Karlsruhe Institute of Technology

Identifikation von Rückverfolgbarkeitsverbindungen zwischen Anforderungen mittels Sprachmodellen

Bachelorarbeit
von

Niklas Ewald

An der Fakultät für Informatik
Institut für Programmstrukturen
und Datenorganisation (IPD)

Erstgutachter:	Prof. Dr. Walter F. Tichy
Zweitgutachter:	Prof. Dr. Anne Koziolek
Betreuender Mitarbeiter:	M.Sc. Tobias Hey

Bearbeitungszeit: 23.12.2020 – 23.04.2021

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Die Regeln zur Sicherung guter wissenschaftlicher Praxis im Karlsruher Institut für Technologie (KIT) habe ich befolgt.

Karlsruhe, 23.04.2021

(**Niklas Ewald**)

Publikationsgenehmigung

Melder der Publikation

Hildegard Sauer

Institut für Programmstrukturen und Datenorganisation (IPD)
Lehrstuhl für Programmiersysteme
Leiter Prof. Dr. Walter F. Tichy

+49 721 608-43934
hildegard.sauer@kit.edu

Erklärung des Verfassers

Ich räume dem Karlsruher Institut für Technologie (KIT) dauerhaft ein einfaches Nutzungsrecht für die Bereitstellung einer elektronischen Fassung meiner Publikation auf dem zentralen Dokumentenserver des KIT ein.

Ich bin Inhaber aller Rechte an dem Werk; Ansprüche Dritter sind davon nicht berührt.

Bei etwaigen Forderungen Dritter stelle ich das KIT frei.

Eventuelle Mitautoren sind mit diesen Regelungen einverstanden.

Der Betreuer der Arbeit ist mit der Veröffentlichung einverstanden.

Art der Abschlussarbeit:	Bachelorarbeit
Titel:	Identifikation von Rückverfolgbarkeitsverbindungen zwischen Anforderungen mittels Sprachmodellen
Datum:	23.04.2021
Name:	Niklas Ewald

Karlsruhe, 23.04.2021

(**Niklas Ewald**)

Kurzfassung

Die Rückverfolgbarkeit zwischen Anforderungen ist ein wichtiger Teil der Softwareentwicklung. Zusammenhänge werden dokumentiert und können für Aufgaben wie Auswirkungs- oder Abdeckungsanalysen verwendet werden. Da das Identifizieren von Rückverfolgbarkeitsverbindungen von Hand zeitaufwändig und fehleranfällig ist, ist es hilfreich, wenn automatische Verfahren eingesetzt werden können. Anforderungen werden häufig während der Entwicklung verfeinert. Entstehende Anforderungen lassen sich zu den ursprünglichen Anforderungen zurückverfolgen. Die entstehenden Anforderungen befinden sich auf einem anderen Abstraktionslevel. Dies erschwert die automatische Identifizierung von Rückverfolgbarkeitsverbindungen. Auf großen Textkorpora trainierte Sprachmodelle stellen eine mögliche Lösung für dieses Problem dar. In dieser Arbeit wurden drei auf Sprachmodellen basierende Verfahren entwickelt und verglichen: Feinanpassung einer Klassifikationsschicht, Ausnutzen der Ähnlichkeit der jeweiligen Satzeinbettungen und eine Erweiterung des zweiten Verfahrens, bei dem zusätzlich zunächst Cluster gebildet werden. Es wurden sechs öffentlich verfügbare Datensätze verwendet, um die Verfahren zu evaluieren. Von den drei Verfahren erreichen jeweils das Sprachmodell mit Klassifikationsschicht und das Ausnutzen der Ähnlichkeit zwischen Satzeinbettungen für drei Datensätze die besten Ergebnisse, die aber hinter den Ergebnissen von anderen aktuellen Verfahren zurückbleiben. Das feinangepasste Sprachmodell mit Klassifikationsschicht erzielt eine Ausbeute von bis zu 0,96 bei einer eher geringen Präzision von 0,01 bis 0,26.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Rückverfolgbarkeit von Anforderungen ist ein wichtiger Teil der Softwareentwicklung. Sie erlaubt es zu belegen, dass alle Anforderungen eines Auftraggebers umgesetzt wurden. Rückverfolgbarkeitsverbindungen können auch während der Entwicklung eingesetzt werden, um abzuschätzen, welche Auswirkungen eine Änderung auf andere Teile des Projekts hat [CHGZ12]. In einigen Bereichen, wie zum Beispiel der Luftfahrt, wird die Rückverfolgbarkeit zwischen Anforderungen sogar explizit gefordert. Das Identifizieren von Rückverfolgbarkeitverbindungen zwischen Anforderungen von Hand ist zeitaufwändig und fehleranfällig. Anforderungen werden im Laufe eines Projekts verfeinert, so dass eine eher abstrakte, grobe Anforderung durch eine oder mehrere detaillierte Anforderungen abgedeckt wird. Abbildung 1.1 zeigt drei Anforderungen. Es ist zu sehen, dass Anforderung d11_1 einen Teil von Anforderung r11 aufgreift: In der Benutzeroberfläche soll es eine Möglichkeit geben, die Länge einer Aufgabe zu ändern. Während diese Anforderungen etwa den gleichen Abstraktionsgrad haben, gibt es auch Anforderungen, bei denen die unterschiedlichen Abstraktiongrade ein großes Problem für die automatische Identifikation von Rückverfolgbarkeitsverbindungen ist. In dieser Arbeit sollen Sprachmodelle, wie zum Beispiel BERT [DCLT19], verwendet werden, um ein Verfahren zu entwickeln, das automatisch Rückverfolgbarkeitsverbindungen identifizieren kann. Sprachmodelle, die auf großen Textkorpora vortrainiert werden, sollen generelle Eigenschaften von natürlicher Sprache lernen. Durch Training im Kontext eines Satzes werden Informationen über Beziehungen zwischen Wörtern gelernt. Das Verständnis von Sprache und insbesondere das Verständnis von Beziehungen zwischen Wörtern hilft möglicherweise, das Problem der unterschiedlichen Abstraktionsgrade zu lösen, so dass Ähnlichkeiten zwischen Anforderungen entdeckt werden könnten, die bisher nicht von anderen Methoden gefunden werden.

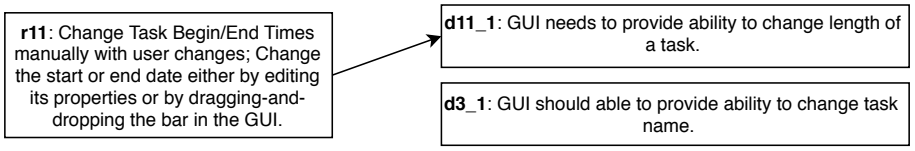

Abbildung 1.1: Beispiel für Anforderungen aus dem GANTT-Datensatz [HHD09] mit Rückverfolgbarkeitsverbindung

1.1 Zielsetzung

Das Ziel dieser Arbeit ist der Entwurf und die Evaluation von Verfahren zur automatischen Identifikation von Rückverfolgbarkeitsverbindungen zwischen natürlichsprachlichen Anforderungen mit unterschiedlichen Abstraktionsgraden. Hierfür werden geeignete Sprachmodelle gewählt. Auf den Sprachmodellen basierend sollen verschiedene Methoden aufgebaut werden, deren Ergebnisse miteinander verglichen werden.

1.2 Aufbau der Arbeit

In Kapitel 2 werden, die zum Verständnis dieser Arbeit benötigten Grundlagen erklärt. In Kapitel 3 werden Arbeiten beschrieben, die Teillösungen für das Problem bereits gelöst haben, oder die das Problem der Identifikation von Rückverfolgbarkeitsverbindungen zwischen Anforderungen mit einer anderen Methode angegangen sind. Die auftretenden (Teil-) Probleme werden in Kapitel 4 genauer untersucht. Basierend auf den Ergebnissen wird ein Entwurf angefertigt. Die konkrete Umsetzung des Entwurfs wird in Kapitel 5 beschrieben. In Kapitel 6 werden die entworfenen Verfahren evaluiert, miteinander und mit existierenden Verfahren verglichen. Abschließend werden in Kapitel 7 die wichtigen Punkte der Arbeit zusammengefasst und es wird ein Ausblick zu möglichen Verbesserungsmöglichkeiten für zukünftige Arbeiten gegeben.

2 Grundlagen

2.1 Anforderungsmanagement

Das Anforderungsmanagement ist ein Bestandteil des Entwicklungsprozesses von Software und umfasst das Identifizieren und Dokumentieren, sowie das Verfolgen und Verwalten von Anforderungen [HHM+04].

2.1.1 Artefakt

Artefakte sind Teile von Software, zwischen denen eine Rückverfolgbarkeitsverbindung identifiziert werden kann [GCHH+12]. Zum Beispiel können einzelne oder gruppierte Anforderungen, UML-Diagramme oder Java-Klassen Artefakte sein. Artefakte können in Quell- und Zielartefakte aufgeteilt werden, um ihren Verwendungszweck beim Identifizieren von Rückverfolgbarkeitsverbindungen zu beschreiben. Die Menge an Artefakten, von denen aus Rückverfolgbarkeitsverbindungen identifiziert werden, sind die Quellartefakte. Ein Artefakt, das untersucht wird, um eine mögliche Rückverfolgbarkeitsverbindung zwischen einem Quellartefakt und ihm zu identifizieren, ist ein Zielartefakt.

2.1.2 Anforderung

Eine Anforderung beschreibt das Verhalten, dass ein (Software-) Projekt erfüllen muss. In dieser Arbeit bezeichnen grobe oder allgemeine Anforderungen Anforderungen, die im Laufe der Entwicklung eines Projektes verfeinert werden. Diese verfeinerten, detaillierteren Anforderungen werden als detaillierte Anforderungen bezeichnet.

2.1.3 Rückverfolgbarkeitsverbindung

Eine Rückverfolgbarkeitsverbindung ist eine Zuordnung zwischen einem Quell- und einem Zielartefakt [GCHH+12]. Rückverfolgbarkeitsverbindungen haben einen Typ, der die Beziehung der Artefakte zueinander beschreibt. Ein Zielartefakt kann ein zugehöriges Quellartefakt zum Beispiel „verfeinern“, „implementieren“ oder „testen“. In dieser Arbeit werden vor allem Rückverfolgbarkeitverbindungen zwischen Anforderungen betrachtet, deren Zielartefakte die jeweiligen Quellartefakte verfeinern.

2.2 Sprachverarbeitung

Einige Methoden der Sprachverarbeitung werdne bei der Identifizierung von Rückverfolgbarkeitsverbindungen zwischen Anforderungen benötigt. Diese werden hier vorgestellt.

2.2.1 Stoppwortentfernung

Wörter, die sehr häufig vorkommen und die kaum Mehrwert für die Bedeutung eines Textes bieten, werden als Stoppwörter bezeichnet [MRS08]. Sie werden manchmal in der Vorverarbeitung von Texten entfernt. Stoppwörter sind oft Funktionswörter, wie zum Beispiel Artikel oder Präpositionen. Eine Liste mit zu entfernenden Wörtern kann erstellt werden, indem die häufigsten Wörter aus den betrachteten Dokumenten verwendet werden.

2.2.2 Lemmatisierung

Texte enthalten in der Regel unterschiedliche grammatikalische Formen eines Wortes. Durch die Lemmatisierung werden diese Formen auf die Grundform (das Lemma) des jeweiligen Wortes abgebildet [MRS08]. Zum Beispiel werden „am“, „are“ und „is“ auf „be“ abgebildet. Nachfolgende Verfahren behandeln so die Wörter, als wären sie dasselbe. Ein Suchalgorithmus kann so zum Beispiel Dokumente finden, die zwar nicht exakt ein Wort der Suchanfrage, aber eine gemeinsame Grundform enthalten.

2.2.3 Tokenisierung

Die Tokenisierung ist ein Verfahren, das einen Text in Tokens zerlegt [MRS08]. Ein Token ist eine Zeichenfolge in einem Dokument, die eine sinnvolle Bedeutung für die Weiterverarbietung hat. Häufig besteht ein Token aus einem Wort, also einer Folge aus Buchstaben, die im Text von Leer- oder Satzzeichen umgeben ist. Die Art der Tokenisierung ist abhängig von der Sprache, in der ein Dokument geschrieben ist. In manchen Sprachen, wie zum Beispiel Chinesisch oder Japanisch, werden keine Leerzeichen verwendet, um einzelne Wörter zu trennen. Im Deutschen werden zusammengesetzte Nomen nicht getrennt. Es gibt auch Zeichenfolgen, die als ein einzelnes Token betrachtet werden sollten, obwohl sie durch ein Leerzeichen getrennt sind. Oft ist dies der Fall bei Eigennamen (zum Beispiel: „New York“) oder Datumsangaben („2nd December 2016“).

2.3 Information-Retrieval

Manning et al. geben in ihrem Buch „Introduction to Information Retrieval“ [MRS08] für Information Retrieval folgende Definition: „Information Retrieval is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)“. Information Retrieval-Verfahren sind Verfahren, die in einer Menge an Dokumenten diejenigen finden, die ein bestimmtes Informationsbedürfnis erfüllen.

2.4 Maschinelles Lernen

Maschinelles Lernen ist ein Teilgebiet der künstlichen Intelligenz. Mitchell beschreibt es als Gebiet, das sich mit dem Erstellen von Computerprogrammen befasst, die sich automatisch durch Erfahrung verbessern [Mit97]. Das Ziel ist es also, Algorithmen zu entwerfen, die mithilfe von vorhandenen Trainingsdaten automatisch so angepasst werden, dass sie ihre Leistung für eine vorgesehene Aufgabe verbessern.
Es wird häufig zwischen überwachtem und unüberwachtem maschinellem Lernen unterschieden. Beim überwachten Lernen erhält der Algorithmus Beispiele, die von Hand Klassen zugeordnet wurden. Anhand dieser Beispiele sollen Strukturen gelernt werden, so dass für ungesehene Beispiele vorhergesagt werden kann, welcher Klasse sie zugeordnet werden. Beim unüberwachten Lernen muss ein Algorithmus Gemeinsamkeiten zwischen den Trainingsdaten finden [MAP18].
Ein künstliches neuronales Netz ist eine Methode des maschinellen Lernens, die vom

menschlichen Gehirn inspiriert wurde. Haykin beschreibt neuronale Netze als Prozessoren/Verarbeiter, die aus einfachen Verarbeitungseinheiten bestehen und eine Neigung dazu haben, Wissen zu speichern [Hay99]. Die Verarbietungseinheiten werden künstliches Neuron genannt.
Ein Neuron erhält gewichtete Eingaben, entweder von anderen Neuronen oder als Eingabe von außerhalb des Netzes, fasst diese zusammen und berechnet daraus eine Aktivität. Um die Aktivität zu berechnen, werden zum Beispiel Schwellwertfunktionen, die ab einem bestimmten Wert zum Beispiel 1 und sonst 0 ausgibt, oder eine Gleichrichter-Funktion, die 0 für negative Eingaben oder die Identität für positive Eingaben ausgibt. Die Aktivität wird verwendet um mit einer Ausgabefunktion die Ausgabe des Neurons zu berechnen. Als Ausgabefunktionen wird häufig die Identität verwendet, so dass einfach die Aktivität weitergegeben wird [LC20].
Abhängig davon, wie Neuronen verbunden werden, ergeben sich unterschiedliche Architekturen. Es sind zum Beispiel Netze möglich, in denen ein Neuron mit einem Neuron der übernächsten oder einer vorherigen Schicht verbunden ist [LC20]. Um ein neuronales Netz zu trainieren, können die Gewichte der Verbindungen verändert werden. Ein möglicher Algorithmus zum Bestimmen der neuen Gewichte ist der Fehlerrückmelde-Algorithmus (engl. Backpropagation). Dafür erhält das Netz eine Eingabe, zu der eine erwartete Ausgabe existiert. Die Ausgabe des Netzes wird mit der erwarteten Ausgabe verglichen und die Gewichte werden so angepasst, dass die Ausgabe des Netzes näher an der erwarteten ist.

3 Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten beschrieben. Zunächst werden Datensätze vorgestellt, die in dieser und in verwandten Arbeiten verwendet werden. Der zweite Abschnitt gibt einen Überblick über Ansätze, um das Problem der Identifikation von Rückverfolgbarkeitsverbindungen zwischen Anforderungen zu lösen. Im dritten Abschnitt werden Arbeiten vorgestellt, die verwandte Problemstellungen mit Sprachmodellen lösen.

3.1 Datensätze

Eigenschaften von in dieser und in einigen verwandten Arbeiten verwendeten Datensätze sind in Tabelle 3.1 zu sehen. Detaillierte Anforderungen können zwar Rückverfolgbarkeitverbindungen zu mehreren groben Anforderungen haben, in der Praxis ist es allerdings häufig nur eine.

