Karlsruher Institut für Technologie

Software Design and Quality

Komposition von Trace Link Recovery Ansätzen

Bachelor-Arbeit von

Ian Winter

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

Erstgutachter:	Prof. Dr.-Ing. Anne Koziolek
Zweitgutachter:	Prof. Dr. Ralf Reussner
Betreuender Mitarbeiter:	M.Sc. Jan Keim

16. Juni 2021 – 15. Oktober 2021

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Zusammenfassung

Das Erstellen von Trace-Links, die beispielsweise Dokumentationen mit Entwurfsmodellen verknüpfen, ist ein wertvoller Bestandteil der Softwareentwicklung. Da ein manuelles Herauslesen der Trace-Links oft nicht praktikabel ist, sollte dieser Prozess automatisiert werden. Es existieren schon viele verschiedene Ansätze der Trace-Link-Recovery, welche jedoch unterschiedliche Stärken und Schwächen aufweisen. In dieser Arbeit wird untersucht, ob die Stärken unterschiedlicher Recovery-Ansätze durch Komposition verknüpft werden können, um ggf. die Schwächen auszugleichen. Dazu habe ich mehrere einfache Kompositionen implementiert und deren Ergebnisse ausgewertet. Die Metriken Ausbeute, Präzision, F1- und F2-Werte verschiedener Kompositionen wurden in drei Fallstudien ermittelt und mit denen der Basis-Ansätze verglichen. Einige Kompositionen schneiden dabei bis zu 12% besser ab als der Durchschnitt der Basis-Ansätze dieser Kompositionen und bis zu 2% besser als der beste Einzelansatz.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Schon lange ist bekannt, dass zu guter Softwareentwicklung mehr gehört als reines Programmieren [5]. Anforderungen erheben, Softwarearchitektur entwerfen, testen, dokumentieren und vieles Weitere sind wichtig, um auch bei größeren Projekten gute Softwarequalität liefern zu können.

Diese einzelnen Schritte reden oft über die gleichen Elemente, aber benennen diese anders oder umschreiben sie nur. Es entstehen Überlappungen und damit zwischen den Artefakten ein Netz von Abhängigkeiten und Verweisen. Das Netz der Abhängigkeiten zwischen den Artefakten ist fast nie explizit ausgearbeitet, sondern versteckt sich z.B. in der natürlichen Sprache der Anforderungen oder den Beschreibungen der Entwurfsmodelle. Es ist somit sehr mühsam und zeitaufwendig, manuell diese Abhängigkeiten (auch *Trace-Links* genannt) zu finden. Noch weiter erschwert wird diese Aufgabe, weil sich die Artefakte ändern können. Trotzdem sind diese Abhängigkeiten hilfreich. Zum Beispiel die Abhängigkeiten zwischen Anforderungen und Quellcode - mit ihnen kann nachvollzogen werden, welche Anforderungen schon realisiert wurden. Bei wiederauftretenden Anforderungen können in alten Projekten schnell die Code-Teile gefunden werden, die für die Implementierung der Anforderung zuständig waren. Diese können dann extrahiert und wiederverwendet werden. Speziell bei der Entwicklung sicherheitskritischer Systeme, bei denen Normen und Richtlinien eingehalten werden müssen, ist es von enormem Vorteil (oder gar Pflicht, siehe DIN-Norm), durch diese Abhängigkeiten zeigen zu können, dass die Normen und Anforderungen eingehalten und implementiert wurden. In der Auswirkungsanalyse können Trace-Links verwendet werden, um bei Änderungen der Anforderungen zu sehen, welche anderen Artefakte von der Änderung betroffen sind. Dadurch kann der Aufwand, diese Änderung vorzunehmen, besser abgeschätzt werden. In der Abdeckungs- & Projektstatusanalyse kann aus Anforderungen, welche nicht mit dem Code verbunden werden können, abgeleitet werden, dass diese noch nicht implementiert wurden. Auch der Status des Systems kann so leicht ermittelt werden, denn Trace-Links können den Entwicklungsstand eines Projektes über die Zeit hinweg widerspiegeln und somit helfen, den Überblick zu behalten. Diese Daten können Zusammenhänge persistieren. Außerdem kann die Trace-Link-Gewinnung helfen, Konsistenz zwischen formalen und informalen Software-Artefakten herzustellen und beizubehalten, wie von Jan Keim et al. 2019 beschrieben [10].

In dieser Arbeit soll es vor allem um die Trace-Links zwischen Entwurfsmodellen (genauer Palladio-Component-Models) und deren Dokumentation gehen. Mit diesen Links ist es möglich, die Dokumentation mit dem zugehörigen Modell zu vergleichen. Da beide unterschiedliche Schwerpunkte und Darstellungen haben, kann ein Leser besser nachvollziehen, warum spezielle Entwurfsentscheidungen getroffen wurden. Das verringert beispielsweise die Zeit, die ein Programmierer braucht, um sich in einem Projekt zurechtzufinden, wenn

er oder sie neu in das Entwicklungsteam einsteigt, oder sich nach einiger Zeit wegen neuer gewünschter Anforderungen einarbeiten muss.

Weil es mühsam ist, Trace-Links manuell zu ermitteln, sie aber trotzdem sehr nützlich sind, ist es äußerst effektiv, diese automatisch generieren zu lassen. Das Feld der Trace-Link-Recovery beschäftigt sich mit diesem Problem. Es werden mittels unterschiedlichster Ansätze die Trace-Links eines Projektes ermittelt. Durch das automatische Erstellen der Trace-Links können alle oben stehenden Vorteile genutzt werden, ohne jedoch auf zeitaufwendiges, manuelles Erstellen der Links zurückgreifen zu müssen. Eine besondere Herausforderung bildet dabei die Diversität, sowie die oft unformale Natur der Artefakte. Manche sind in natürlicher Sprache formuliert, andere in Programmiersprachen oder gar visuell dargestellt. Für jede Darstellung müssen somit andere Faktoren betrachtet werden.

Es existieren schon einige Ansätze zur Trace-Link-Gewinnung, die von unterschiedlichen Forschenden vorgeschlagen und getestet wurden. Verschiedene Ansätze haben unterschiedliche Vor- und Nachteile. Viele stützen sich auf unterschiedliche Ähnlichkeitsmetriken von Texten. Meist wird jedoch nur eine einzige verwendet. Diese Metriken liefern jedoch häufig orthogonale Ergebnisse [13]. Somit lässt sich vermuten, dass es hilfreich ist, mehrere Metriken zu kombinieren. Womöglich könnte es sogar zu Verbesserungen kommen, wenn völlig unterschiedliche Ansätze zusammenarbeiten, um die Trace-Links zu extrahieren. Um die Gewinnung von Trace-Links zu verbessern, soll in dieser Arbeit untersucht werden, auf welche Weise sich Trace-Link-Recovery-Ansätze verknüpfen lassen - also wie sich unterschiedliche Kompositionen mehrerer Ansätze auf das Ergebnis auswirken.

In dieser Arbeit soll es um die Verknüpfung von Entwurfsmodellen mit deren Dokumentation gehen. Dieses Ziel wird auch vom ArDoCo-Framework [2] verfolgt. Es bietet ein Rahmenwerk, in dem verschiedene Ansätze eingebaut und ausgeführt werden können. Außerdem ist dort schon ein Trace-Link-Recovery-Ansatz implementiert. In dieser Arbeit sollen weitere Ansätze in das ArDoCo-Framework eingebaut und dort kombiniert werden. Die Arbeit soll dabei helfen, Recovery-Ergebnisse systematisch zu verbessern, indem sich die Stärken einzelner Recovery-Ansätze durch Komposition mit anderen ergänzen.

1.1. Forschungsfrage

Aus den vorangegangenen Überlegungen ergibt sich nun folgende Forschungsfrage:

1. Wie verändert sich die Güte der Trace-Link-Recovery durch verschiedene Kompositionen von Trace-Link-Recovery-Ansätzen im Vergleich zu den Einzelansätzen?

