 30. Oktober 2020. URL: <https://huggingface.co/t5-small>.
- [26] huggingface. *Transformers*. Abgerufen am 13. Oktober 2020. URL: <https://huggingface.co/transformers/>.
- [27] IMPACT. *IT Governance. Developing a successful governance strategy. A Best Practice Guide for decision makers in IT*. The National Computing Centre, 2005.
- [28] B. Inmon. *Turning Text into Gold: Taxonomies and Textual Analytics*. Technics Publications, 2016.
- [29] *Integration Architecture*. LeanIX GmbH. Abgerufen am 13. November 2020. URL: <https://docs.leanix.net/docs/integration-architecture>.
- [30] ISACA. *COBIT 2019 Framework Introduction and Methodology*. 2018.
- [31] ISACA. *COBIT 2019 Framework. Governance and Management Objectives*. 2018.
- [32] J. Jung. *Purpose of Enterprise Architecture Management: Investigating Tangible Benefits in the German Logistics Industry*. 2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW), 2019.
- [33] J. Jung & R. Schlör. *Analysis Using the Business Support Matrix: Elaborating Potential for Improving Application Landscapes in Logistics*. 2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop, 2018.
- [34] J. D. Kelleher. *Deep Learning*. The MIT Press, 2019.
- [35] Q. Le & T. Mikolov. *Distributed Representations of Sentences and Documents*. 2014.

- [36] Y. LeCun, L. Bottou, Y. Bengio & P. Haner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 1998.
- [37] C.-Y. Lin. *ROUGE: A Package for Automatic Evaluation of Summaries*. Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), 2004.
- [38] H. P. Luhn. *The Automatic Creation of Literature Abstracts*. IBM Journal of Research und Development, 1958.
- [39] L. J. P. van der Maaten & G. E. Hinton. *Visualizing High-Dimensional Data Using t-SNE*. Journal of Machine Learning Research 9(Nov):2579-2605, 2008.
- [40] C. D. Manning, P. Raghavan & H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [41] M. Mayo. *The Main Approaches to Natural Language Processing Tasks*. Abgerufen am 20. Juli 2020. URL: <https://www.kdnuggets.com/2018/10/main-approaches-natural-language-processing-tasks.html>.
- [42] T. Mikolov, K. Chen, G. Corrado & J. Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013.
- [43] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator & W. R. Swartout. *Enabling Technology for Knowledge Sharing*. AI Magazine Volume 12 Number 3, 1991.
- [44] M. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.
- [45] *numpy*. Python Software Foundation. Abgerufen am 17. Dezember 2020. URL: <https://pypi.org/project/numpy/>.
- [46] D. O'Connel. *Integrated Portfolio Management: Better Visibility, Easier Decisions, Lower Costs*. Aite Group/ Software AG, 2020.
- [47] J. Pennington, R. Socher & C. D. Manning. *GloVe: Global Vectors for Word Representation*. Abgerufen am 19. Dezember 2020. URL: <https://nlp.stanford.edu/projects/glove/>.
- [48] J. M. C. Pérez. *Augmenting Humans*. TU München (Dissertation), 2017.
- [49] C. Raffel, N. Shazeer, A. Roberts & al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2020.
- [50] P. C. Rao, A. Reedy & B. Bellman. *Certified Enterprise Architect*. McGraw-Hill Education, 2019.
- [51] R. Řehůřek. *Doc2vec paragraph embeddings*. Abgerufen am 05. August 2020. URL: <https://radimrehurek.com/gensim/models/doc2vec.html>.
- [52] R. Řehůřek. *Word2Vec Model*. Abgerufen am 08. Oktober 2020. URL: https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html#sphx-glr-auto-examples-tutorials-run-word2vec-py.
- [53] M. Reiss. *Dokumentationsmanagement – Basis für IT-Governance*. Springer Vieweg, 2018.

- [54] A. L. Samuel. *Some studies in machine learning using the game of checkers*. IBM Journal of research und development, 3(3), 210-229., 1959.
- [55] *scikit-learn*. Python Software Foundation. Abgerufen am 16. Dezember 2020. URL: <https://pypi.org/project/scikit-learn/>.
- [56] *ServiceNow Application Portfolio Management*. Datasheet. ServiceNow, Inc., 2020.
- [57] R. Silipo & V. Tursi. *From Words to Wisdom*. KNIME Press, 2018.
- [58] D. Simon, K. Fischbach & D. Schoder. *Application Portfolio Management - An Integrated Framework and a Software Tool Evaluation Approach*. Communications of the Association for Information Systems: Vol. 26 , Article 3, 2010.
- [59] spacy.io. *Doc (Class)*. Abgerufen am 01. September 2020. URL: <https://spacy.io/api/doc>.
- [60] spacy.io. *English Available pretrained statistical models for English*. Abgerufen am 16. Dezember 2020. URL: <https://spacy.io/models/en>.
- [61] spacy.io. *Rule-based matching*. Abgerufen am 02. August 2020. URL: <https://spacy.io/usage/rule-based-matching>.
- [62] spacy.io. *Word Vectors and Semantic Similarity*. Abgerufen am 02. August 2020. URL: <https://spacy.io/usage/vectors-similarity>.
- [63] Statista. *Statista-Expertenbefragung BVL & Statista Logistikmonitor 2018*. Abgerufen am 20. Juli 2020. URL: <https://de.statista.com/prognosen/943357/expertenbefragung-zur-kuenstlichen-intelligenz-in-der-logistikbranche>.
- [64] M. Tang, P. Gandhi & M. A. Kabir. *Progress Notes Classification and Keyword Extraction using Attention based Deep Learning Models with BERT*. 2019.
- [65] *The Definitive Guide to Application Portfolio Management*. LeanIX GmbH.
- [66] *The MIT License*. Open Source Initiative. Abgerufen am 16. Dezember 2020. URL: <https://opensource.org/licenses/MIT>.
- [67] *The TOGAF Standard 9.2*. The Open Group. Abgerufen am 19. Dezember 2020. URL: <https://pubs.opengroup.org/architecture/togaf92-doc/arch/index.html>.
- [68] S. Vajjala, B. Majumder, A. Gupta & H. Surana. *Practical Natural Language Processing*. O'Reilly, 2019.
- [69] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser & I. Polosukhin. *Attention Is All You Need*. 2017.
- [70] A. Vogelsang & J. Winkler. *Automatic Classification of Requirements Based on Convolutional Neural Networks*. 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), 2016.
- [71] L. White, R. Togneri, W. Liu & M. Bennamoun. *Neural Representations of Natural Language*. Springer, 2018.

- [72] R. Yampolskiy. *Turing Test as a Defining Feature of AI-Completeness*. In: Yang X.-S. (Herausgeber). Artificial Intelligence, Evolutionary Computing und Metaheuristics. Studies in Computational Intelligence, vol 427. Springer, 2013.

Eidesstattliche Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen sind, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher Form noch keiner anderen Prüfbehörde vorgelegen.

Frankfurt am Main, im Dezember 2020

Anhang

Der Python-Code, der für die in dieser Arbeit beschriebenen Evaluationen verwendet wurde, ist in einem Zip-Archiv zu finden, das mit der Ausarbeitung abgegeben wird.