3.2 Identifikation von Rückverfolgbarkeitsverbindungen

In „Improving Requirements Tracing via Information Retrieval“ [HDO03] fassen Hayes et al. das Problem Rückverfolgbarkeitsverbindungen zu generieren als Information Retrieval Problem auf. Sie implementieren drei Information Retrieval Methoden. Darunter ist eine einfache Information Retrieval Methode, die auf einem generischen Information Retrieval System aufbaut und tf-idf Werte zur Gewichtung verwendet. Die zweite Methode verwendet Schlüsselbegriffe, so dass technische Begriffe aus einer zuvor erstellten Liste wie normale Begriffe behandelt werden. Zum Beispiel enthalten zwei Anforderungen die Begriffe „ecs production environment“ und „ecs archive metadata“, die mit dem Standardmodell für „ecs“ einen Treffer generieren würden. Bei der dritten Methode wird ein Thesaurus in

Tabelle 3.1: Eigenschaften der verwendeten Datensätze

Datensatz	Grob	Detailliert	Verbindungen	mögliche Paare
GANTT [HHD09]	17	69	68	1173
CCHIT [SCH12]	116	1064	587	123424
CM-1 [SHDH10]	235	220	361	51700
InfusionPump [HPL19]	21	126	131	2646
WARC [HPL19]	42	89	136	3738
Modis [SHDH10]	19	49	41	931

Form von Paaren von Wörtern mit ähnlicher Bedeutung gebildet. Die Wortpaare werden von Hand mit einem Ähnlichkeitswert versehen, der in der Berechnung der Ähnlichkeit der Anforderungen verwendet wird. Zur Evaluation wurden zwei Datensätze aus NASA Moderate Resolution Imaging Spectroradiometer Dokumenten erstellt. Der erste enthält 19 Quell- und 50 Zielartefakte (19x50) und der andere enthält eine Teilmenge mit 10 Quell- und 10 Zielartefakten (10x10). Die erste Methode wird mit den Ergebnissen von zwei Analytikern auf dem 10x10 Datensatz verglichen. Die Ausbeute auf dem 10x10 Datensatz ist mit 23,0 % identisch zur Ausbeute des ersten Analytikers, die Präzision ist mit 17,6 % 2,6 Prozentpunkte höher. Der zweite Analytiker hatte mit einer Ausbeute von 42,9 % und einer Präzision von 30,0 % bessere Werte als die einfache Methode. Für den 19x50 Datensatz ist die Ausbeute mit 25,4 % ein bisschen höher, aber die Präzision mit 11,4 % niedriger als die Ergebnisse für den 10x10 Datensatz. Die zweite Methode hat auf dem 10x10 Datensatz mit 27,3 % eine leicht erhöhte Ausbeute, aber mit 5,2 % ein schlechtere Präzision als die erste Methode. Die dritte Methode wird auf dem 19x50 Datensatz evaluiert und mit einem von Hand erstellten Ergebnis und dem Werkzeug SuperTracePlus verglichen. Die Information Retrieval Methode erreicht höhere Werte für die Ausbeute und niedrigere Werte für die Präzision.

Hayes et al. untersuchen in „Advancing Candidate Link Feneration for Requirements Tracing: The Study of Methods“ [HDS06], welche Ziele ein Werkzeug, das Analytikern beim Identifizieren von Rückverfolgbarkeitsverbindungen hilft, erfüllen sollte, und stellen ihr Werkzeug RETRO (REquirements TRacing On-target) vor. Ein Werkzeug sollte glaubwürdig sein. Die Autoren setzen Glaubwürdigkeit aus Genauigkeit, Skalierbarkeit und Nutzbarkeit zusammen. Das zweite Ziel ist Unterscheidbarkeit. Einem Analytiker sollen Kandidaten für Rückverfolgbarkeitsverbindungen und ihre Ähnlichkeitswerte so angezeigt werden, dass es leicht ist, echte Verbindungen von falschen zu unterschieden. Ein Werkzeug soll außerdem erträglich sein. Das bedeutet, es soll Verbindungskandidaten generieren und basierend auf Rückmeldungen des Analytikers Verbindungskandidaten neu generieren, ohne dass dies mühsam ist. RETRO verwendet verschiedene Information Retrieval Methoden, um Rückverfolgbarkeitsverbindungen zwischen Anforderungen zu identifizieren. Neben der einfachen Methode mit Tf-Idf Werten und der Methode mit einem einfachen Thesaurus, wie die Autoren sie in „Improving Requirements Tracing via Information Retrieval“ [HDO03] beschreiben, wird Latent Semantic Indexing (LSI) verwendet. Rückmeldungen eines Analytikers darüber, welche Verbindungskandidaten echte Rückverfolgbarkeitsverbindungen sind und welche nicht, werden dazu verwendet den Anfragevektor anzupassen. Anschließend werden neue Verbindungskandidaten berechnet. Dieser Vorgang wird wiederholt, bis der Analytiker mit dem Ergebnis zufrieden ist. Um RETRO zu evaluieren, simulieren Hayes et al. einen idealen Analytiker, der jeweils den ersten, die ersten zwei, die ersten drei oder die ersten vier noch nicht betrachteten Kandidaten überprüft und bestimmt, ob es sich um eine korrekte Rückverfolgbarkeitsverbindung handelt oder nicht. Getestet werden TF-IDF, TF-IDF+Thesaurus, LSI und LSI+Thesaurus mit je acht Iterationen auf dem MODIS und dem CM-1 Datensatz. Jede Iteration werden gefilterte Listen gespeichert, die die Kandidaten enthalten, deren Relevanzwert höher als 0,05, 0,1, 0,15, 0,2 beziehungsweise 0,25 war. Für den MODIS-Datensatz erreicht TF-IDF+Thesaurus die beste Kombination von Ausbeute und Präzision mit den Filtern 0,1 und 0,15. Für den CM-1-Datensatz erreichen TF-IDF und LSI eine ähnliche Ausbeute. Die beste Präzision von LSI liegt nur bei 0,38 gegenüber 0,55, wie es TF-IDF erreicht. Auf dem MODIS-Datensatz beginnt TF-IDF mit einem F_2-Maß von 20-30 % und verbessert sich auf 60-70 % für alle Filter ungleich 0. TF-IDF+Thesaurus startet bei 35-40 Prozent und endet mit 78-86 %. Auf dem CM-1-Datensatz gibt es für TF-IDF für die Filter 0, 0,05 und 0,1 kaum Änderungen, aber für die Filter 0,15 und 0,2 steigt der Wert von 30-40 % auf über 50 %.

Zou et al. untersuchen in „Improving automated requirements trace retrieval: a study of

term-based enhancement methods“ [ZSCH10] drei auf Begriffen basierende Methoden um die Identifizierung von Rückverfolgbarkeitsverbindungen mit einem Wahrscheinlichkeitsnetzwerk zu verbessern. Dieses Modell berechnet die Relevanz zwischen zwei Artefakten als bedingte Wahrscheinlichkeit, definiert als Funktion der Häufigkeit von Begriffen, die in beiden Artefakten vorkommen. Suchbegriffabdeckung erhöht die berechnete Relevanz zwischen zwei Artefakten, die zwei oder mehr verschiedene Begriffe gemeinsam haben. Die Erhöhung ist proportional zur Anzahl der verschiedenen Begriffe. Die zweite Methode verbessert die Präzision, indem statt einzelner Wörter auch Ausdrücke betrachtet werden. Zum Beispiel soll „section“ in „road section“ und „tutorial section“ nicht gleichgesetzt werden. Zou et al. berechnen eine neue Wahrscheinlichkeit aus der alten Wahrscheinlichkeit, die nur einzelne Wörter betrachtet, und einem Beitrag, der abhängig ist von Ausdrücken, die in den Artefakten vorkommen. Der dritte Ansatz verwendet das Projektglossar, um wichtige Begriffe zu identifizieren, die höher gewichtet werden sollen als andere. Die drei Methoden werden auf Datensätzen getestet, die Anforderungen und UML- oder Java-Klassen beziehungsweise Anforderungen mit unterschiedlichem Abstraktionsgrad enthalten. Im Allgemeinen verbessern die Methoden die Präzision gegenüber dem Basisalgorithmus. Die Suchbegriffabdeckung führt zu Verbesserungen der Präzision von bis zu 20 % bei einer Ausbeute von 10 %. Auf einem Datensatz verschlechtert sich die Päzision allerdings für Ausbeuten bis 30 % gegenüber dem Basisalgorithmus.

In „Ontology-based multiperspective requirements traceability framework“ [ASP10] gehen Assawamekin et al. auf das Problem von multiperspektiven Anforderungsartefakten ein und stellen ihre multiperspective requirements traceability (MUPRET) Rahmenarchitektur vor. Multiperspektive Anforderungsartefakte bezeichnen Anforderungsartefakte, die aus unterschiedlichen Perspektiven des Problems, möglicherweise mit unterschiedlichem Vokabular, geschrieben werden. Da sich Anforderungsartefakte überlappen können, gibt es Probleme beim Generieren von Rückverfolgbarkeitsverbindungen, falls unterschiedliche Begriffe für gleiche Konzepte oder die gleichen Begriffe für unterschiedliche Konzepte verwendet werden. Die MUPRET Rahmenarchitektur verwendet Techniken zur Verarbeitung von natürlicher Sprache, regelbasierte Ansätze und Ontologiekonzepte. Sie besteht aus fünf Modulen: Anforderungsanalysator, Anforderungselementgenerator, Basisontologiekonstruktor, Anforderungsontologiekonstruktor und Ontologieableicher. Für die Evaluation wurden Anforderungen aus zwei verschiedenen Bereichen verwendet, die drei beziehungsweise zwei verschiedene Perspektiven enthalten. Die durchschnittliche Präzision ist je nach Art der Rückverfolgbarkeitsverbindung (komplette Überschneidung, eine Anforderung enthält eine andere, teilweise Überschneidung) zwischen 78 % und 87 %, die durchschnittliche Ausbeute zwischen 72 % und 77 %.

Lohar et al. stellen in „Improving Trace Accuracy through Data-Driven Configuration and Composition of Tracing Features“ [LAZCH13] ihren Ansatz Dynamic Trace Configuration (DTC) vor, der ein Verfahren zur Identifikation von Rückverfolgbarkeitsverbindungen zur Laufzeit konfiguriert. Es wird maschinelles Lernen verwendet, um anhand von bestätigten Rückverfolgbarkeitsverbindungen eine gute Konfiguration zu finden. DTC enthält ein Eigenschaftenmodell, das spezifiziert, welche Methoden zum Identifizieren von Rückverfolgbarkeitsverbindungen verfügbar sind, eine Simulationsumgebung, in der Konfigurationen Rückverfolgbarkeitsverbindungen identifizieren können, und eine Suchkomponente, die nach einer guten Konfiguration sucht. Eine Konfiguration wird bewertet, indem die identifizierten Rückverfolgbarkeitverbindungen mit den bestätigten Rückverfolgbarkeitverbindungen verglichen und die mittleren durchschnittlichen Präzisionen berechnet werden. Der Ansatz wird auf sechs Datensätzen getestet. Für jeden Datensatz erreichen unterschiedliche Konfigurationen das beste gefundene Ergebnis. Zum Beispiel verwendet die beste gefunde Konfiguration für einen Datensatz VSM, während die beste gefundene Konfiguration für einen anderen Datensatz LSI benutzt. Keine der besten gefundenen Konfigurationen liefert

gute Ergebnisse für alle getesteten Datensätze.

Guo et al. stellen in „Semantically Enhanced Software Traceability Using Deep Learning Techniques“ [GCCH17] eine Netzwerkarchitektur vor, die mithilfe von Worteinbettungen und rekurrenten neuronalen Netzen Rückverfolgbarkeitsverbindungen generiert. Als Datensätze für das Training und die Evaluation wurden Teilmengen aus 1651 Quellartefakten und 466 Zielartefakten des Positive Train Control Projekts verwendet. Außerdem wurden 52,7 MB Text aus Dokumenten des entsprechenden Wissensgebiets und für eine Variante der Worteinbettungskonfiguration zusätzlich 19,92 GB Text eines Wikipedia Speicherauszugs benutzt. Das Hauptziel war ein Verfahren zu finden, in dem das Netzwerk mit von Hand erstellten Rückverfolgbarkeitsverbindungen eines Projekts trainiert wird, so dass dieses automatisch neue Rückverfolgbarkeitsverbindungen generieren kann, wenn das Projekt weiterentwickelt wird. Zunächst werden für ein Quellartefakt und ein Zielartefakt für alle Wörter Worteinbettungen berechnet. Anschließend werden die Vektoren an das rekurrente neuronale Netz weitergeleitet, um für Quell- und Zielartefakt jeweils einen Vektor zu bilden. Abschließend werden die Vektoren miteinander verglichen. Guo et al. haben insgesamt 360 verschiedene Konfigurationen des Netzwerks getestet. Das beste gefundene Modell ist ein bidirektionales rekurrentes neuronales Netz mit Gated Recurrent Unit. Um die Netzwerkarchitektur mit führenden Algorithmen zum Generieren von Rückverfolgbarkeitsverbindungen zu vergleichen, wurde die mittlere durchschnittliche Präzision (MAP) auf einem Testdatensatz für das Netzwerk, VSM und LSI bestimmt. Das Netzwerk von Guo et al. erreichte eine MAP von 0.598, VSM 0.423 und LSI 0.451.

Zhao und Cao stellen in „An Improved Approach to Traceability Recovery Based on Word Embeddings“ [ZCS17] ein Verfahren vor, das Worteinbettungen und Learning to rank verwendet, um Rückverfolgbarkeitverbindungen zu generieren. Das Verfahren besteht aus zwei Phasen. Zunächst werden Worteinbettungen mithilfe von softwarebezogenen Wikipediaartikeln gelernt. In der gleichen Phase werden die Softwareartefakte vorbereitet, indem Satzzeichen, Zahlen und Stoppwörter entfernt werden. Für die zweite Phase wird Kosinus-Ähnlichkeit zwischen den Worteinbettungen verwendet. Die Suchanfrage wird erweitert, indem für die ersten 30% der Begriffe mit der höchsten inversen Dokumentenhäufigkeit Wörter hinzugefügt werden, die eine Kosinus-Ähnlichkeit von mindestens 0,7 haben. Um gefundene Kandidaten zu sortieren, wird die Learning to rank Methode Ranking SVM verwendet. Paare von Anfragen und Dokumenten werden anhand von Eigenschaften (Semantische Ähnlichkeit, Jaccard-Koeffizient, Summe von IDF-Werten von Anfragewörtern im Dokument, Anzahl der Schlüsselwörter im Dokument, Länge des Dokuments) auf einen Vektor abgebildet. Es wird der Abstand zwischen Vektorpaaren berechnet und das Problem wird auf ein Klassifikationsproblem abgebildet, das mit einen normalen SVM-Löser gelöst wird. Zhao und Cao experimentieren mit den fünf Datensätzen CM1-NASA, GANTT, eTOUR, ITrust und EasyClinic. Sie vergleichen ihre Methode WQI mit LSI und word2vec. WQI erreicht bessere durchschnittliche F_1-Werte für Rückverfolgbarkeitsverbindungen zwischen Anforderungen, zwischen Anwendungsfällen und Quelltext, zwischen Anwendungsfällen und Testfällen sowie zwischen Anwendungsfällen und Interaktionsdiagrammen. Learning to rank verbessert die mittlere durchschnittliche Präzision um 15,9 %.

In „Enhancing Unsupervised Requirements Traceability with Sequential Semantics“ [CWWW19] stellen Chen et al. ihren Ansatz S2Trace vor, der Sequenzmuster aus Softwareartefakten extrahiert und damit einen Dokumentenvektor berechnet, um Rückverfolgbarkeitsverbindungen zu identifizieren. S2Trace besteht aus fünf Schritten. Zuerst werden die textuellen Artefakte vorverarbeitet. Dafür werden Bezeichner geteilt und der Text in Tokens zerlegt. Anschließend werden nicht alphanumerische Zeichen, Stoppwörter und unnötige Tokens entfernt. Auf den Wörtern wird ein Stemming-Algorithmus ausgeführt. In den verarbeiteten Texten werden Sequenzmuster gesucht, um die sequentiellen Beziehungen zwischen Tokens einzufangen. Um unnötige Informationen zu vermeiden und um die Effizienz zu er-

höhen, wird der Abstand zwischen den Tokens eines Sequenzmusters beachtet. Mithilfe der ursprünglichen Begriffe und der gefundenen Sequenzmuster werden Dokumentenvektoren für die Softwareartefakte berechnet. Um Redundanz zu verringern und um aussagekräftigere Darstellungen der Artefakte zu bekommen, wird eine Methode zur Hauptkomponentenanalyse angewendet. Um Rückverfolgbarkeitsverbindungen zu identifizieren, wird die Kosinus-Ähnlichkeit zwischen den Dokumentenvektoren der Artefakte berechnet. Beträgt die Ähnlichkeit mehr als 70 % der maximalen Ähnlichkeit zwischen zwei Artefakten, identifiziert S2Trace eine Rückverfolgbarkeitsverbindung. Chen et al. verwenden die fünf Datensätze CM1-NASA, GANTT, eTOUR, iTrust und EasyClinic. Die Ergebnisse ihres Ansatzes vergleichen sie mit LSI, word2vec und WQI [ZCS17]. Gegenüber LSI verbessern sich Präzision und Ausbeute durchschnittlich um 31,2 % beziehungsweise 13,2 %. Verglichen mit word2vec sind Präzision und Ausbeute von S2Trace durchschnittlich um 5,15 % und 9,46 % besser. S2Trace erreicht einen leicht höheren durchschnittlichen Wert als WQI für die Präzision, aber einen niedrigeren Wert für die Ausbeute.

Wang et al. zeigen in „Enhancing Automated Requirements Traceability by Resolving Polysemy“ [WNLN18], wie sie ein künstliches neuronales Netz verwenden, um Mehrdeutigkeit in unterschiedlichen Anforderungen aufzulösen, mit dem Ziel, die Präzision von Information Retrieval Methoden zu verbessern. Zum Auflösen von Mehrdeutigkeit kann Koreferenz benutzt werden. Es wird angenommen, dass ein mehrdeutiger Begriff in verschiedenen Artefakten, der unterschiedliche Objekte bezeichnet, unterschiedliche Bedeutungen hat. Es wird der folgende Ablauf verwendet: Zuerst wird eine Begriff-Anforderungs-Matrix gebildet. Mithilfe eines vorwärtsgerichteten neuronalen Netzes wird für einen Begriff die Koreferenz mit dem höchsten Wert bestimmt. Ein weiteres Netz berechnet für einen Begriff aus verschiedenen Artefakten und seinen zuvor bestimmten Koreferenzen, ob dieser Begriff das gleiche Objekt bestimmt. Ist dies nicht der Fall, wird die Begriff-Anforderungs-Matrix aktualisiert, indem die Spalte des Begriffs geteilt wird. Abschließend wird eine Information Retrieval Methode eingesetzt, um Rückverfolgbarkeitsverbindungen zu identifizieren. Das Verfahren wird auf sechs Softwareprojekten trainiert und getestet. Anschließend wird es auf dem MODIS und dem CM-1 Datensatz evaluiert. Dafür werden die verbesserten Verfahren VSM-Polysemy und LSI-Polysemy mit VSM und VSM-POS-N beziehungsweise LSI und LSI-POS-N verglichen. Gegenüber VSM-POS-N gewinnt VSM-Polysemy nur leicht an Ausbeute, von 0,83 für MODIS und 0,90 für CM-1 auf 0,87 und 0,91. Ähnlich verhält es sich für LSI-POS-N mit Ausbeuten von 0,85 und 0,90 beziehungsweise LSI-Polysemy mit Ausbeuten von 0,91 und 0,93. Allerdings ist die Präzision gestiegen. VSM-Polysemy erreicht für MODIS eine Präzision von 0,32 und für CM-1 0,30, während VSM-POS-N Präzisionen von 0,22 und 0,23 erreicht. Die LSI-Ansätze verhalten sich ähnlich. LSI-Polysemy erreicht Präzisionen von 0,33 und 0,34. LSI-POS-N errreicht entsprechend 0,24 und 0,28.