Die Güte bestimme ich mittels des Vergleichs der F1- und F2-Werte der Kompositionen und deren atomaren Bestandteilen. Auch wenn der F1- und F2-Wert ausschlaggebend sind, werde ich auch auf die Veränderung von der Ausbeute und der Präzision eingehen.

Außerdem will ich noch eine zweite untergeordnete Forschungsfrage untersuchen:

2. Wie effizient ist eine Komposition in der Implementierung und der Ausführung im Vergleich zu den einzelnen Ansätzen?

Dazu soll der grobe Implementierungsaufwand der Komposition in Relation zu dem der einzelnen Ansätze gestellt werden. Das gleiche gilt für die Ausführungszeit.

2. Grundlagen

In diesem Kapitel werde ich auf einige Grundlagenthemen eingehen, die dabei helfen können, die folgenden Kapitel nachzuvollziehen. Als erstes werde ich in Abschnitt 2.1 die Grundlagen der Trace-Link-Recovery darstellen. Daraufhin werde ich kurz erklären, was Palladio und sogenannte Palladio-Component-Models sind, da ich diese als Entwurfsmodelle verwenden werde (Abschnitt 2.2). Darauf folgen noch kleinere Abschnitte zu unterschiedlichen Themen, die in meiner Ausarbeitung am Rande erwähnt werden, wie z.B. Deep-Learning.

2.1. Trace-Link-Recovery

Trace-Links sind Abhängigkeiten oder Verweise zwischen Software-Artefakten. Wenn ein Satz in der Dokumentation eine Komponente im Entwurfsmodell erwähnt, dann besteht beispielsweise ein Trace-Link zwischen diesem Satz in der Dokumentation und der Komponente im Entwurfsmodell. Die Trace-Link-Recovery beschäftigt sich damit, diese Trace-Links zu erkennen.

Leider werden jedoch oft nicht exakt die selben Namen bzw. Bezeichnungen verwendet, was die Trace-Link-Gewinnung erschwert. Vor allem, da sich die Software-Artefakte stark in ihrer Form unterscheiden können, ist es schwer, automatisch zu erkennen, wo ein Zusammenhang besteht und wo nicht.

In dieser Arbeit wird es ausschließlich um Trace-Links zwischen Entwurfsmodellen und deren Dokumentation gehen. Die Trace-Links bestehen dann aus einem Tupel aus Dokumenten-Id und Modellelement-Id. Die Dokumenten-Id ist die Satznummer in der Dokumentation, während die Modellelement-Id eine Komponente des Entwurfsmodells eindeutig kennzeichnet. Durch diese beiden Angaben kann eine Verbindung zwischen diesem Satz in der Dokumentation und dem Element im Entwurfsmodell dargestellt werden.

Zusätzlich fügen die Trace-Link-Recovery-Ansätze den Trace-Links noch einen weiteren Wert hinzu. Dabei handelt es sich um die Konfidenz. Diese gibt an, wie wahrscheinlich der Ansatz die Korrektheit dieses Trace-Links einstuft. Sie wird durch einen Wert aus dem Intervall von 0 bis 1 dargestellt.

Beispielsweise könnte ein Ansatz einen Trace-Link finden, bei dem ein Komponentenname direkt in einem Satz auftaucht. Somit ist es sehr wahrscheinlich, dass es sich hierbei auch wirklich um diese Komponente handelt und der Ansatz fügt dementsprechend einen hohen Wahrscheinlichkeitswert an. In einem anderen Fall wird lediglich das Wort „hash“ in der Dokumentation verwendet, das oft in Verbindung mit Passwortspeicherung verwendet wird. Deswegen generiert der Ansatz auch hier einen Trace-Link zwischen diesem Satz und einer Passwortspeicherkomponente. Jedoch ist diese Verbindung nicht so

eindeutig wie zuvor, da es noch andere Anwendungen von Hashes gibt. Somit weist der Ansatz diesem Trace-Link nur eine geringe Wahrscheinlichkeit zu.

Einige konkrete Trace-Link-Recovery-Ansätze werde ich in Kapitel 3 zu verwandten Arbeiten aufzeigen. Die in dieser Arbeit verwendeten Ansätze werde ich separat in dem dazugehörigen Kapitel 5 „Atomare Recovery-Ansätze“ beschreiben.

2.2. Palladio

Palladio ist ein Simulator für Software-Architekturen [14]. Es soll Software schon auf der Modellebene analysieren, um auf mögliche Probleme bei Effizienz, Skalierbarkeit oder Verfügbarkeit aufmerksam zu machen. Palladio soll Software-Entwicklern helfen, qualitativ hochwertige Software zu erarbeiten, indem Entwurfsfehler schon in der Entwurfsphase und somit frühzeitig erkannt werden.

In meiner Ausarbeitung werde ich Palladio-Component-Models (PCM) verwenden. Das sind Komponenten-Modelle, die von Palladio analysiert werden können. Hier wird jedoch die Simulation ebenso wie Palladio keine große Rolle spielen. Ich werde lediglich Palladio-Component-Models als Entwurfsmodelle verwenden. Dann können Trace-Links zwischen den Palladio-Component-Models und deren Dokumentation mittels Trace-Link-Recovery-Verfahren extrahiert werden.

2.3. Deep-Learning

Deep-Learning ist eine Art der Datenverarbeitung, bei der ein künstliches neuronales Netzwerk (KNN) aufgebaut und anhand von Beispieldaten trainiert wird. Dieses Verfahren wurde inspiriert durch das menschliche Gehirn und dessen Aufbau aus Neuronen. Solche Ansätze werden oft verwendet, wenn es schwer ist, klare Regeln zur Datenverarbeitung aufzustellen. Beispielsweise bei der Gesichtserkennung in Bildern. Mathematisch zu definieren, was in einem Bild ein Gesicht ausmacht, ist schwer. Stattdessen werden viele Bilder mit schon vorher markierten Gesichtern als Trainingsdaten verwendet. Im besten Fall kann durch das Training die Gesichtserkennung in der Struktur des Netzes und den Gewichten codiert werden und das Netzwerk kann unbekannte Bilder verarbeiten.

3. Verwandte Arbeiten

Viele Forschungsarbeiten beschäftigen sich vor allem mit Trace-Link-Recovery-Ansätzen, die versuchen, Anforderungen und Programmcode miteinander zu verknüpfen. Dabei gibt es viele Herangehensweisen, wie zum Beispiel Deep-Learning, die Betrachtung als Optimierungsproblem oder das Verwenden von Information-Retrieval-Ansätzen. In diesem Kapitel werde ich zuerst einige Trace-Link-Recovery-Ansätze aufzeigen und erläutern, was ihre Stärken und Schwächen sind. Danach werde ich auf einige Methoden der Komposition einzelner Ansätze eingehen, bis hin zu einer Arbeit, die verschiedene Informationsquellen mit einem auf dem „Satz von Bayes“ basierendes hierarchisches Netzwerk verknüpft.

3.1. Trace-Link-Recovery

Es gibt schon einige Ansätze der Trace-Link-Recovery, die alle ihre Stärken und Schwächen haben. Vor allem die oft unformale Natur der Dokumentationstexte in natürlicher Sprache erschwert die Extraktion der Trace-Links. Bei Trace-Links ist es außerdem schwer, genau zu definieren, was einen Trace-Link ausmacht. Es ist manchmal unklar, ob ein Trace-Link vorliegt oder nicht. Somit kann hier versucht werden, anhand von Trainingsdaten ein künstliches neuronales Netz zu trainieren, welches dann neue Projekte nach Trace-Links durchsuchen kann. Ein Vorteil hierbei ist, dass dieses Netz nur einmal trainiert werden muss und dann vergleichsweise schnell und effizient auf neuen Daten ausgeführt werden kann. Ein Nachteil hingegen ist, dass es oft schwer zu erkennen ist, ob das Netz wirklich die gewünschte Funktionalität erlernt hat. Es kann nämlich dazu kommen, dass das Netz etwas ganz Anderes gelernt hat, was zufällig für die Trainingsdaten ein gutes Ergebnis liefert. Außerdem hängt der Lernerfolg stark von der Qualität und der Menge an Daten ab. Für gute Ergebnisse sind oft immense Datenmengen nötig, die je nach dem manuell erstellt werden müssen.

Die Forschungsarbeit „Semantically Enhanced Software Traceability Using Deep Learning Techniques“ von Jin Guo et al. beschäftigt sich beispielsweise mit der Gewinnung von Trace-Links durch Deep-Learning [7]. Zusätzlich zum Ansatz des Deep-Learnings versuchten sie weiteres Wissen über die Domäne einfließen zu lassen, in der das Softwareprojekt angesiedelt ist. Informationen, die von Domänenexperten zusammengetragen wurden, sollten dem KNN helfen, Trace-Links zu erkennen. Sie konnten damit im Vergleich zu zwei anderen Recovery-Ansätzen ein deutlich verbessertes Ergebnis erzielen.

Optimierungsprobleme hingegen versuchen die Eingabedaten bezüglich einer Funktion zu maximieren, zu minimieren oder nahe an einen speziellen Wert zu bringen. Das Training eines KNNs kann beispielsweise ein Optimierungsproblem sein, bei dem der der Fehlerwert minimiert werden soll. Um ein solches Problem zu lösen, gibt es verschiedens-

te Möglichkeiten. Welche am besten ist, hängt stark von den Rahmenbedingungen der Funktion und der Eingabemenge ab.

Die Forschungsarbeit „Multi-Objective Information Retrieval-Based NSGA-II Optimization for Requirements Traceability Recovery“ von Danissa V. Rodriguez et al. hat sich damit beschäftigt, ob Opimierungsproblem-Ansätze auch für die Trace-Link-Recovery genutzt werden können [15]. Genauer verwendeten sie den NSGA-II Algorithmus und eine Textähnlichkeitsmetrik als zu optimierende Funktion. Der NSGA-II-Opimierungsproblem-Ansatz ist ein evolutionsbasierter Ansatz. Auch dieser wurde durch die Natur inspiriert. Ganz getreu dem Motto „Survival of the fittest“ müssen immer wieder verschiedene potentielle Lösungen gegeneinander „antreten“. Die welche bezüglich der zu optimierenden Funktion besser abschneidet, kommt in die nächste Runde, während die andere verworfen wird. Analog zur sexuellen Fortpflanzung werden nun die verbleibenden Werte miteinander Kombiniert. Diese treten daraufhin wieder gegeneinander an und der Prozess wiederholt sich für eine festgelegte Anzahl von Iterationen. Doch genau darin liegt auch eine mögliche Schwäche, auf die auch die Forschenden hinwiesen. Sie wählten eine Anzahl von Iterationen, die für die gegebenen Projekte das beste Ergebnis erzielte. Dadurch leidet jedoch womöglich die Generalisierbarkeit. Die Fehlerwerte schwankten über die Iterationen hinweg und es war keine klare Konvergenz zu erkennen. Somit bleibt unklar, ob dieses Verfahren im Allgemeinen sinnvoll anwendbar ist.

Das Feld der Information-Retrieval (IR) beschäftigt sich hingegen damit, (komplexe) Inhalte aus Daten zu extrahieren. Vor allem im World Wide Web werden viele IR-Ansätze verwendet um aus riesigen Datenmengen die gewünschten Informationen zu extrahieren [6]. Doch auch zur Trace-Link-Gewinnung können diese Techniken helfen. Die Forschungsarbeit „An Empirical Study on Recovering Requirement-to-Code Links“ von Zhang et al. befasst sich beispielsweise damit, Trace-Links zwischen Anforderungen und Programmcode zu gewinnen [18]. Die Forschungsgruppe erkannte, dass andere IR-basierte Ansätze schon einige Trace-Links extrahieren können, jedoch auch noch einige Mängel vorweisen. Beispielsweise fand die Gruppe heraus, dass in Anforderungen und Code oft nicht die gleichen Worte vorkommen, diese jedoch meist Synonyme sind. Außerdem steckt vor allem in Anforderungen viel unnötige Information. Mit diesen Erkenntnissen bauten sie in ihrem eigenen Trace-Link-Recovery-Ansatz einige Mechanismen ein, um diese Probleme zu lösen.

Bei der Dokumentation fokussiert sich ihr Ansatz insbesondere auf das Verb und das Objekt. In diesen steckt meist der größte und zuverlässigste Informationsgehalt. Außerdem werden die Worte auf ihren Wortstamm gekürzt um so einfacher gleiche Worte in unterschiedlichen grammatischen Fällen erkennen zu können. Diese Mechanismen konnten die Effektivität des Ansatzes deutlich verbessern.

Dieser Ansatz ist vor allem sehr interessant, da ich in dieser Arbeit einen auf der Arbeit von Zhang et al. basierenden Recovery-Ansatz verwenden werde. Diesen Ansatz werde ich in meiner Ausarbeitung mit dem *Zhang-Ansatz* betiteln. Der größte Unterschied ist, dass der Ursprüngliche Ansatz Anforderungen mit Programmcode verknüpft, während der hier Verwendete Entwurfsmodelle mit deren Dokumentation verknüpft.

Wenn ich auch keine der oben genannten Recovery-Ansätze in dieser Arbeit direkt ver-

wenden werde, bilden sie dennoch eine wichtige Grundlage - auch für potentielle weitere Arbeiten. Damit sich die Stärken unterschiedlicher Ansätze durch Kompositionen ergänzen können, muss es zwangsläufig erst einmal verschiedene Ansätze geben, die auf unterschiedliche Weisen arbeiten.

3.2. Komposition von einzelnen Ansätzen

So wie bei den vorangegangenen Trace-Link-Recovery-Ansätzen wird meist nur eine bestimmte Herangehensweise der Trace-Link-Generierung verwendet. Es wurde jedoch schon gezeigt, dass es viele Anwendungsfälle gibt, bei denen es hilfreich ist, verschiedene Ansätze zu kombinieren, um bessere Ergebnisse zu erhalten [8].

Beispielsweise bei Wettermodellen konnten signifikant bessere Prognosen abgegeben werden, wenn die Ergebnisse mehrerer, verschiedener Berechnungsmodelle kombiniert wurden. Die Komposition aus den Einzelansätzen war nicht nur besser als deren Durchschnitt, sondern sogar besser als der beste Einzelansatz [8]. Selbst „schlechte“ Modelle können einen wertvollen Beitrag leisten, solange sie nicht immer und in jeder Hinsicht schlechter als der Durchschnitt sind. Auch ist es nicht zielführend, statt des Multimodells einfach das beste Einzelmodell zu verwenden, da es oft kein universell bestes Einzelmodell gibt. Eine Herangehensweise kann in einem Fall gut sein und in einem anderen wiederum nicht. Durch eine geschickte Komposition können die Stärken verschiedener Ansätze zu einem besseren Gesamtergebnis beitragen.

Auch in der Informatik gibt es Paradigmen, die dieses Prinzip ausnutzen. Beim N-Version-Programming wird das gleiche Problem von vielen verschiedenen Ansätzen gelöst [3] [4]. Danach wählt ein Entscheider ein Ergebnis aus, dass das Problem für diese Eingabe am besten löst. Auch hier können Ansätze zum Ergebnis beitragen, obwohl sie vielleicht schlechter abschneiden als der Durchschnitt, wenn sie für spezielle Eingaben sehr gut geeignet sind.