Mills et al. stellen in „Automatic Traceability Maintenance via Machine Learning Classification“ [MEAH18] ihren Ansatz TRAIL (TRAceability lInk cLassifier) vor. TRAIL verwendet bestätigte Rückverfolgbarkeitsverbindungen, um einen Klassifikator zu trainieren, der vorhersagt, ob neue oder aktualisierte Rückverfolgbarkeitsverbindungen gültig sind. Die TRAIL-Rahmenstruktur besteht aus vier Hauptkomponenten. Eine Menge von Eigenschaften wird verwendet, um mögliche Rückverfolgbarkeitsverbindungen darzustellen. Mills et al. verwenden in ihrer Implementierung von TRAIL auf Information Retrieval basierende, Anfragequalitäts- und Dokumentstatistikeigenschaften. Eine Komponente zur Auswahl von Eigenschaften extrahiert aussagekräftige Eigenschaften. Da Trainingsdaten oft viel mehr ungültige als gültige Rückverfolgbarkeitverbindungen enthalten, wird eine Ausgleichstechnik auf die Daten angewendet, bevor der Klassifikator trainiert wird. Die vierte Komponente ist der Klassifikationsalgorithmus. Untersucht werden unter anderem k-nächste-Nachbarn mit $k = 5$, ein naiver Bayes-Klassifikator und Random Forest. Mills et al. evaluieren TRAIL auf elf Datensätzen mit unterschiedlichen Arten von Artefakttypen.

Die beste gefundene Konfiguration erreicht einen durchschnittlichen F-Wert von 75,18 % und verwendet SMOTE als Ausgleichstechnik, Pearson-Korrelation für die Auswahl der Eigenschaften und Random Forest als Klassifikationsalgorithmus. Der durchschnittliche F-Wert der jeweils besten Information Retrieval Methoden beträgt nur 48,71 %.

In „Trace Link Recovery using Semantic Relation Graphs and Spreading Activation“ von Schlutter und Vogelsang [SV20] wird ein Ansatz vorgestellt, der den Inhalt von Anforderungen als semantischen Graphen darstellt und Aktivierungsausbreitung verwendet, um Rückverfolgbarkeitsverbindungen zu generieren. Der Graph ähnelt einer Baumstruktur mit mehreren Wurzeln. Wurzelknoten stehen für die Anforderungen, Blätter für kleine Teile von natürlicher Sprache, wie einzelne Wörter. Knoten dazwischen repräsentieren Sätze. Entsprechend sind einzelne Wörter weiter von Anforderungen entfernt als Sätze. Um den Graphen aufzubauen, werden sowohl semantische Inhalte, als auch strukturelle Informationen, wie zum Beispiel die Ordnerstruktur, in der die Dokumente abgelegt sind, benutzt. Um Rückverfolgbarkeitsverbindungen zu identifizieren, wird ein Ausbreitungsaktivierungsalgorithmus verwendet, der von einem Quellartefakt ausgehend in jedem Schritt Ausgabeaktivierung, Kantenaktivierung, Eingabeaktivierung und Knotenaktivierung berechnet. Die finalen Knotenaktivierungen werden verwendet, um die Liste der Zielartefaktkandidaten zu sortieren. Um die gefundenen Ergebnisse zu verstehen, werden Ausbreitungsgraphen verwendet, die zeigen, welche Teile des ursprünglichen Graphen zur Aktivierung beigetragen haben. Für einen Benutzer können damit die relevanten Textstellen in den entsprechenden Anforderungen hervorgehoben werden. Für die Evaluation des Verfahrens werden fünf Datensätze verwendet und jeweils die durchschnittliche Präzision und Lag berechnet, wobei nur die höchsten 5, 10 beziehungsweise 30 gefundenen Dokumente berücksichtigt werden. Als Vergleich verwenden Schlutter und Vogelsang einen VSM-Ansatz. Der VSM-Ansatz liefert bessere Ergebnisse. Die Autoren sind der Meinung, dass Verbesserungen vor allem in der Veränderung des Graphen gesucht werden sollten.

Du et al. schlagen in „Automatic traceability link recovery via active Learning“ [DSH+20] einen Ansatz zum Identifizieren von Rückverfolgbarkeitverbindungen, der Aktives Lernen verwendet, vor. Der Ansatz bestimmt zufällig für eine kleine Anzahl an Artefaktpaaren, ob diese zusammengehören. Auf der so erstellten Trainingsmenge wird ein Klassifikator trainiert. Anschließend wird ein bisher nicht klassifiziertes Artefaktpaar gewählt und ein Experte muss von Hand bestimmen, ob eine Rückverfolgbarkeitsverbindung identifiziert werden soll. Je höher die Unsicherheit über die Klasse des Artefaktpaares ist, desto wahrscheinlicher ist es, dass es gewählt wird. Das Paar wird der Trainingsmenge hinzugefügt und der Klassifikator wird auf der Menge trainiert und ein neues Paar wird gewählt, das von Hand klassifiziert werden soll. Das wiederholt sich, bis eine Abbruchbedingung erfüllt ist. Es wird eine Menge an Eigenschaften bestimmt, die Rückverfolgbarkeitsverbindungen repräsentieren. Da die Anzahl an gültigen Rückverfolgbarkeitsverbindungen kleiner ist als die Anzahl an ungültigen, wird eine Ausgleichstechnik angewendet. Mithilfe der Trainingsmenge wird ein Random Forest als Klassifikator trainiert. Du et al. evaluieren ihren Ansatz auf sieben Datensätzen und vergleichen die Ergebnisse mit VSM mit Kosinus-Ähnlichkeit, wobei eine Schranke für jeden Datensatz so gewählt wird, dass der F-Wert möglichst hoch ist. Paare mit einem Ähnlichkeitswert über der Schranke werden als identifizierte Rückverfolgbarkeitsverbindungen betrachtet. Der Ansatz mit aktivem Lernen erreicht einen F-Wert von 54,19 % für Trainingsmengen, die 6 % des ursprünglichen Datensatzes enthält, durchschnittlich einen besseren Wert als VSM mit einem F-Wert von 38,36 %.

3.3 Ähnlichkeitsberechnung mit Sprachmodellen

In „Measurement of Semantic Textual Similarity in Clinical Texts: Comparison of Transformer-Based models“ stellen Yang et al. [YHZ+20] ihre auf Transformern basierenden Modelle

vor, die zum Bestimmen von semantischer Ähnlichkeit zwischen klinischen Dokumenten entwickelt wurden. Die vortrainierten Modelle werden in zwei Phasen trainiert. Zuerst werden sie mit einem Datensatz für allgemeine englische semantische Ähnlichkeit und anschließend mit einem Datensatz mit klinischen Dokumenten feinangepasst. Neben den allgemeinen base- und large-Versionen von BERT, RoBERTa und XLNet werden auch Versionen verwendet, die zusätzlich mit Arztberichten vortrainiert wurden. Für die Evaluation wird jeweils die Pearson-Korrelation bestimmt. Das beste Ergebnis erreicht das RoBERTa-large Modell mit einer Pearson-Korrelation von 0,9065. XLNet-large übertrifft die base-Variante mit einer Pearson-Korrelation von 0,8864 gegenüber 0,8470. Mit 0,8615 erreicht BERT-base einen besseren Wert als BERT-large, das 0,8549 erreicht. Die large- und base-Varianten der drei Modelle übertreffen die jeweiligen Modelle, die mit Arztberichten vortrainiert wurden.

Reimers und Gureviych stellen in „Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks“ [RG19] Sentence-BERT, eine Modifikation von BERT, die siamesische Netzstrukturen verwendet, um Satzeinbettungen zu berechnen, vor. Sentence-BERT verwendet die Ausgabe von BERT oder RoBERTa in einer Aggregations-Schicht, um Satzeinbettungen mit einer festen Länge zu erhalten. Um BERT beziehungsweise RoBERTa feinanzupassen, wird eine siamesische Netzstruktur verwendet. Zwei Sätze werden in jeweils eine BERT-Schicht mit gekoppelten Gewichten gegeben. Aus deren Ausgabe wird eine Satzeinbettung berechnet und die entstandenen Vektoren werden als Parameter einer Zielfunktion verwendet, die von den vorhandenen Trainingsdaten abhängt. Reimers und Gureviych trainieren Sentence-BERT auf SNLI [BAPM15], einem Datensatz, der Satzpaare enthält, die entweder mit „contradiction“, „entailment“ oder „neutral“ markiert sind, und auf MNLI [WNB18], einem Datensatz, der Satzpaare enthält, die viele Arten von Text enthalten. Sentence-BERT wird auf mehreren Aufgaben zur Bestimmung der semantischen Ähnlichkeit von Text ohne besonderes Training evaluiert. Reimers und Gureviych verwenden vier Varianten von Sentence-BERT, indem jeweils die base- und large-Variante von BERT und RoBERTa in der BERT-Schicht verwendet werden. Verglichen wird Sentence-BERT mit fünf anderen Methoden, darunter Universal Sentence Encoder [CYK+18], der BERT CLS-Vektor und der Durchschnitt von BERT-Ausgaben als Satzeinbettung. Das beste durchschnittliche Ergebnis hat Sentence-BERT mit RoBERTa-large. Für die Evaluation auf dem STS benchmark Testdatensatz [CDA+17] wurden Sentence-BERT und BERT mit dem Trainingsdatensatz feinangepasst. Sentence-BERT liefert bessere Ergebnisse als die nicht feinangepasste Version, allerdings übertrifft es nicht BERT, das auch auf dem NLI-Datensatz und dem STS benchmark Trainingsdatensatz feinangepasst wurde.

Um aus einer großen Anzahl an E-Mails diejenigen zu finden, die zu einer Anfrage passen, verwenden Sanjeev et al. in „Realtime Semantic Similarity Analysis of Bulk Outlook Emails Using BERT“ [SRT20] das Modell BERT, um die semantische Ähnlichkeit zwischen einer Anfrage und E-Mails in einem Postfach zu berechnen. Mithilfe von BERT werden Satzeinbettungen berechnet. Es wird eine Datenbank aufgebaut, in die die vorhandenen und alle neuen E-Mails aufgenommen werden. Dafür wird der Inhalt jeder E-Mail in Sätze zerlegt und es werden mithilfe von BERT Satzeinbettungen berechnet. Um nach einer E-Mail zu suchen, wird ein Anfragetext ebenfalls in einzelne Sätze zerlegt und es werden Satzeinbettungen berechnet. Zusätzlich bestimmt der Nutzer eine Zahl p. Es wird die Summe der p höchsten Werte für die Kosinus-Ähnlichkeiten zwischen der Anfrage und den Sätzen einer E-Mail berechnet. Abschließend werden die E-Mails mit der höchsten Ähnlichkeit angezeigt. Sanjeev et al. zeigen, dass für die Anfrage „Solution of a linear function“ in ihrem E-Mail Korpus, für eine einfache Suche, eine E-Mail gefunden wird, die von einer Salzlösung handelt. Die Methode mit den Satzeinbettungen findet eine E-Mail über lineare Funktionen, obwohl das Wort „solution“ nicht vorkommt.

Ostendorff et al. untersuchen in „Aspect-based Document Similarity for Research Papers“

[ORB+20] auf Transformern basierende Modelle, um Ähnlichkeit zwischen Forschungsarbeiten zu klassifizieren. Das System erhält als Eingabe den Titel und die Zusammenfassung von zwei Forschungsarbeiten und soll klassifizieren, auf welche Weise sie zusammenhängen. Es wird angenommen, dass zwei Arbeiten A und B ähnlich sind, wenn B in A zitiert wird. Als Klassenbezeichnung wird der Titel des Abschnitts verwendet, in dem B zitiert wird. Da eine Arbeit in mehreren Abschnitten zitiert werden kann, können Arbeiten in mehr als einem Aspekt ähnlich sein. Als Vergleich zu den auf Transformern basierenden Modellen wird ein neuronales Netz mit langem Kurzzeitgedächtnis verwendet. Für das Training und die Evaluation wurden zwei Datensätze verwendet. Einer enthält Forschungsarbeiten aus dem Bereich der Computerlinguistik, der andere enthält Forschungsarbeiten zu COVID-19. Alle auf Transformer basierende Modelle übertreffen das neuronale Netz. SciBERT [BLC19], ein auf wissenschaftlichen Arbeiten vortrainiertes Sprachmodell, übertrifft für die meisten Messwerte die anderen Modelle.

4 Analyse und Entwurf

Softwareanforderungen dokumentieren, welches Verhalten ein Softwareprojekt erfüllen muss. Das Vorhandensein von Rückverfolgbarkeitsverbindungen zwischen den Anforderungen unterstützt die zukünftige Entwicklung eines Projektes. Zum Beispiel ist es leichter auszuwerten, welche Auswirkungen Änderungen auf andere Teile des Projektes haben. Außerdem helfen Rückverfolgbarkeitsverbindungen dabei nachzuvollziehen, ob alle groben Anforderungen durch andere Artefakte, insbesondere durch detailliertere Anforderungen, abgedeckt sind.
Die unterschiedlichen Abstraktiongrade der Anforderungen sind ein Problem beim automatischen Identifizeren von Rückverfolgbarkeitsverbindungen zwischen Anforderungen. Eine Anforderung könnte zum Beispiel fordern, dass ein Programm „portabel“ sein soll, während eine zugehörige detailliertere Anforderung „Java“ als Programmiersprache fordert. Es können also nicht einfach nur die verwendeten Wörter miteinander verglichen, sondern die Bedeutung der Anforderungen muss untersucht werden. Beispiel 4.1 zeigt dazu ein Beispiel aus dem GANTT-Datensatz [HHD09] in dem zwei Anforderungen d7_1 und d7_3 jeweils Teile von Anforderung r1 genauer beschreiben. Die Anforderungen r7 und d7_1 verwenden das Wort „delete“, Anforderung d7_3 verwendet jedoch „removed“. In beiden Fällen geht es darum, dass etwas gelöscht wurde beziehungsweise dass etwas gelöscht werden soll.

Beispiel 4.1: Zusammengehörige Anforderungen aus dem GANTT-Datensatz [HHD09]

r7: Delete Resources (person); Delete a person and all its associated information such as dependencies to all tasks the person is supposed to work on.

d7_1: GUI needs to provide ability to delete resources from a project.

d7_3: When a resource is removed, all the tasks where the resource is assigned need to be updated.

Das Beispiel zeigt eine weitere Eigenschaft von Anforderungen. Detailliertere Anforderungen häufig nur Teile von zugehörigen gröberen Anforderungen. Anforderung r1 beschreibt, dass es eine Möglichkeit geben soll, um Ressourcen zu löschen. Verbundene Informationen, wie Abhängigkeiten von Aufgaben, für die eine gelöschte Ressource eingetragen war, sollen

auch gelöscht werden. Die Anforderung d7_1 beschreibt den ersten Teil, dass Ressourcen gelöscht werden sollen. Zusätzlich wird gefordert, dass diese Möglichkeit von der Benutzeroberfläche bereit gestellt wird. Anforderung d7_3 fordert, dass entsprechende Aufgaben angepasst werden.

Das Problem Rückverfolgbarkeitsverbindungen zwischen Anforderungen mit unterschiedlichem Abstraktionsgrad zu identifizieren, kann als Informationsrückgewinnungsproblem betrachtet werden [HDO03], [ZSCH10]. Dabei wird eine Anforderung als Anfragetext verwendet und es werden relevante Anforderungen identifiziert. In Systemen, die nicht komplett automatisch verwendet werden, kann eine Liste von Anforderungen sortiert nach ihrer jeweiligen berechneten Relevanz angezeigt werden.

Alternativ kann das Problem auch als Klassifikationsproblem aufgefasst werden. Es wird ein Klassifikator trainiert, der für ein Paar von Anforderungen vorhersagt, ob eine Rückverfolgbarkeitsverbindung zwischen diesen existiert oder nicht.

4.1 Zielsetzung

Es soll ein Verfahren entworfen werden, das mithilfe von Sprachmodellen automatisch Rückverfolgbarkeitsverbindungen zwischen Anforderungen mit unterschiedlichem Abstraktionsgrad identifiziert. Es werden nur natürlichsprachliche Anforderungen in Textform betrachtet.
Um dieses Ziel müssen zunächst vorhandene Sprachmodelle untersucht werden, um diejenigen auswählen zu können, bei denen davon auszugehen ist, dass sie gute Ergebnisse liefern können. In diesem Zusammenhang muss auch überprüft werden, ob eine weitere Feinanpassung von Modellen sinnvoll ist, die bereits für andere Probleme feinangepasst wurden.
Anschließend wird für die ausgewählten Sprachmodelle erarbeitet, wie sie jeweils geeignet eingesetzt werden können. Zum Beispiel können manche Modelle so feinangepasst werden, dass sie als Klassifikator dienen. Andere Sprachmodelle erreichen möglicherweise bessere Ergebnisse, indem mit ihnen Satzeinbettungen erstellt werden, mit denen ein Ähnlichkeitswert zwischen Anforderungen berechnet wird.
Die Art des Trainings muss definiert werden. Unter anderem muss festgestellt werden, wie vorhandene Datensätze aufbereitet werden müssen, um für das Training der Sprachmodelle verwendet werden zu können. Zusätzlich müssen Limitierungen der Sprachmodelle, wie zum Beispiel eine maximale Länge der Eingabe, beachtet und entsprechende Lösungen gefunden werden.