Ebenfalls hilft es, ein komplexes Problem in einfachere Teilprobleme zu zerlegen, getreu dem Teile-und-Herrsche-Paradigma [16]. Auch die Trace-Link-Recovery kann als komplexes Problem betrachtet werden. Bei einigen Ansätzen wird die Trace-Link-Gewinnung in verschiedene Teilprobleme aufgeteilt, wie bei einigen IR-Ansätzen (Füllwortentfernung, Wortstammbildung, Synonymsuche, Vergleich). Auf einer höheren Ebene könnte man auch einzelne Trace-Link-Recovery-Ansätze als Löser für Teilprobleme ansehen, die dann zusammengeführt werden können. Beispielsweise könnte ein Ansatz das Teilproblem der Gewinnung von Trace-Links mit Textähnlichkeit lösen, während ein anderer das Teilproblem der Gewinnung von Trace-Links mit Domäneninformationsüberlappung löst. Viele Ansätze mit orthogonalen Lösungen ergäben so zusammen eine gesamtheitliche Lösung.

Die Forschungsarbeit „Improving the Effectiveness of Traceability Link Recovery using Hierarchical Bayesian Networks“ von Kevin Moran et al. untersucht einen ähnlichen Ansatz [12]. Die Forschungsgruppe hat verschiedene Informationsquellen durch ein hier-

archisches Netzwerk basierend auf dem Satz von Bayes verknüpft. Der Satz von Bayes ist ein Werkzeug der Statistik. Er passt die Wahrscheinlichkeit, dass ein Ereignis eintritt, an, sobald neue Informationen hinzugefügt werden. Die Forschenden verwendeten sowohl IR-Ansätze, Machine-Learning (ML), Experten-Feedback, als auch transitive Beziehungen zwischen Artefakten. Diese spezielle Art der Komposition generierte zuverlässigere Ergebnisse als IR- oder ML-Ansätze individuell.

Die Komposition im Falle von Kevin Moran et al. war bereits verhältnismäßig komplex aufgebaut. In dieser Arbeit werde ich vor allem einige grundlegende Komposition wie Mengenvereinigung oder Schnittmenge betrachten. Jedoch weist die Arbeit von Kevin Moran et al. gekonnt darauf hin, dass es durchaus nützlich ist, verschiedene Metriken zu kombinieren, um ein besseres Ergebnis zu erhalten.

4. Vorgehensweise

In dieser Arbeit will ich automatisiert Trace-Links aus Software-Projekten extrahieren, indem ich verschiedene Trace-Link-Recovery-Ansätze kombiniere. Um herauszufinden, wie sich diese Kompositionen der Recovery-Ansätze auf die Güte der extrahierten Trace-Links auswirken und um damit die in Kapitel 1.1 gestellten Forschungsfragen zu beantworten, werde ich drei Recovery-Ansätze auf unterschiedliche Weisen komponieren. Die Kompositionen sollen dabei sehr generisch gehalten werden und wenig Annahmen über die zugrundeliegenden Recovery-Ansätze treffen. Dadurch könnten zukünftig andere Ansätze evaluiert werden, die in dieser Ausarbeitung nicht betrachtet werden. Ich werde als minimale Annahme davon ausgehen, dass Recovery-Ansätze eine Menge an Trace-Links liefern. Diese Trace-Links können durch ein Trippel von Dokumenten-Id, Modellelement-Id und einer Wahrscheinlichkeit dargestellt werden. Ein solcher Trace-Link gibt somit an, mit welcher Wahrscheinlichkeit sich ein Satz aus der Dokumentation auf ein Modellelement des Entwurfmodells bezieht. Dabei ist die Wahrscheinlichkeit kein wirklicher Bestandteil des Trace-Links, sondern nur eine Aussage des Recovery-Ansatzes darüber, wie sicher diese Zuordnung eingeschätzt wurde.

Eine Komposition verknüpft nun mehrere Mengen von gefundenen Trace-Links zu einer einzigen, wie in Abbildung 4.1 dargestellt. Diese ist dann das Ergebnis dieser Komposition und kann als solches evaluiert werden. So lassen sich Standardkompositionen wie beispielsweise die Mengenvereinigung definieren. Danach soll es möglich sein, aus diesen einfachen Kompositionen komplexere aufzubauen. Kompositionen und atomare Recovery-Ansätze liefern den gleichen Ergebnis-Typ (eine Menge aus Trace-Links). Somit können Kompositionen auch die Ergebnisse anderer Kompositionen als Eingabe verwenden. Bei der Umsetzung implementieren Recovery-Ansätze und auch Kompositionen die gleiche Schnittstelle. Auf diese Weise können Kompositionen jeder Art flexibel erstellt und über eine einheitliche Schnittstelle angesprochen werden.

In meiner Ausarbeitung werde ich zunächst in Kapitel 5 kurz auf die von mir verwendeten atomaren Recovery-Ansätze *Pipeline*, *Zhang* und *Naive* eingehen. Anschließend folgt eine Beschreibung der von mir erstellten Kompositionen in Kapitel 6. Nachdem alle Grundbausteine vorgestellt wurden, werde ich in Kapitel 7 auf die Auswertungsmethodik und die Ergebnisse der Auswertung eingehen und diese darstellen. Als letztes folgt die Beantwortung der beiden Forschungsfragen.