4.2 Sprachmodelle

Sprachmodelle lernen eine Repräsentation von Sprache und verwenden diese, um Aufgaben der Sprachverarbeitung zu lösen. Insbesondere Sprachmodelle, die auf großen Textkorpora vortrainiert wurden und sich für eine Aufgabe feinanpassen lassen, sind für diese Arbeit interessant. Diese Sprachmodelle haben ein generelles Verständnis einer oder mehrerer Sprachen gelernt. Da nicht nur einzelne Wörter, sondern auch deren Kontext, gelernt werden, haben Sprachmodelle ein Verständnis der Bedeutung von Wörtern. So werden Wörter mit der gleichen oder einer ähnlichen Bedeutung ähnlich aufgefasst. Das kann von Vorteil sein, wenn verschiedene Texte mit ähnlichen Bedeutungen verglichen werden sollen. Beim Identifizieren von Rückverfolgbarkeitverbindungen kann so möglicherweise der unterschiedliche Abstraktionsgrad der Anforderungen ausgeglichen werden. Ein weiterer großer Vorteil eines vortrainierten Sprachmodells ist die Menge an Trainingsdaten, die benötigt wird, um es für eine Aufgabe anzupassen. Die Modelle werden häufig auf allgemeinen englischen Texten trainiert und können deshalb beim Identifizeren von Rückverfolgbarkeitsverbindungen helfen, da es sich bei Anforderungen in der Regel um natürlichsprachliche Texte handelt.

Bidirectional Encoder Representations from Transformers (BERT) [DCLT19] ist ein auf Transformern basierendes Sprachmodell, dass zum Zeitpunkt seiner Veröffentlichung neue Rekorde für mehrere Sprachverarbeitungsaufgaben erreicht hat. Anders als andere Sprachmodelle lernt BERT bidirektional, so dass sowohl der Kontext vor als auch nach dem betrachteten Wort gelernt wird. Dafür wird eine „masked language model“ (MLM) Aufgabe verwendet, bei der Tokens in der Eingabe zufällig durch ein Maskentoken ersetzt werden. Die Trainingsaufgabe ist es, anschließend vorherzusagen, welches Wort ersetzt wurde. Zusätzlich wird BERT mit einer Aufgabe trainiert, bei der für ein Satzpaar vorhergesagt werden muss, ob es sich beim zweite Satz um den Satz handelt, der auf den ersten folgt. In der ursprünglichen Veröffentlichung werden die zwei Varianten BERT-base und BERT-large vorgestellt. Die Modelle verwenden die gleiche Struktur, unterschieden sich aber in der Anzahl der Komponenten und damit in der Gesamtanzahl der Parameter. Beide Modelle wurden auf dem BooksCorpus [ZKZ+15] und Artikeln der englischen Wikipedia vortrainiert. In der Regel sind die mit BERT-large erzielten Ergebnisse besser als die, die mit BERT-base erzielt wurden. Allerdings werden für die Benutzung und vor allem für das Training mehr Ressourcen benötigt, abhängig von der Größe des Modells. Devlin et al. fanden, dass die Feinanpassung für BERT-large auf kleinen Datensätzen instabil sein kann. Um das Problem zu lösen wählen sie das beste Modell aus mehreren Trainingsversuchen, in denen die Klassifizierungsschicht zufällig initialisiert und die Trainingsdaten zufällig gemischt wurden.

Das Sprachmodell RoBERTa [LOG+19] verwendet die gleiche Struktur wie BERT. Es unterscheidet sich von diesem durch den Prozess zum Vortrainieren. Zum Beispiel werden Hyperparameter so eingestellt, dass bessere Trainingergebnisse erzielt werden. Zusätzlich werden viel mehr Trainingsdaten, über 160 GB statt etwa 16 GB, für das Vortrainieren verwendet.

Unabhängig von der verwendeten Architektur gibt es die Möglichkeit ein Modell von Grund auf vorzutrainieren. Es ist in der Regel zu erwarten, dass Sprachmodelle, die auf Texten vortrainiert wurden, die zum Wissensgebiet der späteren Aufgabe gehören, bessere Ergebnisse liefern. Das Vortrainieren ist allerdings teuer und benötigt sehr große Datenmengen um den Vorteil der Sprachmodelle zu erhalten, möglichst gut generalisieren zu können. Im Fall von Softwareanforderungen ist es nur schwer möglich die benötigten Mengen zu finden. Zusätzlich sind Anforderungen nicht nur auf ein Wissensgebiet beschränkt. Da Software in vielen verschiedenen Bereichen, wie zum Beispiel der Medizin oder dem Verkehrswesen, eingesetzt wird, enthalten Anforderungen entsprechend Ausdrücke und Beschreibungen, die zu den jeweiligen Bereichen gehören. Daher ist es fraglich, ob ein mit Softwareanforderungen vortrainiertes Sprachmodell einen großen Mehrwert bietet, gegenüber einem Modell, das auf allgemeinen englischen Texten trainiert wurde.
SciBERT [BLC19] ist ein auf der BERT-Architektur basierendes Sprachmodell, dass auf wissenschaftlichen Veröffentlichungen aus dem Bereich der Medizin und Informatik vortrainiert wurde. Wissenschaftliche Texte entsprechen zwar nicht exakt Anforderungstexten, allerdings ist es möglich, dass das in computerwissenschaftlichen Arbeiten verwendete Vokabular ähnlicher zu dem von generellen Anforderungstexten ist, als das von allgemeinen Texten, mit denen andere Sprachmodelle vortrainiert werden. Insbesondere bei Projekten aus dem medizinischen Bereich könnte das Vortraining mit medizinischen Veröffentlichungen bei der Identifizierung von Rückverfolgbarkeitsverbindungen helfen.

4.2.1 Feinangepasste Modelle

Sentence-BERT [RG19] ist, wie in 3 beschrieben, eine Modifikation des BERT-Sprachmodells, das entwickelt wurde um Satzeinbettungen zu berechnen, die dazu verwendet werden können um Ähnlichkeitsberechnungen durchzuführen. Daher ist zu erwarten, dass die Sentence-BERT-Architektur für Abbildungsverfahren gute Ergebnisse liefert, die darauf

basierend Satzeinbettungen miteinander zu vergleichen. Ein Vorteil Satzeinbettungen berechnen zu lassen besteht darin, dass keine Anforderungspaare benötigt werden. Grobe und detaillierte Anforderungen können jeweils einzeln übergeben werden. So können Ressourcen gespart werden. Jede Anforderung muss zudem nur einmal dem Sprachmodell übergeben werden, wodurch Rechenzeit gespart wird.

4.2.2 Entwurf

Aufgrund der genannten Probleme, wie dass nicht genug Trainingsmaterial zur Verfügung steht, wird im Rahmen dieser Arbeit kein Sprachmodell von Grund auf trainiert. Stattdessen wird mit SciBERT experimentiert, um herauszufinden, ob das Vortraining mit ähnlichen, aber nicht gleichen, Texten Vorteile bringt. Um die Auswirkungen dieser und anderer Entwurfentscheidungen besser vergleichen zu können, sollte ein Sprachmodell verwendet werden, das für alle gewählten Abbildungsverfahren eingesetzt werden kann und das sich schnell feinanpassen lässt. Da SciBERT nur mit der Struktur von BERT-base existiert sollten die beiden Sprachmodelle auf die gleiche Weise feinangepasst werden um vergleichbare Ergebnisse zu erhalten. Da allerdings auch ein möglichst gutes Verfahren zum Identifizieren von Rückverfolgbarkeitsverbindungen zwischen Anforderungen gefunden werden soll, muss auch die large-Variante von BERT betrachtet werden, da diese meistens bessere Ergebnisse liefert. Dabei ist zu beachten, dass aufgrund von beschränkter Hardware die theoretischen Möglichkeiten von Sprachmodellen und insbesondere großen Sprachmodellen, wie BERT-large, nicht komplett ausgereizt werden können.

4.3 Abbildungsverfahren

Häufige Verfahren, um Rückverfolgbarkeitsverbindungen zu identifizieren, sind das Benutzen von Klassifikatoren und die Bestimmung eines Ähnlichkeitswertes zwischen einer vereinfachten Darstellung der Anforderungen.

4.3.1 Klassifikation

Um ein Sprachmodell als Klassifikator verwenden zu können wird eine Klassifikationsschicht benötigt oder ein zusätzlicher Klassifikator, der die Ausgabe des Sprachmodells als Eingabe erhält. Ein eigenständiger Klassifikator lässt sich frei wählen und leichter manuell an das Problem anpassen, sorgt jedoch dafür dass das Training komplizierter wird. Eine Klassfikationsschicht, die Teil des Modells selbst ist, wird während dem Feinanpassen trainiert. Die Kodierung kann also abhängig von der Ausgabe der Klassifikationsschicht gelernt werden.
Unabhängig von der gewählten Klassifikationsmethode handelt es sich um ein binäres Klassifikationsproblem. Entweder es gibt eine Rückverfolgbarkeitsverbindung zwischen zwei Anforderungen oder nicht. Für die Vorhersage mithilfe einer Kalssfikationsschicht müssen beide die allgemeinere Anforderung und die detailliertere Anforderung gleichzeitig in das Modell gegeben werden.

4.3.2 Satzeinbettung

Sprachmodelle können verwendet werden um Satzeinbettungen zu berechnen. Dafür werden allgemeine Anforderungen und detailliertere Anforderungen einzeln in das Sprachmodell gegeben. Die jeweiligen Satzeinbettungen können anschließend durch ein Ähnlichkeitsmaß verglichen werden.
Da die Berechnung der Ähnlichkeit für die Satzeinbettungen einen Wert zwischen 0 und 1 berechnet, muss für ein automatisches Verfahren entschieden werden, wann eine Rückverfolgbarkeitsverbindung identifiziert werden soll. Für Verfahren, die lediglich helfen sollen

Rückverfolgbarkeitsverbindungen von Hand zu identifizieren reicht häufig eine sortierte Liste. Die erste Möglichkeit ist es einen Schwellwert festzulegen. Für jede potenzielle Verbindung wird überprüft ob der berechnete Ähnlichkeitswert den Schwellwert übertrifft. Ist dies der Fall wird die eine Rückverfolgbarkeitsverbindung identifiziert. Ein Vorteil ist, dass allgemeine Anforderungen berücksichtigt werden, zu denen keine detaillierteren Anforderungen gehören. Allerdings ist die Wahl des Schwellwertes schiwerig, da die Bereiche der gefundenen Ähnlichkeitswerte stark vom betrachteten Softwareprojekt abhängig sind. Eine Alternative ist es die Anzahl der gefundenen detaillierteren Anforderungen für jede allgemeinere Anforderung auf ein Anzahl n zu beschränken. So werden jeweils für die n detaillierten Anforderungen eine Rückverfolgbarkeitsverbindung identifiziert, deren Ähnlichkeitswert zur allgemeineren Anforderung am höchsten ist. Diese Methode löst das Problem den Schwellwert festlegen zu müssen. Dafür werden für werden jeder allgemeinen Anforderung genau n detaillierte Anforderungen zugeordnet, unabhängig davon wie viele es eigentlich sein sollten. Eine dritte Möglichkeit besteht darin für jede allgemeine Anforderung die detailliertere Anforderung auszuwählen, für die der höchste Ähnlichkeitswert berechnet wurde und abhängig davon für alle Anforderungen eine Rückverfolgbarkeitsverbindung zu identifizieren, deren Ähnlichkeitswert mindestens einem festgelegten Prozentsatzes des Ähnlichkeitswertes der gewählten Anforderung entspricht. Auch bei dieser Methode werden allgemeineren Anforderungen mindestens eine detailliertere Anforderung zugeordnet. Allerdings ist diese Methode anpassungfähiger als die Festlegung einer Anzahl an identifizierten Rückverfolgbarkeitsverbindungen.

Eine Alternative zum direkten Vergleich zwischen den Satzeinbettungen von allgemeineren Anforderungen und detaillierteren Anforderungen ist die Clusterbildung von detaillierteren Anforderungen. Anforderungen können n:m-Beziehungen zueinander haben. Zu jeder allgemeineren Anforderung können beliebig viele detailliertere Anforderung gehören und jede detaillierte Anforderungen kann zu mehren allgemeineren Anforderungen gehören. Wie in Abschnitt 3.1 beschrieben, gibt es allerdings häufig eine 1:n-Beziehung zwischen allgemeineren und detaillierteren Anforderungen. Unter der Annahme, dass zusammengehörige detaillierte Anforderungen ähnlich zueinander sind, können zwischen ihnen Cluster gebildet werden, die dann zusammen allgemeineren Anforderungen zugeordnet werden können, so dass zwischen der allgemeineren Anforderung und jeder detaillierteren Anforderung im Cluster eine Rückverfolgbarkeitsverbindung identifiziert wird.

4.3.3 Entwurf

Basierend auf den vorgestellten Abbildungsverfahren sollen zwei verschiedene Grundverfahren entworfen werden. Das erste verwendet eine Klassifikationsschicht. Hierfür wird die Variante verwendet, die eine Klassifikationsschicht gemeinsam mit dem Sprachmodell feinanpasst. Bei diesem Verfahren, werden dem Sprachmodell jeweils eine grobe Anforderung und eine detaillierte Anforderung gleichzeitig übergeben. Die Klassifikationsschicht sagt anhand der Ausgaben der inneren Schichten voraus, ob zwischen den Anforderungen eine Rückverfolgbarkeitsverbindung identifiziert werden soll.
Das zweite Verfahren verwendet Sentence-BERT um Satzeinbettungen zu generieren. Grobe und detaillierte Anforderungen werden einzeln dem Sprachmodell übergeben um Satzeinbettung berechnen zu lassen. Die Satzeinbettungen werden für beide vorgestellten Arten verwendet. Für die erste Art wird die Ähnlichkeit zwischen den Anforderungen jeder Kombination zwischen grober und detaillierter Anforderung berechnet. Das verwendete Ähnlichkeitsmaß ist die Kosinus-Ähnlichkeit. Die Entscheidung, welche potenziellen Verbindungen als Rückverfolgbarkeitsverbindung identifiziert werden sollen kann später getroffen werden. Da bereits für alle möglichen Kombinationen von groben und detaillierten Anforderungen die Ähnlichkeiten berechnet werden können diese einfach in einer Liste gespeichert werden. Im Rahmen der Evaluation können einfach verschiedene Konfigurationen zur

Wahl von zu identifizierenden Rückverfolgbarkeitsverbindungen getestet werden. Die Art der Auwahl, wann das Verfahren eine Rückverfolgbarkeitsverbindung identifiziert, wird experimentell bestimmt. Für die zweite Art werden zunächst die detaillierten Anforderungen verglichen um Cluster zu generieren. Für jedes Cluster wird ein Repräsentant berechnet, der mit den Satzeinbettungen der groben Anforderungen verglichen wird.

4.4 Training

In den folgenden Abschnitten werden die Probleme bei der Feinanpassung der Sprachmodelle betrachtet.

4.4.1 Datensätze

Um sinnvoll für die Identifikation von Rückverfolgbarkeitsverbindungen eingesetzt werden zu können, müssen die Sprachmodelle feinangepasst werden. Dafür müssen Datensätze für das Training erstellt werden. Die einfachste Möglichkeit dafür wäre es, aus den in Abschnitt 3.1 vorgestellten Datensätzen alle möglichen Kombinationen von groben und detaillierten Anforderungen zu verwenden und diese entsprechend zu markieren, ob ein Rückverfolgbarkeitsverbindung identifiziert werden soll. Dieser Ansatz hat das Problem, dass die Datensätze unbalanciert sind. Der GANTT-Datensatz enthält 1173 mögliche Kombination zwischen den Anforderungen, aber nur bei 68 davon soll eine Rückverfolgbarkeitsverbindungen identifiziert werden. Bei anderen Datensätzen ist das Verhältnis zwischen positiven und negativen Beispielen noch schlechter. Aus den 51700 möglichen Kombinationen von Anforderungen im CM-1-Datensatz sind nur 325 positive Beispiele für Anforderungen zwischen denen eine Rückverfolgbarkeitsverbindung identifiziert werden soll. Unbalancierte Trainingsdaten bewirken beim Training eine Tendenz der Vorhersagen zur Klasse, die deutlich mehr Beispiele enthält. Um das zu verhindern, können Ausgleichstechniken verwendet werden. Es werden die zum Trainieren verwendeten Beispiele der größeren Klasse reduziert oder die der kleineren Klasse erhöht, so dass es in beiden Klassen etwa die gleiche Anzahl Trainingsbeispiele gibt. Am einfachsten ist es zufällig gewählte Beispiele aus der größeren Klasse zu entfernen. Diese Möglichkeit ist unabhängig vom Klassifikator beziehungsweise der späteren Aufgabe. Allerdings können Beispiele entfernt werden, die besonders aussagekräftig sind, weil sie zum Beispiel vom Klassifikator unsicher vorhergesagt werden würden. Um die Anzahl der Beispiele der kleineren Klasse der größeren anzupassen, lassen sich die Beispiele der kleineren Klasse kopieren. So werden sie häufiger im Trainingsprozess gesehen. Das kann wiederum dazu führen, dass das Sprachmodell zu sehr an die positiven Beispiel angepasst wird und die Fähigkeit zu generalisieren verloren geht.

4.4.2 Parameter

Viele Sprachmodelle haben einen Parameter für die maximale Sequenzlänge, der angibt, wie viele Tokens auf einmal bearbeitet werden. Abhängig von der maximalen Sequenzlänge wird mehr oder weniger Speicher benötigt. Die Maximale Sequenzlänge und die Batchgröße haben neben der Wahl des Sprachmodells die größte Auswirkung auf den Ressourcenverbrauch und müssen entsprechend gemeinsam abhängig von der verfügbaren Hardware gewählt werden. Optimal ist eine maximale Sequenzlänge, die so groß ist, dass keine Eingabe gekürzt werden muss. Werden Satzpaare, wie bei der Klassifikationsaufgabe, verwendet, besteht die Eingabe aus beiden Sätzen. Tabelle 4.1 enthält die durchschnittliche Anzahl an Tokens, die ein für BERT-base-uncased passender Tokenisierer erstellt. Tokenisierer für unterschiedliche Sprachmodelle berechnen verschiedene Anzahlen an Tokens, aber für viele Sprachmodelle liegt die Anzahl in einem ähnlichen Bereich, so dass die Daten in der Tabelle als grobe Einschätzung verwendet werden können. Im Schnitt

Tabelle 4.1: Anzahl der Tokens für Anforderungspaare aus verschiedenen Datensätze für das Modell BERT-base-uncased

Anzahl Tokens	GANTT	CCHIT	CM-1	InfusionPump	WARC	Modis
Längstes Paar	177	268	659	374	166	160
Durchschnitt Paar	69.69	58.42	146.48	126.58	59.58	73.47
Durchschnitt grob	41.71	27.47	35.67	90.05	26.79	47.53
Durchschnitt detailliert	28.99	31.94	111.82	37.54	33.79	26.94

würde eine maximale Sequenzlänge von 128 häufig ausreichen. Allerdings hat jedes betrachtete Projekt Anforderungspaare, die diese Länge, teilweise sehr deutlich, übersteigen. Zusätzlich ist zu beachten, dass es sich bei den sechs Datensätzen nur um einen kleinen Ausschnitt der möglichen Anforderungen handelt. Unabhängig davon, welche maximale Sequenzlänge verwendet wird, muss eine Strategie gewählt werden, was passiert wenn eine Anforderung oder ein Anforderungspaar die maximale Sequenzlänge überschreitet. Möglichkeiten um das Problem zu lösen, die für kaum zusätzlichen Aufwand beim Trainieren und beim Auswerten sorgen, sind, die Anforderungen zu kürzen. Für Anforderungspaare kann entweder ein Teil der groben oder der detaillierteren Anforderung abgeschnitten werden. Die offensichtliche Lösung wäre es von die jeweils längere Anforderung zu kürzen. Allerdings enthalten vor allem die groben Anforderungen Informationen, die auf mehrere detaillierte Anforderungen hinweisen während detaillierte Anforderungen eine Funktion genau beschreiben. Entsprechend ist es sinnvoll die detaillierten Anforderungen zu kürzen, um nicht zu riskieren, dass die benötigten Informationen am Ende der groben Anforderung stehen.

4.4.3 Entwurf

Die Art des Training ist abhängig vom gewählten Abbildungsverfahren. Für das Klassifikationsverfahren werden aus den vorhandenen Datensätzen alle Kombinationen von groben und detaillierten Anforderungen verwendet und es wird markiert, ob zwischen den Anforderungen eine Rückverfolgbarkeitsverbindung identifiziert werden soll oder nicht. Für das Verfahren, bei dem Satzeinbettungen direkt miteinander verglichen werden wird der gleiche Datensatz verwendet. Das Verfahren, dass zunächst Cluster bildet, wird so trainiert, dass für alle detaillierten Anforderungen, die Rückverfolgbarkeitsverbindungen zu einer gemeinsamen groben Anforderung haben, die berechnete Satzeinbettungen nahe beeinander liegen. Anforderungen, die keine grobe Anforderung teilen, werden entsprechend so markiert, dass ihre Satzeinbettung möglichst weit auseinander liegen soll. Um das Problem der Überanpassung zu vermeiden wird als Ausgleichstechnik nur das zufällige Entfernen von Negativbeispielen verwendet.
Die maximale Sequenzlänge wird auf 256 gesetzt. Damit sind die meisten Anforderungspaare abgedeckt. Durch die beschränkte Hardware sinkt entsprechend die maximal mögliche Batchgröße. Für die wenigen betrachteten Anforderungspaare, die mehr als 256 Tokens benötigen, wird die detaillierte Anforderung gekürzt, da benötigte Informationen häufig bereits am Anfang stehen, während für grobe Anforderungen auch oft Informationen am Ende benötigt werden um Rückverfolgbarkeitsverbindungen zu identifizieren.

4.5 Vorverarbeitung

Für viele Aufgaben zur Verarbeitung von natürlicher Sprache werden die verwendeten Texte vorverarbeitet. Dazu gehören Stoppwortentfernung und Stemming. Da Sprachmodelle

bereits vortrainiert sind scheint es logisch zu sein die Eingaben so vorzubereiten, dass sie ähnlich zu den für das Training verwendeten Texten sind. So bleiben eventuell gelernte Satzstrukturen erhalten und helfen möglicherweise bei der entsprechenden Aufgabe. Da die Sprachmodelle in der Regel auf natürlichsprachlichen Texten trainiert werden ist eine extra Stoppwortentfernung oder Stemming nicht sinnvoll, da es sich bei Anforderungen ebenfalls um natürlichsprachliche Texte handelt. Andere Anpassungen sind möglicherweise sinnvoll. In Beispiel 4.2 sind einige Anforderungen aus dem InfusionPump-Datensatz zu sehen, die eine Überschrift beziehungsweise eine kurze Zusammenfassung in Stichworten enthalten. Offensichtlich handelt es dabei sich nicht um natürlichsprachliche Sätze. Um eine natürlichsprachliche Darstellung zu erhalten können die Überschriften einfach entfernt werden. Für Anforderungen, wie R6.3.0(2) und R6.3.0(5), sollte sich dadurch keine große Änderung ergeben. Die verwendete Überschrift kommt mit fast der gleichen Wortwahl im Beschreibungstext vor. Allerdings enthält die Beschreibung von R7.2.0(1) nicht die entsprechende Überschrift, das Wort „confidentiality“. Sie hat aber die gleiche Bedeutung. Auch wenn ein Sprachmodell auf natürlichsprachlichen Texten vortrainiert wurde, so ist es nicht zwangsweise auf diese beschränkt. Entfernt man die Überschriften der Anforderungen, so gehen Informationen oder zumindest Betonungen der wichtigen Inhalte der Anforderungen verloren.

Beispiel 4.2: Detaillierte Anforderungen aus dem InfusionPump-Datensatz

R6.3.0(5): battery failure alarm. The PCA pump shall detect battery failure and issue a battery failure alarm.”

R6.3.0(2): remaining battery minutes. The user interface must show that the PCA pump is working on battery backup and an estimate of the number of minutes of battery-powered infusion remain.”

R7.2.0(1): confidentiality. Patient information must be restricted to those providing care for the patient and the patient.”

4.6 Zusammenfassung des Entwurfs

Es werden drei Verfahren entwickelt, die miteinander verglichen werden. Zwei von ihnen basieren auf Satzeinbettungen, während das dritte einen Klassifikator verwendet um Rückverfolgbarkeitsverbindung zu identifizieren. Abbildung Abbildung 4.1 (a) zeigt das Klassifikationsverfahren. Eine grobe und eine detaillierte Anforderung werden gemeinsam tokenisiert und dem Sprachmodell übergeben. Eine Klassifikationsschicht sagt vorher ob zwischen den beiden Anforderung eine Rpckverfolgbarkeitsverbindung identifiziert wird oder nicht.
In Abbildung Abbildung 4.1 (b) ist das erste von zwei Verfahren, die Sentence-BERT für Satzeinbettungen verwenden, zu sehen. Grobe und detaillierte Anforderungen werden getrennt tokenisiert und in identische BERT-Schichten gegeben. Eine Aggregations-Schicht berechnet aus der Ausgabe des Sprachmodells eine Satzeinbettung. Die beiden Satzeinbettungen werden anschließend mithilfe der Kosinus-Ähnlichkeit verglichen. Anhand eines Auswahlkriteriums wird entschieden, ob eine Rückverfolgbarkeitsverbindung identifiziert wird.
Das dritte Verfahren ist in Abbildung Abbildung 4.1 (c) zu sehen. Wie beim zweiten Verfahren werden für grobe und detaillierte Anforderungen getrennt Satzeinbettungen berechnet. Bevor sie miteinander verglichen werden, werden die detaillierten Anforderungen

in Cluster zusammengefasst.

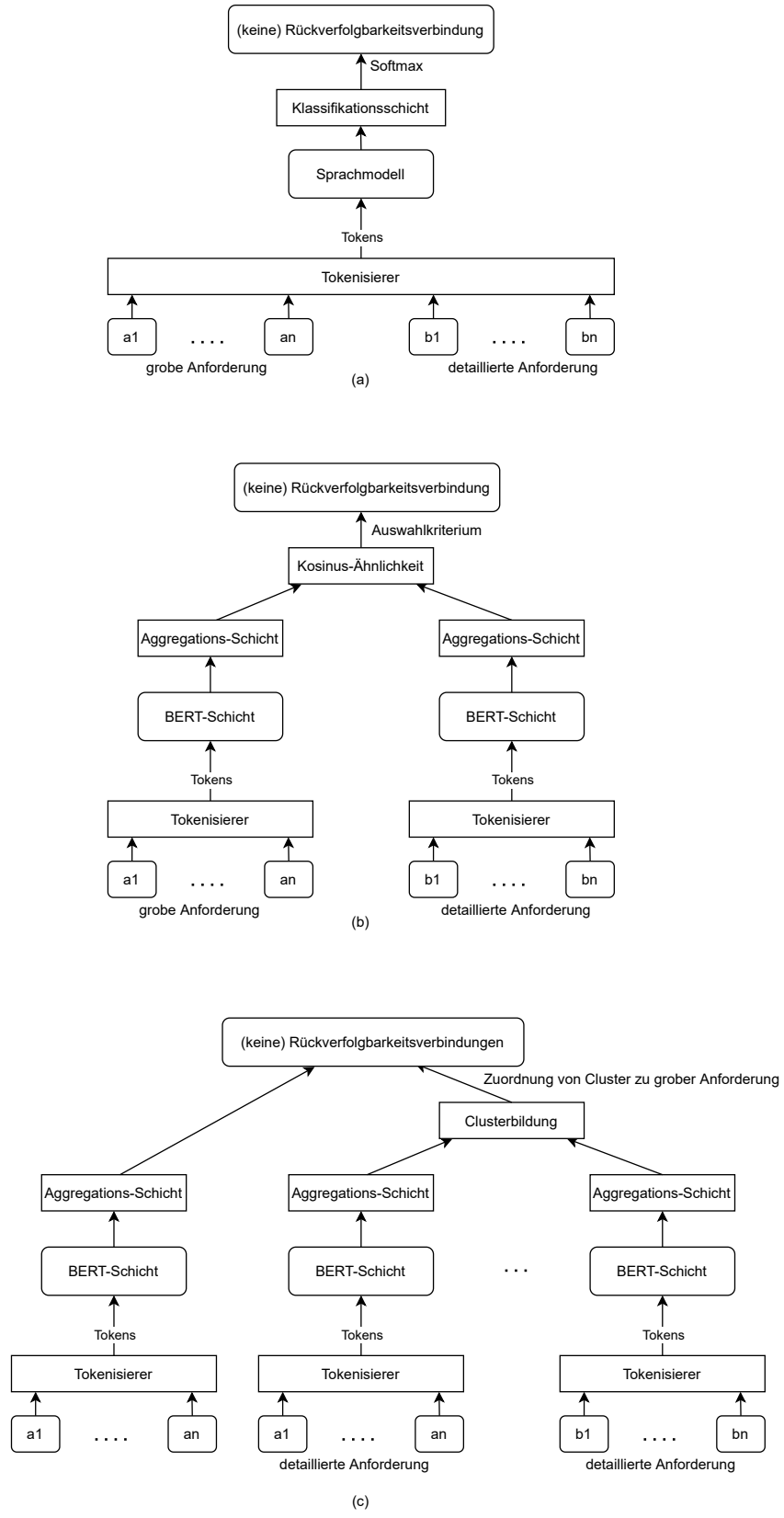

Abbildung 4.1: Struktur der Verfahren zum Identifizieren von Rückverfolgbarkeitsverbindungen

5 Implementierung

In diesem Kapitel wird beschrieben, wie die in Kapitel 4 getroffenen Entscheidungen implementiert werden. Alle Verfahren wurden in Python implementiert. Startwerte für Zufallszahlen sind zufällig gewählt und werden an den entsprechenden Stellen angegeben.

5.1 Datensätze

Für die Feinanpassung und die Evaluation der Sprachmodelle werden sechs Datensätze verwendet: GANTT, CCHIT, CM-1, InfusionPump, WARC und Modis. Die Datensätze werden als jeweils drei CSV-Dateien gespeichert und als *DataFrame* der pandas-Bibliothek [RMJ+20] geladen. Die ersten beiden Dateien enthalten die IDs von allen groben beziehungsweise allen detaillierteren Anforderungen und ihre zugehörigen Texte. In der dritten Datei sind alle Kombinationen von IDs, zwischen denen eine Rückverfolgbarkeitsverbindung existiert, enthalten. Mithilfe der CSV-Dateien werden zusätzlich alle Kombinationen an groben und detaillierteren Anforderungen gebildet, zwischen denen keine Rückverfolgbarkeitsverbindung existiert. Diese werden beim Training und der Evaluation als negative Beispiele verwendet.
Um gute Hyperparameter bestimmen zu können, ohne die Ergebnisse auf den zu testenden Datensätzen zu verfälschen, werden die für das Training verwendeten Datensätze in jeweils einen Trainingsdatensatz und einen Validierungsdatensatz geteilt. Hierfür wird die Methode *train_test_split* aus der Bibliothek scikit-learn [PVG+11] verwendet. Der Parameter *random_state* ist für die positiven Beispiele 1506 und für die negativen Beispiele 162. Um die Klassengrößen zwischen positiven und negativen Beispielen auszugleichen werden mithilfe von *random.sample* negative Beispiele ausgewählt, die mit der Methode *DataFrame.drop* entfernt werden. Der Startwert für die Zufallszahlen ist 45. Aus dem Validierungsdatensatz werden keine negativen Beispiele entfernt um das tatsächliche Verhältnis zwischen Anforderungen mit Rückverfolgbarkeitsverbindungen und ohne beizubehalten.

5.2 Klassifikation

Um die Sprachmodelle für die Klassifikationsaufgabe feinanzupassen wird die Bibliothek Transformers v4.5.0 [WDS+20] verwendet. Es werden die Modelle „bert-base-cased“ [1] und „allenai/scibert_scivocab_cased“ [2] verwendet. Die maximale Sequenzlänge wird auf 256

[1] `https://huggingface.co/bert-base-cased`
[2] `https://huggingface.co/allenai/scibert_scivocab_cased`

gesetzt und um diese einzuhalten wird zunächst die detaillierte Anforderung gekürzt. Mithilfe des entsprechenden Tokenisierers werden für den Trainings- und den Validierungsdatensatz die benötigten Eingabedaten, wie *input_ids* und *attention_mask* berechnet. Die Feinanpassung selbst wird mit der Trainer-Klasse durchgeführt, die viele Teilschritte der Feinanpassung automatisch durchführt. Als Lernrate wird 1e-5 gewählt. Die Batchgröße ist 16 und es wird für 4 Epochen trainiert. Die Trainer-Klasse verwendet standardmäßig einen AdamW-Optimizer [LH19]. Für den Parameter *weight_decay* wird 0.01 verwendet.
Das Modell wird mit der Methode *AutoModelForSequenceClassification.from_pretrained()* geladen. Dadurch werdem einem Modell ohne Klassifikationsschicht automatisch die benötigten Gewichte initialisiert. Zuvor wird mit *torch.manual_seed* ein Startwert für Zufallszahlen festgelegt.
Um das feinangepasste Sprachmodell zu evaluieren werden für den Testdatensatz die Eingabedaten, mit der gleichen Art Tokenisierer und den gleichen Parametern, berechnet. Diese Eingabe werden dem Modell übergeben. Um die vorhergesagte Klasse zu bestimmen, wird über dem ersten Element der Ausgabe die Softmax-Funktion berechnet und die Ergebnisse als Liste gespeichert. Das erste Element der Liste enthält die Wahrscheinlichkeit, dass keine Rückverfolgbarkeitsverbindung vorhergesagt wird, das zweite Element enthält entsprechend die Wahrscheinlichkeit für eine Rückverfolgbarkeitsverbindung.

5.3 Semantische Ähnlichkeit

Für die Berechnung der semantischen Ähnlichkeit zwischen groben und detaillierten Anforderungen mithilfe von Sprachmodellen wird die Bibliothek Sentence-Transformers [RG19] verwendet. Für die BERT-Schicht werden wieder „bert-base-cased“ und „allenai/scibert_scivocab_cased“ mit einer maximalen Sequenzlänge von 256 verwendet. Die Aggregations-Schicht verwendet mean-pooling um die Satzeinbettungen zu bilden. Um den Trainingsverlust zu bestimmen wird *sentence_transformers.losses.CosineSimilarityLoss* verwendet.
Wie bei der Klassifikationsaufgabe werden die Modelle mit einer Batchgröße von 16 für 4 Epochen feinangepasst. Hier wird die *torch.utils.data.DataLoader*-Klasse verwendet, mit welcher auch die Trainingsdaten gemischt werden. Der gewählte Startwert für die Zufallszahlen ist 75.
Um ein Modell zu evaluieren, werden zu allen Anforderungen mithilfe der Methode *encode* Satzeinbettungen berechnet. Anschließend wird die Satzeinbettung jeder groben Anforderung mit den Einbettungen aller detaillierteren Anforderungen verglichen indem mit der Methode *util.pytorch_cos_sim* die Kosinus-Ähnlichkeit berechnet wird.