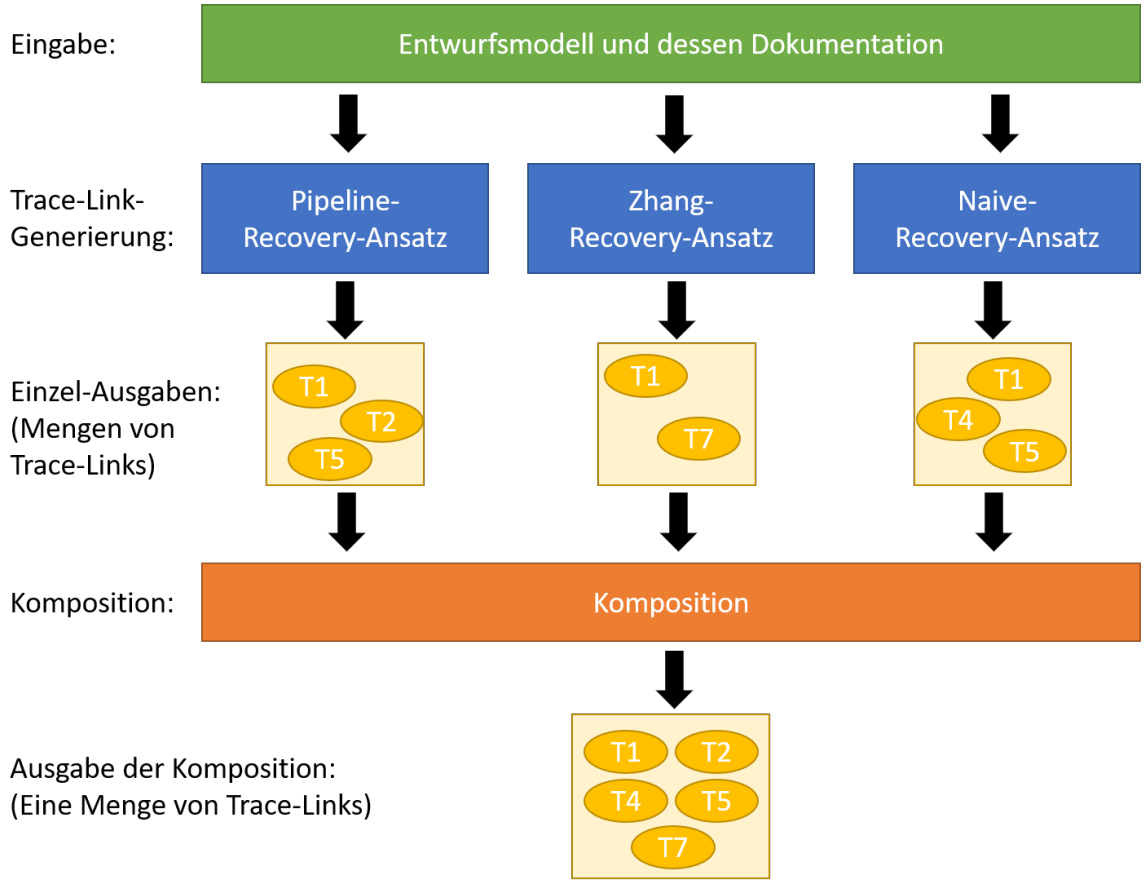


Abbildung 4.1.: Beispiel einer Mengenvereinigungskomposition

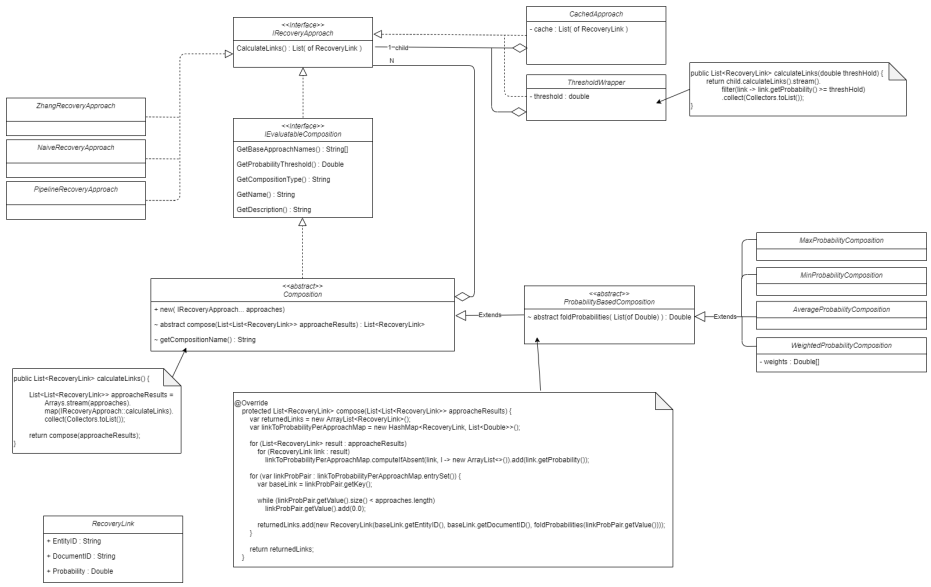


Abbildung 4.2.: Recovery-Klassendiagramm

5. Atomare Recovery-Ansätze

Als atomare Recovery-Ansätze werde ich Recovery-Ansätze bezeichnen, die Trace-Links aus der Eingabe eines Software-Projekts gewinnen. Diese stehen im Kontrast zu Kompositionen, welche nur vorhandene Mengen von Trace-Links zusammenführen. In dieser Ausarbeitung werde ich drei unterschiedliche atomare Ansätze verwenden. Zum Einen gibt es den Pipeline-Ansatz, auf den ich im anschließenden Kapitel 5.1 eingehen werde. Außerdem verwende ich einen auf der Arbeit von Zhang et al. [18] basierenden Ansatz, welcher in Kapitel 5.2 vorgestellt wird und als letztes einen naiven Ansatz, welcher in Kapitel 5.3 näher erläutert wird.

5.1. Pipeline-Ansatz

Der Pipeline-Ansatz ist Teil des SWATTR-Frameworks [1], welches von Jan Keim et al. in ihrer Arbeit „Trace Link Recovery for Software Architecture Documentation“ vorgestellt wird [11]. Der eingebaute Trace-Link-Recovery-Ansatz besteht aus mehreren Schritten. Diese sind Text-Extraktion, Modell-Extraktion, Element-Identifikation und Element-Verbindung (siehe Abbildung 5.1).

In der Textverarbeitung werden verschiedene NLP-Techniken angewendet, wie das Kürzen von Worten auf den Wortstamm. Dabei wird gleichzeitig versucht herauszufinden, welche Elemente es gibt und welchen Typ diese haben.

Die Modell-Extraktion lädt das Modell und extrahiert aus diesem die einzelnen Elemente mit Id, Typ und Name.

Die Element-Identifikation verbindet die Daten aus Text- und Modell-Extraktion.

Diese Daten können letztendlich verwendet werden um Trace-Links zu generieren. Dabei werden die Elemente, die aus dem Text extrahiert wurden, mit denen aus dem Modell verglichen. Ein Trace-Link wird dann erstellt, wenn die Ähnlichkeit zwischen Text und Modell anhand verschiedener Heuristiken groß genug ist.

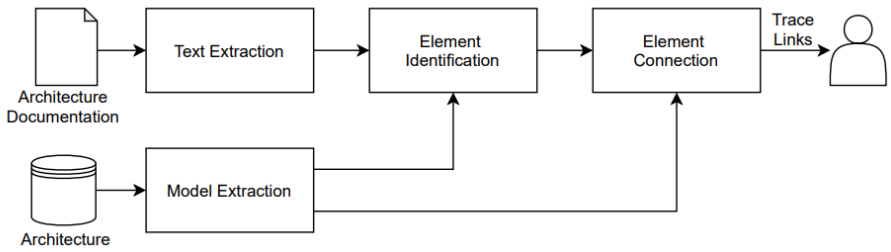


Abbildung 5.1.: Arbeitsweise des Pipeline-Ansatzes (Bild aus [11])

5.2. Zhang-Ansatz

Der Zhang-Ansatz basiert auf einer Arbeit von Zhang et al. aus dem Jahr 2016 [18]. Um Trace-Links zu generieren, werden zunächst alle Sätze aus der Dokumentation und alle Entitäten des Modells per Natural Language Processing analysiert und verarbeitet. Dabei entstehen zu jedem Satz und jeder Entität ein sogenanntes „Bag of words“, welches einzelne Wörter sind, die das Element beschreiben. Außerdem werden Verb-Objekt-Phrases generiert, die aus Prädikat und Objekt eines Satzes bestehen. Bei Entitäten werden diese aus Identifizierern extrahiert, die beispielsweise aus dem Komponentennamen oder Methodensignaturen herausgelesen werden.

Daraufhin werden diese Bag of Words und Verb-Objekt-Phrases in Vektoren eines Vektor-Space-Models überführt. Um herauszufinden, ob ein Trace-Link zwischen einem Satz in der Dokumentation und einem Element des PCMs besteht, wird die Kosinus-Ähnlichkeit der dementsprechenden Vektoren berechnet. Diese Kosinus-Ähnlichkeit definiert die Wahrscheinlichkeit des Trace-Links.

Der größte Unterschied zum ursprünglichen Ansatz von Zhang et al. ist, dass der in dieser Arbeit verwendete Entwurfsmodelle mit deren Dokumentation verknüpft. Der ursprüngliche Ansatz verknüpft hingegen Anforderungen mit Code.

Der Ansatz wurde im Rahmen eines Praktikums von Janek Speit implementiert und verwendet auch Palladio-Component-Modells [17].

5.3. Naiver Ansatz

Der Naive Ansatz ist simpel gehalten und basiert auf einem Textvergleich. Wie beim Zhang-Ansatz werden die Identifizierer (Komponentennamen, Methodensignaturen u.ä.) jeder Entität des Entwurfsmodells aus dem PCM ausgelesen. Jeder Satz in der Dokumentation wird in einzelne Wörter zerlegt. Sowohl die Identifizierer als auch die Wörter der Sätze werden ein Kleinbuchstaben umgewandelt und Satzzeichen werden entfernt. Dann folgt die Zuordnung: Für jede Entität wird überprüft, wie viele der Identifizierer in einem Satz der Dokumentation vorkommen. In Abbildung 5.2 wird dies anhand eines Beispiels gezeigt. Der prozentuale Anteil der Identifizierer in einem Satz ergibt die Wahrscheinlichkeit, dass es sich bei der Kombination dieses Satzes mit dieser Entität um einen Trace-Link handelt. Bei diesem Beispiel wären das zwei von vier zugeordnete Identifizierer und somit eine Wahrscheinlichkeit von 50%. In Pseudocode ausgedrückt werden die Trace-Links wie in den Code-Beispielen 1 und 2 gezeigt berechnet. In Zeile 7 der CalculateTraceLinks-Methode (Algorithmus 1) sieht man, dass nur Sentence-Entity-Paare zur Menge der Trace-Links hinzugefügt werden, die eine Wahrscheinlichkeit von über 0 haben. Andere Filterungen gibt es hier jedoch noch nicht.

Algorithm 1 Calculating Trace-Links

```
1:  function CalculateTraceLinks(DocumentationDocument, PCMModel)
2:      TraceLinks ← {}
3:
4:      for Sentence : DocumentationDocument.GetSentences() do
5:          for Entity : PCMModel.GetEntity() do
6:              p ← calcsimularity(Sentence, Entity)
7:              if p > 0 then
8:                  TraceLinks.Add(new TraceLink(Sentence,Entity,p))
9:              end if
10:         end for
11:     end for
12:
13:     return TraceLinks
14: end function
```

Algorithm 2 Calculating Simularity

```
1:  function CALCSIMULARITY(Sentence, Entity)
2:      FoundIdentifiers ← 0
3:
4:      for Identifier : Entity.GetIdentifiers() do
5:          if Sentence.ContainsWord( Identifier ) then
6:              FoundIdentifiers++
7:          end if
8:      end for
9:
10:     return FoundIdentifiers / Entity.GetIdentifiers().Count()
11: end function
```

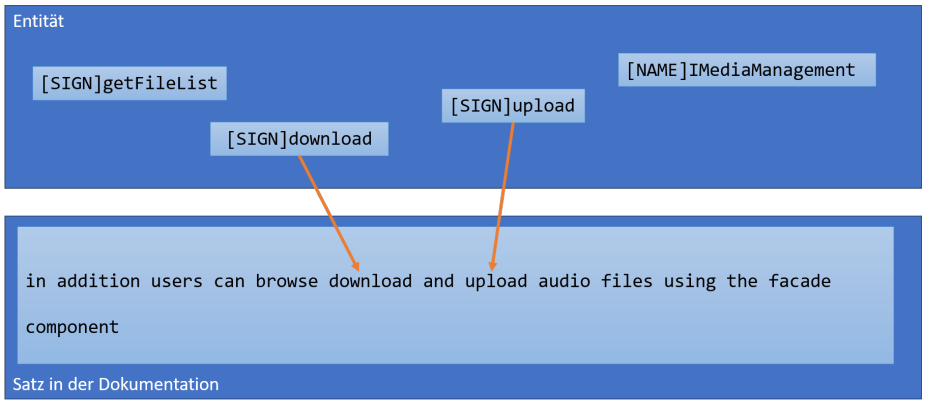


Abbildung 5.2.: Naive Ähnlichkeitsberechnung einer Entität und eines Satzes aus der Dokumentation

6. Kompositionen

Die Kompositionen bilden das Kernstück dieser Arbeit. Sie kombinieren unterschiedliche Ansätze, um neue Ergebnisse zu erhalten. Das Ziel war, sie zudem sehr generisch zu halten, um den Anwendungsbereich nicht einzuschränken. Die minimale Anforderung war somit, dass die von der Komposition verwendeten Ansätze eine Menge von Trace-Links zurück geben. Aus rein funktionaler Sicht war das völlig ausreichend. Zum Zweck der leichteren Evaluation habe ich jedoch zusätzlich das Interface *IEvaluatableApproach* definiert, welches weitere Daten über eine Komposition oder einen Ansatz preisgeben kann und in Abbildung 4.2 zu sehen ist. Diese weiteren Informationen werden jedoch nicht für die Trace-Link-Generierung verwendet, sondern ausschließlich, um die Auswertung der Ergebnisse zu vereinfachen.

Zuerst habe ich zwei spezielle Klassen von Kompositionen betrachtet und einige Vertreter implementiert. Zum Einen mengenbasierte Kompositionen, bei denen ich die typischen Mengenoperationen *Mengenvereinigung* und *Schnittmenge* als Kompositionen implementiert habe. Zum Anderen wahrscheinlichkeitsbasierte Kompositionen, bei denen die Wahrscheinlichkeiten der einzelnen Trace-Links verwendet werden, um beispielsweise die durchschnittliche Wahrscheinlichkeit eines Trace-Links über die atomaren Ansätze hinweg zu berechnen. Im folgenden werde ich in den Abschnitten 6.1 und 6.2 die beiden Klassen und die jeweiligen Vertreter vorstellen. Danach werde ich in Abschnitt 6.3 auf komplexere Kompositionen eingehen, die andere Kompositionen als Eingabe verwenden.

6.1. Mengenbasierte Kompositionen

Eine sehr natürliche und einfache Art, die Mengen von Trace-Links die die einzelnen Ansätze zurückgeben zu verknüpfen, sind Mengenoperationen wie die Mengenvereinigung oder die Schnittmenge. In Abbildung 6.1 ist ein Venn-Diagramm dargestellt, das beispielhaft die Trace-Link-Mengen darstellt. Die Mengenvereinigung umfasst alle Trace-Links, während die Schnittmenge nur die Links beinhaltet, die in allen drei Kreisen liegen.

Es wichtig, anzumerken, dass im Falle meiner Implementierung von Trace-Links zunächst keine sinnvolle Mengenvereinigung im mathematischen Sinne möglich ist. Da ein Trace-Link aus Dokumenten-Id, Modellelement-Id und einer Wahrscheinlichkeit besteht, ist dies nur möglich, wenn die Gleichheit von Trace-Links nicht auf die Gleichheit aller Komponenten zurückgeführt wird. Beispielsweise generieren *Ansätze A und B* Trace-Links der Form (Dokumenten-Id, Modellelement-Id, Wahrscheinlichkeit) mit den Werten (31, 42, 0.5) und (31, 42, 0.7). Dann handelt es sich bei diesen Werten um den gleichen Trace-Link, auch wenn die Wahrscheinlichkeit sich unterscheidet. Aus diesem Grund wird die Wahrscheinlichkeit bei Gleichheitsüberprüfungen nicht beachtet. Jedoch muss definiert werden, welche Wahrscheinlichkeit nun im Endergebnis einer Mengenoperation auftritt.

Abbildung 6.1.: Venn-Diagramm der Trace-Links nach Recovery-Ansatz

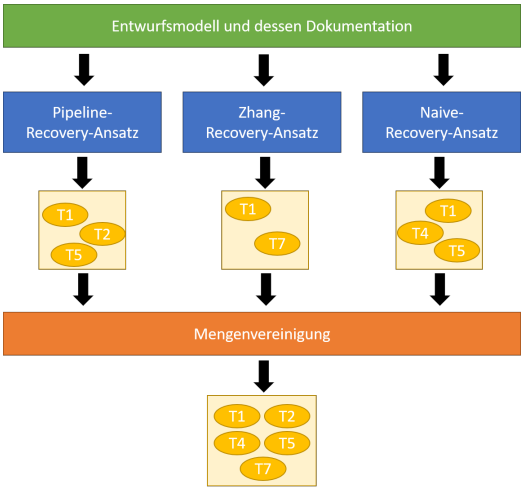
Hierbei habe ich mich für die maximale Wahrscheinlichkeit entschieden. Es wäre jedoch auch möglich, den Durchschnitt oder den kleinsten Wert zu verwenden.

Im Falle einer Mengenvereinigung werden alle Trace-Links, die in mindestens einer Eingabemenge vorkamen (wie in Abbildung 6.2a zu sehen) auch im Ergebnis erscheinen. Diese Komposition liefert somit immer mindestens so viele Trace-Links, wie ein einzelner Ansatz aus dem die Komposition besteht. Dadurch gibt die Komposition meist mehr Trace-Links zurück als die einzelnen Ansätze. Jedoch können sich auf diese Weise fehlerhafte Links „einschleichen“. Eine Beispielrechnung einer Mengenvereinigung ist im Code-Beispiel 3 dargestellt.