Für die Methode, die Cluster verwendet um Rückverfolgbarkeitsverbindungen zu identifizieren wird das Training wie beschrieben durchgeführt. Um Cluster zu bilden wird *KMeans* von scikit-learn verwendet. Die Anzahl der Cluster wird dabei auf die Anzahl der groberen Anforderungen gesetzt. Mit *torch.mean* wird für jeden Cluster ein Repräsentant berechnet. Statt den einzelnen Satzeinbettungen der detaillierten Anforderungen werden die Einbettungen der groben Anforderungen mit den Repräsentanten verglichen.

5.4 10-fache Kreuzvalidierung

Um die Sprachmodelle mithilfe 10-facher Kreuzvalidierung zu evaluieren, wird *StratifiedKFold* von scikit-learn verwendet um das Verhältnis von positiven und negativen Beispielen des Datensätzes für alle Teilmengen beizubehalten. Die Sprachmodelle werden anschließend mit neun Teildatensätzen wie beschrieben trainiert und auf dem zehnten evaluiert. Dies wird zehnmal wiederholt, für jeden Teildatensatz einmal. Für jede Trainingsphase werden aus den entsprechenden neun Teildatensätzen zufällig negative Beispiele entfernt, sodass es gleich viele positive Beispiele, wie negative Beispiele gibt. Es wird die gleiche

Methode zum Entfernen der Beispiele verwendet wie bei der Erstellung der Trainingsdatensätze. Das bedeutet mithilfe von *random.sample* und dem Startwert für Zufallszahlen 45 werden negative Beispiele ausgewählt, die entfernt werden.

6 Evaluation

In diesen Kapitel soll bestimmt werden, welches der drei Verfahren, Klassifikation, Ähnlichkeit von Satzeinbettungen oder Satzeinbettungen mit Clusterbildung, sich am besten eignet um Rückverfolgbarkeitsverbidungen zwischen Anforderungen zu identifizieren. Zusätzlich wird geprüft, ob die Verfahren davon profitieren, dass ein Sprachmodell eingesetzt wird, dass nicht nur auf allgemeinen Texten, sondern auch auf, vor allem medizinischen und computerwissenschaftlichen Arbeiten vortrainiert wurde.

6.1 Experimentdesign

Um die in Kapitel 4 beschriebenen Verfahren zu evaluieren, werden die in Abschnitt 3.1 vorgestellten Datensätze verwendet. Um die Verfahren feinanzupassen, werden fünf der sechs Datensätze verwendet und auf dem sechsten wird das feinangepasste Modell evaluiert. So soll überprüft werden, ob und wie gut die Sprachmodelle Gelerntes generalisieren und auf komplett ungesehene Daten beziehungsweise ungesehene Projekte anwenden können. Zusätzlich werden die Verfahren in 10-facher Kreuzvalidierung evaluiert, um das Verhalten der Modelle auf Daten eines teilweise bekannten Projekts zu testen. Dafür wird zu jedem Zeitpunkt nur ein Datensatz verwendet.

6.2 Metriken

Um die Verfahren sowohl untereinander, als auch mit anderen Verfahren vergleichen zu können müssen die verwendeten Kennzahlen definiert werden. Für die Evaluation binärer Klassifikationsaufgaben gibt es für die Vorhersage jedes Beispiels vier mögliche Ergebnisse. Bei der Identifizierung von Rückverfolgbarkeitsverbindungen sind dies:

- Richtig positiv: Das Verfahren sagt vorher, dass eine Rückverfolgbarkeitsverbindung identifiziert wird und der Musterlösung nach soll eine Rückverfolgbarkeitsverbindung identifiziert werden.
- Richtig negativ: Die Vorhersage ist negativ und es soll keine Rückverfolgbarkeitsverbindung identifiziert werden.
- Falsch positiv: Es soll keine Rückverfolgbarkeitsverbindung identifiziert werden, aber das Verfahren sagt voraus, dass eine Rückverfolgbarkeitsverbindung identifiziert wird.

Tabelle 6.1: Präzision, Ausbeute und F_1-Wert für das Verfahren mit Klassifikationsschicht

Datensatz	BERT			SciBERT		
	P	R	F1	P	R	F1
GANTT	0,110	0,941	0,197	0,110	0,941	0,197
CCHIT	0,011	0,898	0,021	0,019	0,819	0,036
CM1	0,045	0,781	0,085	0,029	0,856	0,056
InfusionPump	0,170	0,672	0,271	0,170	0,672	0,271
WARC	0,042	0,993	0,081	0,039	0,963	0,075
Modis	0,201	0,750	0,317	0,263	0,650	0,374

- Falsch negativ: Das Verfahren sagt vorher, dass keine Rückverfolgbarkeitsverbindung identifiziert wird, obwohl der Musterlösung nach eine existiert.

Mit der Anzahl an Vorkommnissen dieser Ergebnisse lassen sich die folgenden Metriken definieren:
Präzision: Die Präzision ist der Anteil der vorhergesagten Rückverfolgbarkeitsverbindungen, die, der Musterlösung nach, vorhergesagt werden sollen. Eine hoher Präzisionswert gibt also an, dass es sich bei den meisten identifizierten Rückverfolgbarkeitsverbindungen tatsächlich um Rückverfolgbarkeitsverbindungen handelt und dass das Verfahren nur selten Verbindungen erkennt, wenn keine existieren. Die Präzision lässt sich mit Gleichung 6.1 berechnen.

$$Präzision = \frac{richtigpositiv}{richtigpositiv + falschpositiv} \tag{6.1}$$

Ausbeute: Die Ausbeute ist der Anteil der korrekt vorhergesagten Rückverfolgbarkeitsverbindungen aus allen zu identifizierenden Rückverfolgbarkeitsverbindungen. Ein hoher Wert für die Ausbeute zeigt an, dass das Verfahren die meisten Rückverfolgbarkeitsverbindungen identifiziert. Die Ausbeute wird mit Gleichung 6.2 berechnet.

$$Ausbeute = \frac{richtigpositiv}{richtigpositiv + falschnegativ} \tag{6.2}$$

Sowohl Präzision als auch Ausbeute sollten möglichst hoch sein. Beide Metriken lassen sich trivial maximieren. Die Präzision kann maximiert werden, indem nur wenige richtig positive Ergebnisse genommen werden. Dadurch erhält man in der Regel allerding eine niedrige Ausbeute. Umgekehrt lässt sich die Ausbeute trivial maximieren, indem zwischen allen groben und detaillierten Anforderungen eine Rückverfolgbarkeitsverbindung identifiziert wird, da so alle positiven Ergebnisse enthalten sind. Die Präzision ist bei so einem Vorgehen in der Regel sehr schlecht. Da Präzision und Ausbeute alleine nicht unbedingt aussagekräftig sind, wird häufig das harmonische Mittel wie in Gleichung 6.3 berechnet.

$$F_1 = 2\frac{Präzision \cdot Ausbeute}{Präzision + Ausbeute} \tag{6.3}$$

6.3 Klassifikation

Tabelle 6.1 zeigt die Präzision (P), Ausbeute (R) und den F_1-Score für die feinangepassten BERT- und SciBERT-Sprachmodelle mit Klassifikationsschicht. In der Regel ist die Ausbeute hoch und und die Präzision sehr niedrig bis niedrig. Das könnte bedeuten, dass das Balancieren der Datensätze eine zu große Auswirkung hatte und die Tendenz der Vorhersage in die Richtung geht, dass eine Rückverfolgbarkeitsverbindung identifiziert wird. Die Datensätze, Modis und InfusionPump, für die die höchsten Präzisionswerte erreicht wurden, haben gleichzeitig die niedrigsten Ausbeutewerte. Dieser Zusammenhang ist allerdings stark vom verwendeten Datensatz abhängig. Für CM-1 hat das Verfahren mit BERT

Tabelle 6.2: Präzision, Ausbeute und F_1-Wert für den CM-1-Datensatz für ein auf GANTT, InfusionPump, WARC und Modis feinangepasstes BERT-Sprachmodell

Datensatz	P	R	F1
CM-1	0,073	0,576	0,130

eine Ausbeute von 0,781 erreicht. Das ist nur um 0,031 höher als die Ausbeute für Modis. Trotzdem ist die Präzision mit 0,045 deutlich niedriger als die Präzision die für Modis erreicht wurde. Auch wurde für den GANTT-Datensatz ein Präzisionswert von 0,110 erreicht, während die Ausbeute bei über 0,9 liegt. Anders, als zunächst angenommen, scheint es also keinen von den Datensätzen unabhängigen Zusammenhang zwischen Präzision und Ausbeute zu geben. Der CCHIT-Datensatz ist deutlich größer als die anderen Datensätze. Möglicherweise überdecken, die aus dem CCHIT-Datensatz entnommenen Beispiele, die Beispiele anderer Datensätze bei der Feinanpassung. Um diese Hypothese zu prüfen wird BERT beispielhaft auf den vier kleinsten, gemessen an der Anzahl der Rückverfolgbarkeitsverbindungen in der Musterlösung, Datensätzen feinangepasst und anschließend mit den CM-1-Datensatz evaluiert. Das Ergebnis ist in Tabelle 6.2 zu sehen. Durch die Feinanpassung ohne Beispiele aus dem CCHIT-Datensatz ist die Ausbeute um etwas mehr als 0,2 niedriger als bei der Feinanpassung mit dem CCHIT-Datensatz. Gleichzeitig ist die Präzision um 0,03 höher. Dadurch steigt insgesamt auch der F_1-Wert von 0,085 auf 0,130. Beim Feinanpassen von BERT speziell für den CM-1-Datensatz mit dem CCHIT-Datensatz scheinen Strukturen gelernt zu werden, die dafür sorgen, dass für ein Beispiel wahrscheinlicher eine Rückverfolgbarkeitsverbindung vorhergesagt wird. Es lässt sich allerdings keine allgemeingültige Aussage treffen, ob die Feinanpassung mit allen Beispielen der Datensätze zu den besten Ergebnissen führt. Es ist möglich, dass eine balanciertere Auswahl der Trainingsbeispiele zu besseren Ergebnissen führen könnte.

Um der Frage nachzugehen, ob ein Vortraining des Sprachmodells in einem möglicherweise ähnlichem Wissensgebiet beim Identifizieren von Rückverfolgbarkeitsverbindungen einen Vorteil hat, werden die Ergebnisse von BERT und SciBERT miteinander verglichen. Für dne GANTT-Datensatz und den InfusionPump-Datensatz sind die Ergebnisse identisch. Obwohl InfusionPump aus dem Bereich der Medizin stammt, scheint SciBERT kein besseres Verständnis über die Zusammenhänge zwischen den Anforderungen zu haben. Für den zweiten Datensatz aus dem Medizinbereich, CCHIT, verbessern sich die Ergebnisse leicht. Statt einem F_1-Wert von 0,021 liefert SciBERT einen Wert von 0,036. Dabei ist zu beachten, dass die Präzision von 0,011 bei BERT auf 0,019 gestiegen ist, während die Ausbeute um fast 0,08 gesunken ist. Einen ähnlichen Effekt gab es für den MODIS-Datensatz. Die Präzision ist leicht höher als bei BERT, dafür ist die Ausbeute gesunken. Das Gegenteil ist der Fall für den CM-1-Datensatz. Die mit SciBERT gefundene Ausbeute ist höher als die mit BERT gefundene. Dafür ist die Präzision niedriger. Für den WARC Datensatz bringt SciBERT keine Vorteile. Sowohl Präzision, als auch Ausbeute, sind niedriger als mit BERT. Insgesamt lässt sich sagen, dass das Vortraining auf wissenschaftlichen Veröffentlichungen zu leicht besseren Ergebnissen führen kann, dies aber nicht im Allgemeinen passiert. Ebenso sind negative Änderungen an den Ergebnissen eher gering.

6.4 Satzeinbettungen

Die Methode, die Satzeinbettungen berechnet und vergleicht, gibt für eine gegebenes Beispiel nicht direkt eine vorhergesagte Klasse zurück. Abbildung 6.1, 6.2, 6.3, 6.4, 6.5 und 6.6 zeigen die Präzision des Verfahrens auf den Testdatensätzen für verschiedene Ausbeutewerte. Für alle Datensätze, bis auf WARC und Modis, sinkt die Präzision bereits für

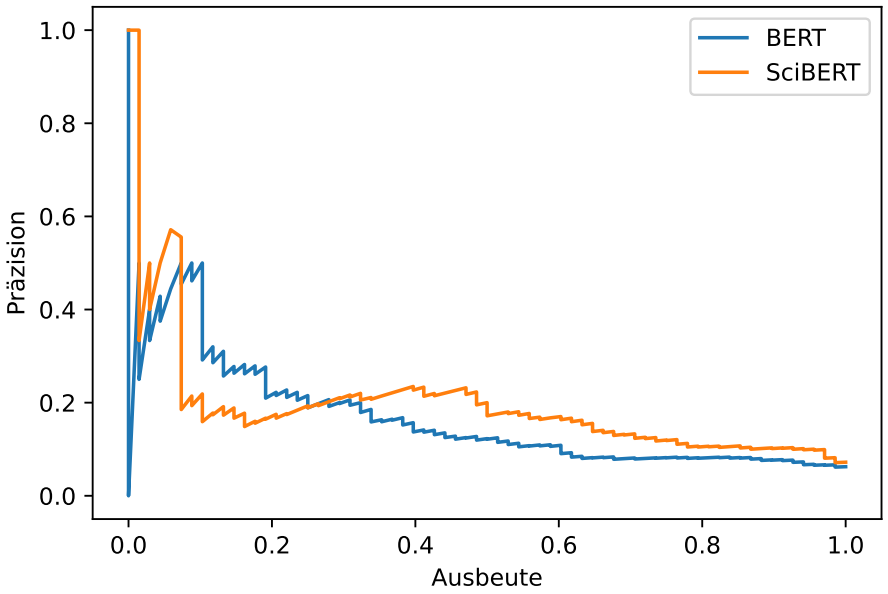

Abbildung 6.1: Präzision-Ausbeute-Kurve für den GANTT-Datensatz

Tabelle 6.3: Schwellwerte und F_1-Werte für die verwendeten Validierungsdatensätze

Validierungsdatensatz	BERT		SciBERT	
	Schwellwert	F1	Schwellwert	F1
ohne GANTT	0,820	0,264	0,792	0,336
ohne CCHIT	0,738	0,272	0,729	0,229
ohne CM1	0,777	0,285	0,768	0,366
ohne InfusionPump	0,822	0,293	0,818	0,332
ohne WARC	0,766	0,286	0,804	0,315
ohne Modis	0,806	0,276	0,744	0,344

kleine Ausbeuten sehr stark. Die Kurve eines gut funktionierenden Verfahrens sinkt in der Regel für kleine Ausbeuten langsam und erst für sehr große Ausbeutewerte schnell. Ein perfektes Verfahren hätte natürlich konstant einen Präzisionswert von 1.0, egal wie groß die Ausbeute ist.
Um ein automatisches Verfahren zu bauen, wird ein Schwellwert bestimmt, den der berechnete Ähnlichkeitswert übersteigen muss, damit zwischen zwei Anforderungen eine Rückverfolgbarkeitsverbindung vorhergesagt wird. Damit wird aus allen möglichen Präzisions-Ausbeute-Paaren eines bestimmt. Da der Schwellwert ein Parameter des Verfahrens ist, der möglichst unabhängig von den Testdaten berechnet werden soll, wird der Validationsdatensatz verwendet um diesen zu bestimmen. Der Schwellwert wird für jedes feinangepasste Sprachmodell separat bestimmt, da für die Feinanpassung der Sprachmodelle jeweils ein anderer Datensatz als Testdatensatz weggelassen wurde. Er wird so gewählt, dass der F_1-Wert für die entstehende Klassifizierung auf dem Validationsdatensatz maximal ist. Tabelle 6.3 enthält die bestimmten Schwellwerte zusammen mit den entsprechenden F_1-Wert.

In Tabelle 6.4 sind die Ergebnisse zu sehen, die erhalten werden, indem die gefundenen Schwellwerte auf die Ausgaben zu dem Testdatensatz des jeweiligen, feinangepassten Sprachmodells angewendet werden. Für den GANTT-Datensatz wurden mit dem zuvor bestimmten Schwellwert keine Rückverfolgbarkeitverbindungen identifiziert. Für alle Da-

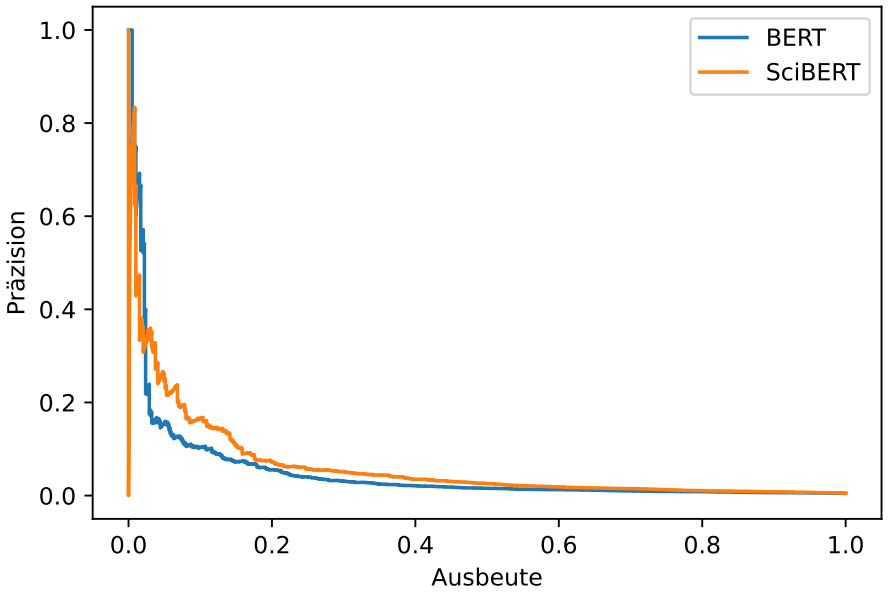

Abbildung 6.2: Präzision-Ausbeute-Kurve für den CCHIT-Datensatz

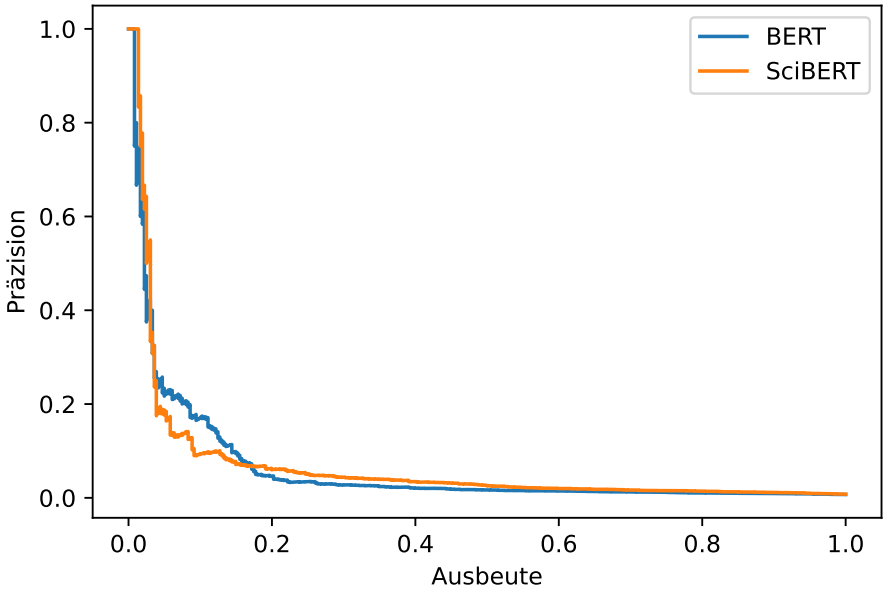

Abbildung 6.3: Präzision-Ausbeute-Kurve für den CM1-Datensatz

Tabelle 6.4: Präzision, Ausbeute und F_1-Wert für das Verfahren mit Satzeinbettungen

Datensatz	BERT			SciBERT		
	P	R	F1	P	R	F1
GANTT	-	0	-	0,556	0,074	0,130
CCHIT	0,128	0,068	0,089	0,095	0,158	0,119
CM1	0,254	0,042	0,071	0,098	0,122	0,109
InfusionPump	0,333	0,008	0,015	0,400	0,015	0,029
WARC	0,172	0,412	0,243	0,327	0,375	0,349
Modis	0,500	0,025	0,048	1,000	0,075	0,140

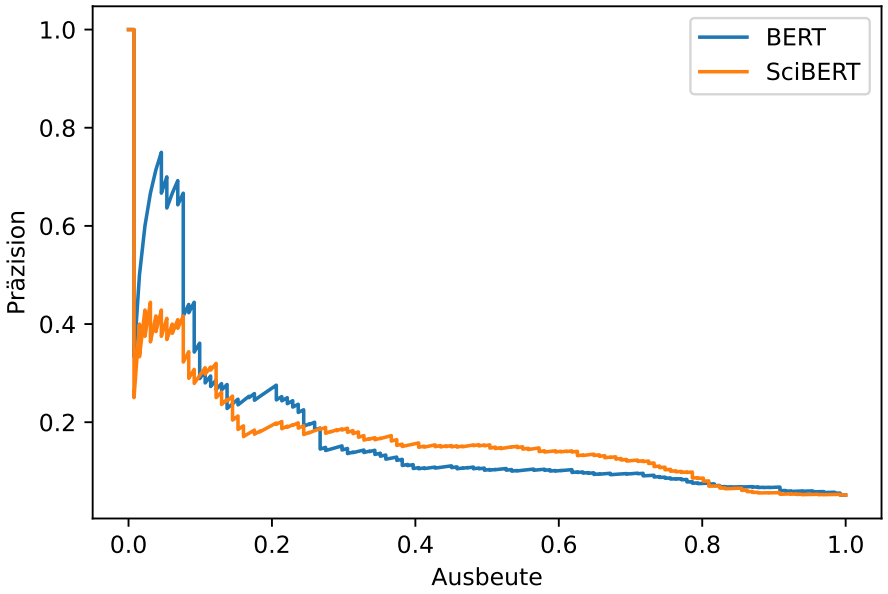

Abbildung 6.4: Präzision-Ausbeute-Kurve für den InfusionPump-Datensatz

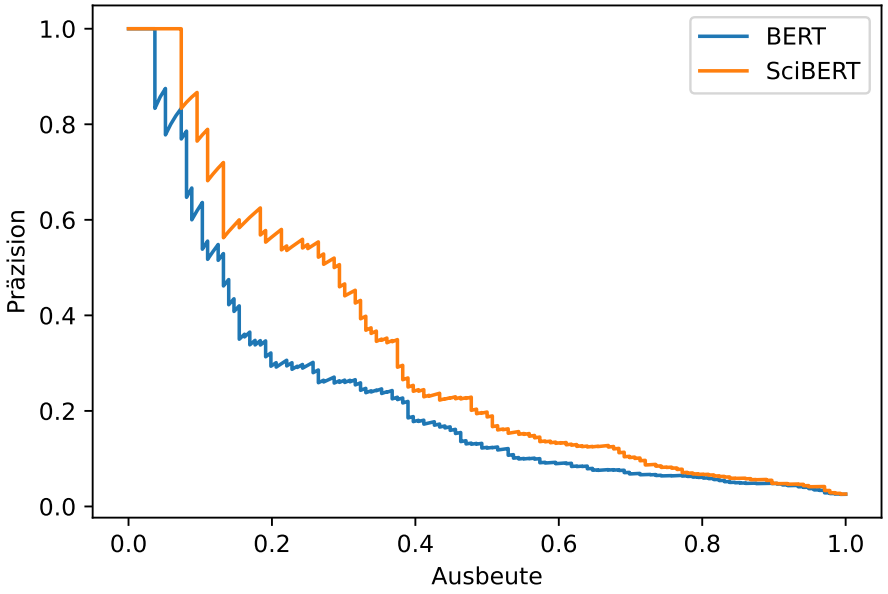

Abbildung 6.5: Präzision-Ausbeute-Kurve für den WARC-Datensatz

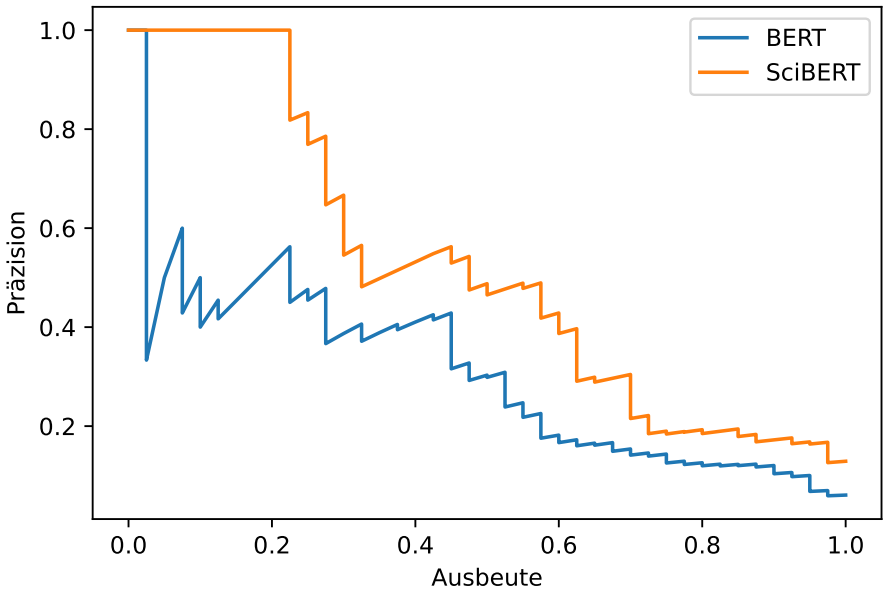

Abbildung 6.6: Präzision-Ausbeute-Kurve für Modis-Datensatz

Tabelle 6.5: Präzision, Ausbeute und F_1-Wert für das Verfahren mit Satzeinbettungen bei optimalen Schwellwerten

Datensatz	BERT			SciBERT		
	P	R	F1	P	R	F1
GANTT	0,200	0,324	0,247	0,232	0,471	0,311
CCHIT	0,102	0,116	0,109	0,135	0,141	0,138
CM1	0,171	0,111	0,135	0,100	0,127	0,112
InfusionPump	0,237	0,237	0,237	0,151	0,542	0,236
WARC	0,243	0,368	0,292	0,453	0,316	0,372
Modis	0,429	0,450	0,439	0,489	0,575	0,529

tensätze, mit Ausnahme von WARC, ist die Ausbeute kleiner als 0,1. Anders als bei dem Verfahren mit Klassifikationsschicht liefert SciBERT auf allen Datensätzen bessere Ergebnisse als BERT, gemessen am F_1-Wert. Um zu überprüfen, ob das Verfahren oder ob die Wahl des Schwellwertes nicht gut funktioniert werden die für die jeweiligen Testdatensätze die optimalen Schwellwerte berechnet. Diese sind in Tabelle 6.5 zu sehen. Für einige Datensätze wurde ein guter Schwellwert bestimmt. der optimale F_1-Wert für den CM1-Datensatz mit SciBERT unterscheidet sich nur um 0,003 vom zuvor bestimmten. Bei manchen Datensätze ist der gewählte Schwellwert sehr schlecht. Für den InfusionPump-Datensatz wurden F_1-Werte unter 0,03 bestimmt, während der optimale Wert für das Verfahren größer als 0,2 ist.

Ein alternatives Verfahren um Satzeinbettungen zu nutzen ist die Clusterbildung, für die in dieser Arbeit der k-Means-Algorithmus verwendet wird. Jede detailliertere Anforderung wird einem Cluster zugeordnet. Für die Zuordnung von Clustern zu groben Anforderungen wird wie beim Verfahren, dass die Satzeinbettungen direkt vergleicht ein Schwellwert verwendet. Um das Potential dieser Methode einschätzen zu können, werden diesmal direkt die bestmöglichen Schwellwerte für die Testdatensätze bestimmt, so dass der F_1-Wert maximal ist. Die Ergebnisse für die optimalen Schwellwerte sind in Tabelle 6.6 zu sehen.

Betrachtet man nur den F_1-Wert, so kann das Verfahren mit Clusterbildung ähnliche Er-

Tabelle 6.6: Präzision, Ausbeute und F_1-Wert für das Verfahren mit Clusterbildung mit optimalen Schwellwerten

Datensatz	BERT			SciBERT		
	P	R	F1	P	R	F1
GANTT	0,148	0,353	0,209	0,203	0,574	0,300
CCHIT	0.065	0.131	0.087	0.072	0.173	0.102
CM1	0.171	0.111	0.135	0.100	0.127	0.112
InfusionPump	0.112	0.489	0.182	0.140	0.573	0.225
WARC	0.181	0.404	0.250	0.303	0.324	0.313
Modis	0.383	0.450	0.413	0.864	0.475	0.613

Tabelle 6.7: Präzision, Ausbeute und F_1-Werte, für die drei Sprachmodellverfahren und existierende Verfahren

Verfahren	GANTT			CM1			Modis		
	P	R	F1	P	R	F1	P	R	F1
Klassifikation	0,110	0,941	0,197	0,045	0,781	0,085	0,201	0,750	0,317
Satzeinbettung	0,232	0,471	0,311	0,100	0,127	0,112	0,489	0,575	0,529
Clustering	0,203	0,574	0,300	0.100	0.127	0.112	0.864	0.475	0.613
S2Trace	0,294	0,541	0,381	0,311	0,483	0,378	-	-	-
LSI-Polysemy	-	-	-	0,34	0,93	0,50	0,33	0,91	0,48

gebnisse liefern, wie das Verfahren, dass nur Satzeinbettungen vergleicht. Allerdings ist der F_1-Wert meistens ein bisschen schlechter. Eine Ausnahme ist hier der Modis-Datensatz für den bei einem optimalen Schwellwert mit SciBERT ein F_1-Wert von 0,613 statt 0,529 erreicht wird. Für die anderen Datensätze ist oft die Ausbeute höher, während die Präzision niedriger ist. Das lässt sich damit erklären, dass detaillierte Anforderungen in ein Cluster aufgenommen, und mit einer groben Anforderung verbunden werden, da die Ähnlichkeit zum Repräsentant des Clusters groß genug ist, obwohl die Ähnlichkeit zu nicht allen detaillierten Anforderungen groß genug wäre. Außer für den CM1-Datensatz werden mit SciBERT bessere Ergebnisse erzielt. Da auch für das Verfahren, das nur die Ähnlichkeit zwischen Satzeinbettungen verwendet, mit SciBERT häufig bessere Ergebnisse gefunden wurden, scheint es vor allem für die Verfahren, die Satzeinbettungen verwenden, ein Vorteil zu sein, ein Sprachmodell zu verwenden, dass auf Texten eines ähnlichem Gebiets vortrainiert wurde.

6.5 Vergleich mit anderen Verfahren

In Tabelle 6.7 sind die Ergebnisse der drei auf Sprachmodellen basierenden Verfahren zusammen mit drei Verfahren zu sehen. Für die beiden Verfahren, die Satzeinbettungen nutzen, wurden jeweils die optimalen Schwellwerte verwendet, sie sind also nur beschränkt vergleichbar. Für das Sprachmodell mit Klassfikationsschicht wurde BERT verwendet, für die Satzeinbettungsverfahren wurde SciBERT verwendet. S2Trace [CWWW19] ist Verfahren, dass Sequenzmuster aus Anforderungen extrahiert und damit Dokumentenvektoren berechnet. LSI-Polysemy [WNLN18] ist ein auf LSI basierendes Verfahren, dass ein neuronales Netz verwendet um Mehrdeutigkeiten aufzulösen.
S2Trace erreicht für GANTT eine ähnliche Ausbeute, wie das Satzeinbettungsverfahren mit Clustering, erzielt dabei aber eine eine um 0,09 höhere Präzision, so dass der F_1-Wert um 0,081 höher ist. Besonders deutlich sind die Unterschiede beim CM1-Datensatz zu sehen. Beide Vergleichsverfahren übertreffen die Sprachmodellverfahren mit Satzeinbettung um mehr als 0,2 für Präzision, Ausbeute und F_1-Wert. Für Modis übertrifft LSI-Polysemy

Tabelle 6.8: Präzision, Ausbeute und F_1-Wert für BERT und TRAIL [MEAH18] und bei der 10-fachen Kreuzvalidierung

	BERT mit Klassifikationsschicht						TRAIL		
	Mittelwert			Gesamt					
Datensatz	P	R	F1	P	R	F1	P	R	F1
Modis	0,138	0,057	0,220	0,131	0,575	0,213	0,659	0,629	0,643

das Klassifikationsverfahren in allen Metriken. Allerdings liefern bei Satzeinbettungsverfahren mit optimal gewählten Schwellwert eine höhere Präzision, was zu einem höheren F_1-Wert führt. Dafür hat LSI-Polysemy mit 0,91 eine deutlich höhere Ausbeute.
Insgesamt liefern die auf Sprachmodellen basierenden Verfahren tendenziell schlechtere Ergebnisse als der Stand der Technik. Sie können allerdings Potenzial haben, wenn der Schwellwert gut bestimmt wird. Die Bestimmung des Schwellwertes wird in dieser Arbeit nur kurz angeschnitten, indem für den Validationsdatensatz ein optimaler Schwellwert bestimmt wird. Die teilweise stark unterschiedlichen Ergebnisse für mit dem Validationsdatensatz erstellten Schwellwerten und den für die Testdatensätze optimalen Schwellwerten legen nahe, dass Validationsdatensatz und Testdatensatz sich teilweise stark unterscheiden. Auch hier kann möglicherweise die Bestimmung des Schwellwerts verbessert werden, indem die Verteilung der Beispiele, aus unterschiedlichen Datensätzen, die den Validationsdatensatz bilden, balanciert wird. So dass die gleiche Anzahl an Beispielen aus jedem Projekt verwendet werden.

6.6 Kreuzvalidierung

Um zu überprüfen wie sich das Sprachmodell verhält, wenn es auf Daten eines Projekts feinangepasst wird und anschließend ungesehene Beispiele des gleichen Projekts als Eingabe erhält, wird eine 10-fache Kreuzvalidierung beispielhaft mit dem Modis-Datensatz durchgeführt. Da SciBERT für die Klassifikationsaufgabe ähnliche Ausgaben wie BERT liefert, wird nur an BERT die 10-fache Kreuzvalidierung durchgeführt. Die in Abschnitt 6.2 vorgestellten Metriken lassen sich auf zwei verschiedene Arten berechnen. Sie können für jedes Teilergebnis einzeln berechnet und anschließend gemittelt werden, oder alle Ausgaben werden zusammengefügt, so dass die entstandene Gesamtausgabe der Ausgabe eines kompletten Datensatzes entspricht.

Das Verfahren mit Klassifikationsschicht erzielt in der Kreuzvalidierung ein schlechteres Ergebnis als das, das mit der Feinanpassung auf anderen Projekten erzielt wird. Ein mögliche Erklärung ist, dass der Modis-Datensatz relativ klein ist und das Sprachmodell eine höhere Epochenanzahl benötigt damit sich ein Trainingserfolg einstellt. Offensichtlich erzielt TRAIL [MEAH18] bessere Werte als das Feinangepasste Sprachmodell.