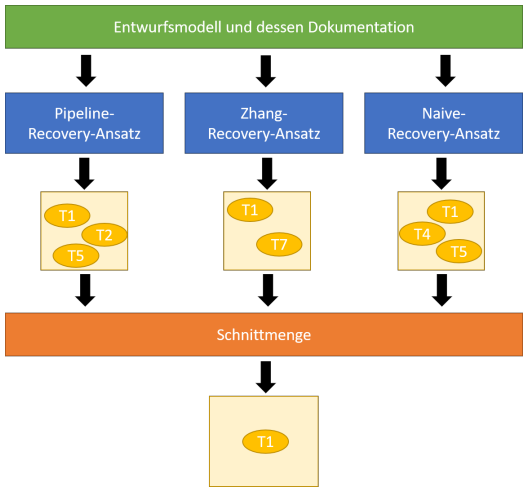
Im Gegensatz dazu steht die Schnittmenge. Nur wenn ein Trace-Link von allen Ansätzen erkannt wird, wird dieser in die Ergebnismenge aufgenommen. Dadurch werden im Allgemeinen viel weniger Trace-Links gefunden. Dies ist in Abbildung 6.2b zu sehen, wo nur ein einziger Trace-Link noch übrig bleibt. Da jedoch alle Ansätze übereinstimmen, steigt die Wahrscheinlichkeit, dass es sich um einen korrekten Trace-Link handelt. Eine Beispielrechnung einer Schnittmengenkomposition ist im Code-Beispiel 4 dargestellt.

Algorithm 3 Calculating Trace-Links with Union-Composition

```
1: ApproachA ← new ApprochFromInputData({(1,a, 0.5), (3,f, 0.2), (5,b, 0.8)})
2: ApproachB ← new ApprochFromInputData({(1,a, 0.9), (4,f, 0.4)})
3:
4: UnionComposition ← new UnionComposition(ApproachA, ApproachB)
5: UnionSet ← UnionComposition.CalculateLinks()
6:
7: Assert( UnionSet = {(1, a, 0.9), (3, f, 0.2), (4, f, 0.4), (5, b, 0.8)} )
```



(a) Komposition basierend auf der Mengenvereinigung



(b) Komposition basierend auf der Schnittmenge

Abbildung 6.2.: Mengenbasierte Kompositionen

Algorithm 4 Calculating Trace-Links with Intersect-Composition

1: $ApproachA \leftarrow new\ ApprochFromInputData(\{(1,a,0.5), (3,f,0.2), (5,b,0.8)\})$
2: $ApproachB \leftarrow new\ ApprochFromInputData(\{(1,a,0.9), (4,f,0.4)\})$
3:
4: $IntersectComposition \leftarrow new\ IntersectComposition(ApproachA, ApproachB)$
5: $IntersectSet \leftarrow IntersectComposition.CalculateLinks()$
6:
7: Assert(IntersectSet = {(1, a, 0.9)})

6.2. Wahrscheinlichkeitsbasierte Kompositionen

Die von mir verwendeten wahrscheinlichkeitsbasierten Kompositionen arbeiten alle auf die gleiche Weise. Zuerst werden alle Kombinationen von Dokumenten-Id und Modellelement-Id gesucht, welche die einzelnen Ansätze ausgeben. Dann werden pro Tupel von Dokumenten-Id und Modellelement-Id die Wahrscheinlichkeiten gesucht, die die jeweiligen Ansätze dieser Kombination zuordnen. Sollte ein Ansatz diese Kombination nicht gefunden haben, wird die Wahrscheinlichkeit implizit als Null angenommen. Anschließend werden die verschiedenen Werte der Wahrscheinlichkeit pro Tupel zu einem einzigen zusammengefasst. Auf welche Weise ist je nach konkreter Komposition unterschiedlich. Diese Zuordnung ist in Abbildung 6.3 zu sehen. Die implizit auf Null gesetzten Werte sind dabei gestrichelt und blasser dargestellt.

Als unterschiedliche wahrscheinlichkeitsbasierte Kompositionen verwende ich die *Minimum-Probability-Composition*, die *Maximum-Probability-Composition* und die *Average-Probability-Composition*, die jeweils die Fold-Operation auf ihre Weise implementieren.

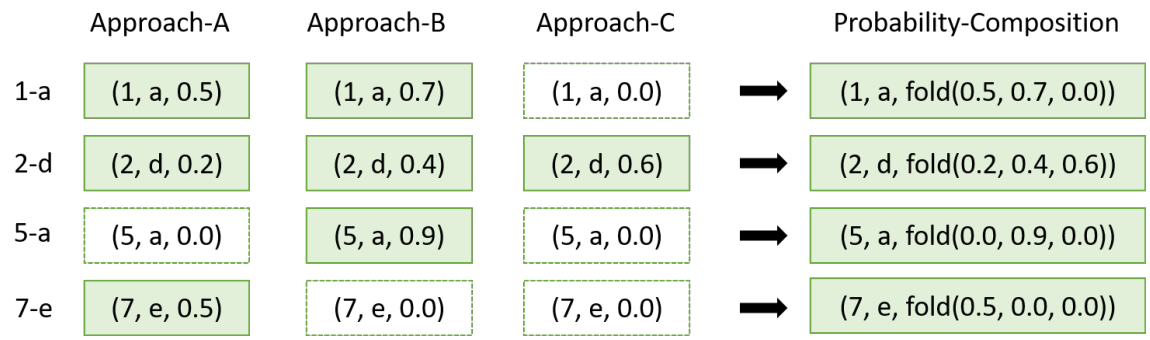


Abbildung 6.3.: Funktionsweise einer Wahrscheinlichkeitskomposition

Hierbei ist anzumerken, dass die Kompositionen zuerst einmal die exakt selben Ergebnisse liefern wie die Mengenvereinigung. Nur die Wahrscheinlichkeiten unterscheiden sich. Aus diesem Grund ist eine wahrscheinlichkeitsbasierte Kompositionen meist nur in Kombination mit einem Wahrscheinlichkeitsfilter sinnvoll. Dieser filtert alle Trace-Links aus einer Ergebnismenge, deren Wahrscheinlichkeit einen bestimmten Grenzwert unterschreiten. Im Programm-Code habe ich einen solchen Filter mit Hilfe des Decorator-Patterns implementiert. Auf diese Weise können jegliche Recovery-Ansätze um einen solchen Filter erweitert werden. Im Code-Beispiel 5 ist eine Maximum-Probability-Composition mit einem Filter gezeigt.

Algorithm 5 Calculating Trace-Links with Maximum-Probability-Composition and Probability-Filter

```
1: ApproachA ← new ApprochFromInputData({(1,a, 0.5), (3,f, 0.2), (5,b, 0.8)})
2: ApproachB ← new ApprochFromInputData({(1,a, 0.9), (4,f, 0.4)})
3:
4: MaxProbComp ← new MaximumProbabilityComposition(ApproachA, ApproachB)
5: FilteredMaxProbComp ← newFilter(MaxProbComp, 0.55)
6: FinalSet ← FilteredMaxProbComp.CalculateLinks()
7:
8: Assert( FinalSet = {(1, a, 0.9), (5, b, 0.8)} )
```

6.3. Komposition von Kompositionen

Da nicht nur atomare Ansätze als Ergebnis eine Menge von Trace-Links zurück liefern, sondern auch Kompositionen, können diese wiederum als Eingabe für andere Kompositionen verwendet werden. Diese Tatsache ermöglicht ein „Plug and Play“-artiges Zusammenbauen von Kompositionen. Beispielsweise die *(High-Average-Probability)-Union-(High-Precision)-Composition* wird aus einer *High-Average-Probability-Composition* und einer *High-Precision-Composition* zusammengesetzt, die durch eine Mengenvereinigung zusammengeführt werden. Eine schematische Darstellung ist in Abbildung 6.4 zu sehen. Die High-Average-Probability-Composition ist dabei eine normale Average-Probability-Composition, die mit