7 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit sollte ein Verfahren entworfen werden, dass mithilfe von Sprachmodellen Rückverfolgbarkeitverbindungen zwischen natürlichsprachlichen Anforderungen identifiziert. Zunächst wurden Sprachmodelle ausgewählt, die später feinangepasst wurden. Diese waren BERT-base-cased und SciBERT-cased. Aus Zeitgründen wurden nur diese zwei Sprachmodelle gewählt. In zukünftigen Arbeiten könnten bessere Ergebnisse erzielt werden, indem auch andere Sprachmodelle getestet werden. Besonders BERT-large oder ein anderes großes Sprachmodell könnten verwendet werden, sofern genug Zeit und Hardwareressourcen vorhanden sind. Auch Sprachmodelle, die bereits auf ähnlichen Aufgaben feinangepasst sind liefern möglicherweise bessere Ergebnisse indem sie Unterschiede zwischen den Anforderungsdatensätzen durch allgemeineres Wissen ausgleichen. Nach der Wahl der Sprachmodelle wurden verschiedene Verfahren entworfen, die Paare von Anforderungen darauf abbilden, ob zwischen ihnen eine Rückverfolgbarkeitsverbindung existiert. Für das erste Verfahren wurden BERT und SciBERT jeweils mit einer Klassifikationsschicht feinangepasst. Beim zweiten und dritten Verfahren wurden die Sprachmodelle so feinangepasst, dass Satzeinbettungen für Anforderungen mit einer Rückverfolgbarkeitsverbindung möglichst ähnlich sind. Die Ähnlichkeit wurde dabei mit der Kosinus-Ähnlichkeit bestimmt. Das zweite Verfahren vergleicht direkt die Satzeinbettungen von groben und detaillierten Anforderungen miteinander. Beim dritten Verfahren werden mithilfe der Satzeinbettungen der detaillierten Anforderungen mit dem k-Means-Algorithmus Cluster gebildet. Die groben Anforderungen werden anschließend mit einen Repräsentant jedes Clusters verglichen. Ein Problem, dass in dieser Arbeit nicht gelöst wurde, ist das Finden eines geeigneten Schwellwertes, den die Kosinus-Ähnlichkeit übertreffen muss, damit eine Rückverfolgbarkeitsverbindung identifiziert wird. Problematisch war es dabei, dass Training- Validierungs- und Testdatensatz oft sehr unterschiedlich waren. Der Einfluss von veränderten Trainingsdaten wurde nur in Form von zufälligen Weglassen von negativen Beispielen betrachtet. Zusätzlich wurde BERT auf dem CM-1-Datensatz ausgewertet, nachdem es nur auf vier der übrigen fünf Datensätze feinangepasst wurde. Die Sprachmodelle mit Klassifikationsschicht erzielt eine Ausbeute von bis zu 0,96 bei einer eher geringen Präzision von 0,01 bis 0,26. Ob das auf wissenschaftlichen Texten vortrainierte SciBERT verwendet wird oder nicht macht nur einen geringen Unterschied. Abhängig vom Testdatensatz erzielt mal BERT, mal SciBERT leicht bessere Ergebnisse. Für die Verfahren mit Satzeinbettungen hat SciBERT gegenüber BERT allerdings deutliche Verbesserungen gebracht. Kein Verfahren erreicht den Stand der Technik. Allerdings liefern die Satzeinbettungsverfahren für den Modis-Datensatz mit den Satzeinbettungsverfahren mit einen optimalen Schwellwert gute Ergebnisse. Möglichkeiten die Verfahren zu verbessern finden sich vor allem beim Training.

So ist zum Beispiel ein zusätzliches Vortraining auf Anforderungstexten denkbar.

Literaturverzeichnis

[ASP10] Assawamekin, Namfon ; Sunetnanta, Thanwadee ; Pluempitiwiriyawej, Charnyote: Ontology-based multiperspective requirements traceability framework. In: *Knowledge and Information Systems* 25 (2010), Dezember, Nr. 3, 493–522. http://dx.doi.org/10.1007/s10115-009-0259-2. – DOI 10.1007/s10115–009–0259–2. – ISSN 0219–3116 (zitiert auf Seite 9).

[BAPM15] Bowman, Samuel R. ; Angeli, Gabor ; Potts, Christopher ; Manning, Christopher D.: A large annotated corpus for learning natural language inference. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal : Association for Computational Linguistics, September 2015, 632–642 (zitiert auf Seite 13).

[BLC19] Beltagy, Iz ; Lo, Kyle ; Cohan, Arman: SciBERT: A Pretrained Language Model for Scientific Text. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China : Association for Computational Linguistics, November 2019, 3615–3620 (zitiert auf den Seiten 14 und 17).

[CDA+17] Cer, Daniel ; Diab, Mona ; Agirre, Eneko ; Lopez-Gazpio, Inigo ; Specia, Lucia: SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (2017). http://dx.doi.org/10.18653/v1/s17-2001. – DOI 10.18653/v1/s17–2001 (zitiert auf Seite 13).

[CHGZ12] Cleland-Huang, Jane (Hrsg.) ; Gotel, Orlena (Hrsg.) ; Zisman, Andrea (Hrsg.): *Software and Systems Traceability*. London : Springer London, 2012. http://dx.doi.org/10.1007/978-1-4471-2239-5. http://dx.doi.org/10.1007/978-1-4471-2239-5. – ISBN 978–1–4471–2238–8 978–1–4471–2239–5 (zitiert auf Seite 1).

[CWWW19] Chen, Lei ; Wang, Dandan ; Wang, Junjie ; Wang, Qing: Enhancing Unsupervised Requirements Traceability with Sequential Semantics. In: *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, 2019, S. 23–30. – ISSN: 1530-1362 (zitiert auf den Seiten 10 und 36).

[CYK+18] Cer, Daniel ; Yang, Yinfei ; Kong, Sheng-yi ; Hua, Nan ; Limtiaco, Nicole ; John, Rhomni S. ; Constant, Noah ; Guajardo-Cespedes, Mario ; Yuan, Steve ; Tar, Chris ; Sung, Yun-Hsuan ; Strope, Brian ; Kurzweil, Ray: Universal Sentence Encoder. In: *arXiv:1803.11175 [cs]* (2018), April. http://arxiv.org/abs/1803.11175. – arXiv: 1803.11175 (zitiert auf Seite 13).

[DCLT19] Devlin, Jacob ; Chang, Ming-Wei ; Lee, Kenton ; Toutanova, Kristina: BERT: Pre-training of Deep Bidirectional Transformers for Language Un-

derstanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota : Association for Computational Linguistics, Juni 2019, 4171–4186 (zitiert auf den Seiten 1 und 17).

[DSH+20] Du, Tian-bao ; Shen, Guo-hua ; Huang, Zhi-qiu ; Yu, Yao-shen ; Wu, De-xiang: Automatic traceability link recovery via active learning. In: *Frontiers of Information Technology & Electronic Engineering* 21 (2020), August, Nr. 8, 1217–1225. http://dx.doi.org/10.1631/FITEE.1900222. – DOI 10.1631/FITEE.1900222. – ISSN 2095–9230 (zitiert auf Seite 12).

[GCCH17] Guo, Jin ; Cheng, Jinghui ; Cleland-Huang, Jane: Semantically Enhanced Software Traceability Using Deep Learning Techniques. In: *Proceedings of the 39th International Conference on Software Engineering*. Piscataway, NJ, USA : IEEE Press, 2017 (ICSE '17). – ISBN 978–1–5386–3868–2, 3–14 (zitiert auf Seite 10).

[GCHH+12] Gotel, Orlena ; Cleland-Huang, Jane ; Hayes, Jane H. ; Zisman, Andrea ; Egyed, Alexander ; Grünbacher, Paul ; Dekhtyar, Alex ; Antoniol, Giuliano ; Maletic, Jonathan ; Mäder, Patrick: Traceability Fundamentals. Version: 2012. http://dx.doi.org/10.1007/978-1-4471-2239-5. In: Cleland-Huang, Jane (Hrsg.) ; Gotel, Orlena (Hrsg.) ; Zisman, Andrea (Hrsg.): *Software and Systems Traceability*. London : Springer, 2012. – DOI 10.1007/978–1–4471–2239–5. – ISBN 978–1–4471–2239–5, 3–22 (zitiert auf Seite 3).

[Hay99] Haykin, Simon S.: *Neural networks: a comprehensive foundation*. 2. ed., [Nachdr.], internat. ed. Upper Saddle River, NJ : Prentice Hall, 1999 (International edition). – ISBN 978–0–13–908385–3 978–0–13–273350–2. – OCLC: 246168632 (zitiert auf Seite 5).

[HDO03] Hayes, J. H. ; Dekhtyar, A. ; Osborne, J.: Improving requirements tracing via information retrieval. In: *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, 2003, S. 138–147 (zitiert auf den Seiten 7, 8 und 16).

[HDS06] Hayes, Jane H. ; Dekhtyar, Alex ; Sundaram, Senthil K.: Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods. In: *IEEE Trans. Softw. Eng.* 32 (2006), Januar, Nr. 1, 4–19. http://dx.doi.org/10.1109/TSE.2006.3. – DOI 10.1109/TSE.2006.3. – ISSN 0098–5589 (zitiert auf Seite 8).

[HHD09] Holbrook, E. A. ; Hayes, J. H. ; Dekhtyar, A.: Toward Automating Requirements Satisfaction Assessment. In: *2009 17th IEEE International Requirements Engineering Conference*, 2009, S. 149–158. – ISSN: 2332-6441 (zitiert auf den Seiten xi, 1, 7 und 15).

[HHM+04] Hesseler, Alexander ; Hood, Colin ; Missling, Christian ; Stücka, Renate ; Versteegen, Gerhard: *Anforderungsmanagement: Formale Prozesse, Praxiserfahrungen, Einführungsstrategien und Toolauswahl*. 2004 http://link.springer.com/openurl?genre=book&isbn=978-3-642-62388-2. – ISBN 978–3–642–18975–3 978–3–642–62388–2. – OCLC: 913801876 (zitiert auf Seite 3).

[HPL19] Hayes, Jane H. ; Payne, Jared ; Leppelmeier, Mallory: Toward Improved Artificial Intelligence in Requirements Engineering: Metadata for Tracing

Datasets. In: *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, 2019, S. 256–262 (zitiert auf Seite 7).

[LAZCH13] Lohar, Sugandha ; Amornborvornwong, Sorawit ; Zisman, Andrea ; Cleland-Huang, Jane: Improving Trace Accuracy Through Data-driven Configuration and Composition of Tracing Features. In: *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. New York, NY, USA : ACM, 2013 (ESEC/FSE 2013). – ISBN 978–1–4503–2237–9, 378–388 (zitiert auf Seite 9).

[LC20] Lämmel, Uwe ; Cleve, Jürgen: *Künstliche Intelligenz: Wissensverarbeitung – Neuronale Netze*. 5. München : Carl Hanser Verlag GmbH & Co. KG, 2020. http://dx.doi.org/10.3139/9783446463639. http://dx.doi.org/10.3139/9783446463639. – ISBN 978–3–446–45914–4 978–3–446–46363–9 (zitiert auf Seite 5).

[LH19] Loshchilov, Ilya ; Hutter, Frank: Decoupled Weight Decay Regularization. In: *arXiv:1711.05101 [cs, math]* (2019), Januar. http://arxiv.org/abs/1711.05101. – arXiv: 1711.05101 (zitiert auf Seite 26).

[LOG+19] Liu, Yinhan ; Ott, Myle ; Goyal, Naman ; Du, Jingfei ; Joshi, Mandar ; Chen, Danqi ; Levy, Omer ; Lewis, Mike ; Zettlemoyer, Luke ; Stoyanov, Veselin: RoBERTa: A Robustly Optimized BERT Pretraining Approach. In: *arXiv:1907.11692 [cs]* (2019), Juli. http://arxiv.org/abs/1907.11692. – arXiv: 1907.11692 (zitiert auf Seite 17).

[MAP18] Mello, Rodrigo Fernandes d. ; Antonelli Ponti, Moacir: *Machine Learning: A Practical Approach on the Statistical Learning Theory*. Cham : Springer International Publishing, 2018. http://dx.doi.org/10.1007/978-3-319-94989-5. http://dx.doi.org/10.1007/978-3-319-94989-5. – ISBN 978–3–319–94988–8 978–3–319–94989–5 (zitiert auf Seite 4).

[MEAH18] Mills, C. ; Escobar-Avila, J. ; Haiduc, S.: Automatic Traceability Maintenance via Machine Learning Classification. In: *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, S. 369–380. – ISSN: 2576-3148 (zitiert auf den Seiten xiii, 11 und 37).

[Mit97] Mitchell, Tom M.: *Machine Learning*. New York : McGraw-Hill, 1997 (McGraw-Hill series in computer science). – ISBN 978–0–07–042807–2 (zitiert auf Seite 4).

[MRS08] Manning, Christopher D. ; Raghavan, Prabhakar ; Schütze, Hinrich: *Introduction to information retrieval*. New York : Cambridge University Press, 2008. – ISBN 978–0–521–86571–5. – OCLC: ocn190786122 (zitiert auf Seite 4).

[ORB+20] Ostendorff, Malte ; Ruas, Terry ; Blume, Till ; Gipp, Bela ; Rehm, Georg: Aspect-based Document Similarity for Research Papers. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online) : International Committee on Computational Linguistics, Dezember 2020, 6194–6206 (zitiert auf Seite 14).

[PVG+11] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830 (zitiert auf Seite 25).

[RG19] Reimers, Nils ; Gurevych, Iryna: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 11 2019 (zitiert auf den Seiten 13, 17 und 26).

[RMJ+20] Reback, Jeff ; McKinney, Wes ; Jbrockmendel ; Bossche, Joris Van D. ; Augspurger, Tom ; Cloud, Phillip ; Gfyoung ; Sinhrks ; Hawkins, Simon ; Roeschke, Matthew ; Klein, Adam ; Petersen, Terji ; Tratner, Jeff ; She, Chang ; Ayd, William ; Naveh, Shahar ; Garcia, Marc ; Schendel, Jeremy ; Hayden, Andy ; Saxton, Daniel ; Jancauskas, Vytautas ; McMaster, Ali ; Battiston, Pietro ; Seabold, Skipper ; Patrick ; Dong, Kaiqi ; Chris-B1 ; H-Vetinari ; Hoyer, Stephan ; Gorelli, Marco: *pandas-dev/pandas: Pandas 1.1.5*. `http://dx.doi.org/10.5281/ZENODO.4309786`. Version: Dezember 2020 (zitiert auf Seite 25).

[SCH12] Shin, Yonghee ; Cleland-Huang, Jane: A comparative evaluation of two user feedback techniques for requirements trace retrieval. In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12*. Trento, Italy : ACM Press, 2012. – ISBN 978–1–4503–0857–1, 1069 (zitiert auf Seite 7).

[SHDH10] Sundaram, Senthil K. ; Hayes, Jane H. ; Dekhtyar, Alex ; Holbrook, E. A.: Assessing traceability of software engineering artifacts. In: *Requirements Engineering* 15 (2010), September, Nr. 3, 313–335. `http://dx.doi.org/10.1007/s00766-009-0096-6`. – DOI 10.1007/s00766–009–0096–6. – ISSN 1432–010X (zitiert auf Seite 7).

[SRT20] Sanjeev, M. M. ; Ramalingam, B. ; T.K, S. K.: Realtime Semantic Similarity Analysis of Bulk Outlook Emails Using BERT. In: *2020 International Conference on Advances in Computing, Communication Materials (ICACCM)*, 2020, S. 89–94. – ISSN: 2642-7354 (zitiert auf Seite 13).

[SV20] Schlutter, Aaron ; Vogelsang, Andreas: Trace Link Recovery using Semantic Relation Graphs and Spreading Activation. (2020), Juni. `http://dx.doi.org/10.14279/DEPOSITONCE-10207`. – DOI 10.14279/DEPOSITONCE–10207. – Publisher: Technische Universität Berlin (zitiert auf Seite 12).

[WDS+20] Wolf, Thomas ; Debut, Lysandre ; Sanh, Victor ; Chaumond, Julien ; Delangue, Clement ; Moi, Anthony ; Cistac, Pierric ; Rault, Tim ; Louf, Remi ; Funtowicz, Morgan ; Davison, Joe ; Shleifer, Sam ; Platen, Patrick von ; Ma, Clara ; Jernite, Yacine ; Plu, Julien ; Xu, Canwen ; Le Scao, Teven ; Gugger, Sylvain ; Drame, Mariama ; Lhoest, Quentin ; Rush, Alexander: Transformers: State-of-the-Art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online : Association for Computational Linguistics, 2020, 38–45 (zitiert auf Seite 25).

[WNB18] Williams, Adina ; Nangia, Nikita ; Bowman, Samuel: A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana : Association for Computational Linguistics, Juni 2018, 1112–1122 (zitiert auf Seite 13).

[WNLN18] Wang, W. ; Niu, N. ; Liu, H. ; Niu, Z.: Enhancing Automated Requirements Traceability by Resolving Polysemy. In: *2018 IEEE 26th International*

Requirements Engineering Conference (RE), 2018, S. 40–51 (zitiert auf den Seiten 11 und 36).

[YHZ+20] Yang, Xi ; He, Xing ; Zhang, Hansi ; Ma, Yinghan ; Bian, Jiang ; Wu, Yonghui: Measurement of Semantic Textual Similarity in Clinical Texts: Comparison of Transformer-Based Models. In: *JMIR Medical Informatics* 8 (2020), November, Nr. 11, e19735. http://dx.doi.org/10.2196/19735. – DOI 10.2196/19735. – ISSN 2291–9694 (zitiert auf Seite 12).

[ZCS17] Zhao, T. ; Cao, Q. ; Sun, Q.: An Improved Approach to Traceability Recovery Based on Word Embeddings. In: *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, 2017, S. 81–89 (zitiert auf den Seiten 10 und 11).

[ZKZ+15] Zhu, Y. ; Kiros, R. ; Zemel, R. ; Salakhutdinov, R. ; Urtasun, R. ; Torralba, A. ; Fidler, S.: Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In: *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, S. 19–27. – ISSN: 2380-7504 (zitiert auf Seite 17).

[ZSCH10] Zou, Xuchang ; Settimi, Raffaella ; Cleland-Huang, Jane: Improving automated requirements trace retrieval: a study of term-based enhancement methods. In: *Empirical Software Engineering* 15 (2010), April, Nr. 2, 119–146. http://dx.doi.org/10.1007/s10664-009-9114-z. – DOI 10.1007/s10664–009–9114–z. – ISSN 1573–7616 (zitiert auf den Seiten 9 und 16).