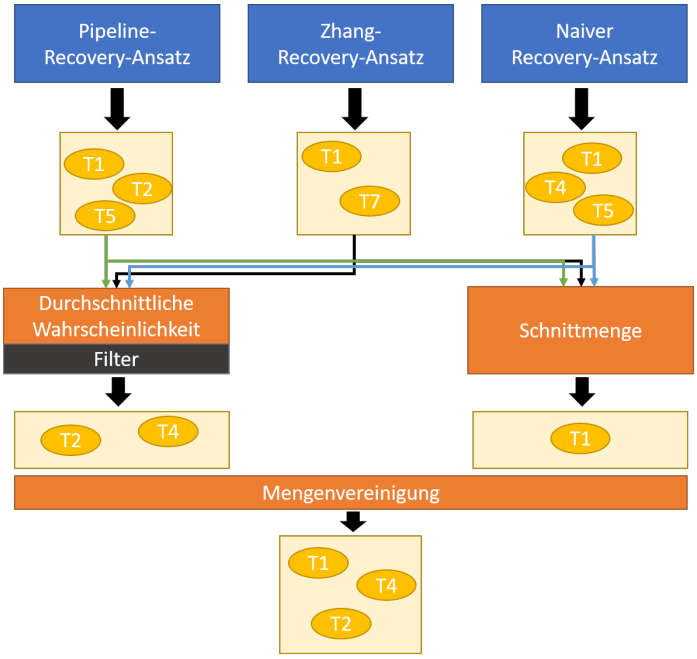


Abbildung 6.4.: Die Komposition mehrerer Kompositionen

einem Filter umgeben ist. Dieser Teil soll Trace-Links zurückgeben, bei denen sich die atomaren Ansätze verhältnismäßig sicher sind. Die High-Precision-Composition hingegen ist eine einfache Schnittmengenkomposition. Sie soll auch Trace-Links erhalten, die zwar eine geringe Wahrscheinlichkeit bei den einzelnen atomaren Ansätzen haben, jedoch von allen atomaren Ansätzen gefunden werden. Diese beiden Ergebnismengen werden dann durch eine Mengenvereinigungskomposition vereinigt.

7. Evaluation

Um herauszufinden, wie sich die Güte durch die Komposition verschiedener Ansätze verändert, werde ich einige Kompositionen implementieren und mit verschiedenen Kombinationen der atomaren Ansätze auf drei Software-Projekten ausführen.

In diesem Kapitel werde ich zuerst erklären, auf welche weise ich die Güte eines Recovery-Ansatzes angeben möchte (Abschnitt 7.1). Daraufhin werde ich die Ergebnisse der Auswertung vorstellen (Abschnitt 7.2).

7.1. Methodik

Die Güte eines Recovery-Ansatzes werde ich durch die Metriken Präzision, Ausbeute, F1 und F2 ausdrücken. Diese ergeben sich alle aus der Ergebnismenge des Ansatzes im Vergleich zu einem manuell erarbeiteten Goldstandard.

Trace-Links, die von dem Ansatz richtig erkannt werden, sind *True Positives* (TP). Ein Trace-Link wurde richtig erkannt, wenn dieser einen Satz in der Dokumentation auf das richtige Element im Modell abbildet. *False Positives* (FP) dagegen sind Links, die der Ansatz erkennt, die aber gar keine sind. Das ist der Fall, wenn ein vom Ansatz vorgeschlagener Link (also das Tupel von Dokumentations-Satznummer und Modellelement) nicht im Goldstandard vorkommt. Außerdem gibt es *False Negatives* (FN). Das sind die Links, die vom Ansatz nicht erkannt wurden, aber laut Goldstandard erkannt werden sollten.

Der Goldstandard, gegen den die Ansätze geprüft werden können, wird gebildet, indem manuell alle Trace-Links eines Software-Projektes ausgearbeitet werden. Durch diese Referenz-Links können die von den Ansätzen zurückgegebenen Links in die oben aufgeführten Kategorien eingeteilt werden.

Anschließend können auf dieser Grundlage die vier Werte berechnet werden, die die Güte eines Ansatzes beschreiben sollen.

$Ausbeute := \frac{TP}{TP+FN}$,

$Präzision := \frac{TP}{TP+FP}$,

$F1 := \frac{2*Ausbeute*Präzision}{Ausbeute+Präzision}$ und

$F2 := \frac{5*Ausbeute*Präzision}{Ausbeute+4*Präzision}$

Die Ausbeute beschreibt den Anteil an Links, die korrekterweise gefunden wurden, in Bezug zu denen, die gefunden werden sollen. Im Gegensatz dazu gibt die Präzision an, wie viele gefundene Trace-Links korrekt sind. F1 und F2 versuchen, beide Größen balanciert

zu vereinigen. Dazu wird das harmonische Mittel verwendet. Der F2-Wert gewichtet dabei die Ausbeute doppelt, da diese oft etwas wichtiger ist als die Präzision. Dies lässt sich aus der Forschungsarbeit von Hayes et al. ableiten. Sie beschäftigt sich damit, welche Werte bei der Trace-Link-Generierung als am wichtigsten eingestuft werden sollten [9].

F1-Wert und F2-Wert sind beide Ausprägungen des F_β-Scores. Ein F_β-Wert berechnet sich im Allgemeinen aus $(1 + \beta^2) * \frac{Ausbeute * Präzision}{Ausbeute + \beta^2 * Präzision}$ mit β als positiver reeller Wert. Bei einem F_β-Wert wir die Ausbeute als β mal so wichtig betrachtet wie die Präzision.

7.2. Ergebnisse der Auswertungen

In diesem Kapitel werde ich zuerst das Ergebnis der einzelnen atomaren Ansätze vorstellen, um diese als Basiswerte verwenden zu können. Danach können die Kompositionen darauf überprüft werden, ob sie eine Verschlechterung oder eine Verbesserung herbeigeführt haben.

7.2.1. Atomare Ansätze im Vergleich

Um einen ersten Eindruck von den atomaren Ansätzen zu erhalten, werde ich in diesem Abschnitt zuerst deren Recovery-Qualität über alle Fallstudien-Projekte hinweg darstellen. Anschließend werde ich noch auf die Ergebnisse bei einzelnen Projekten eingehen und wie sie sich voneinander unterscheiden.

In Abbildung 7.1 ist zu erkennen, dass der Pipeline-Ansatz im Vergleich zum naiven Ansatz und dem Zhang-Ansatz sowohl den besten F1-Wert mit 0,64 als auch F2-Wert mit 0,73 hat. Dieser Wert ist ein Durchschnitt über alle drei Software-Projekte hinweg. Als nächstbester Ansatz kristallisiert sich der naive Ansatz heraus, der trotz seiner Einfachheit mit einem F1-Wert von 0,56 und einem F2-Wert von 0,62 gute Ergebnisse erzielt. Der Zhang-Ansatz schneidet hier am schlechtesten ab. Sowohl der F1-Wert mit 0,32 als auch der F2-Wert mit 0,37 sind um Einiges niedriger als die Werte der anderen Ansätze.

Der Durchschnitt über alle Ansätze, aufgeteilt nach Software-Projekten (wie er in Abbildung 7.2 dargestellt ist), zeigt, dass Teammates und TeaStore sehr ähnliche Werte erzielt haben. Sowohl der F1-Wert als auch der F2-Wert von Teammates liegen knapp unter 0,6. Teammates ist das größte der drei Projekte und hat auch mit 80 Links die meisten Trace-Links. Teastore hat einen etwas schlechteren F1-Wert von 0,53. Dafür ist der F2-Wert mit 0,61 höher. Dies kommt zu Stande weil sich die Ausbeute im Vergleich zu Teammates um 0,07 erhöht hat, während die Präzision um 0,2 gesunken ist. Der Goldstandard von Teastore beinhaltetet wie auch der von Mediastore 25 Trace-Links. Mediastore scheint das Projekt zu sein, aus welchem man am schwersten Trace-Links extrahieren kann. Mit einem F1-Wert von 0,41 und einem F2-Wert von 0,52 schneidet es merkbar schlechter ab als die anderen beiden Projekte. Anzumerken ist hier auch die hohe Diskrepanz zwischen Ausbeute und Präzision. Die Ausbeute ist viel höher als die Präzision. Das lässt darauf schließen, dass die echten Trace-Links einfach zu finden sind, es jedoch auch viele False-Positives gibt.

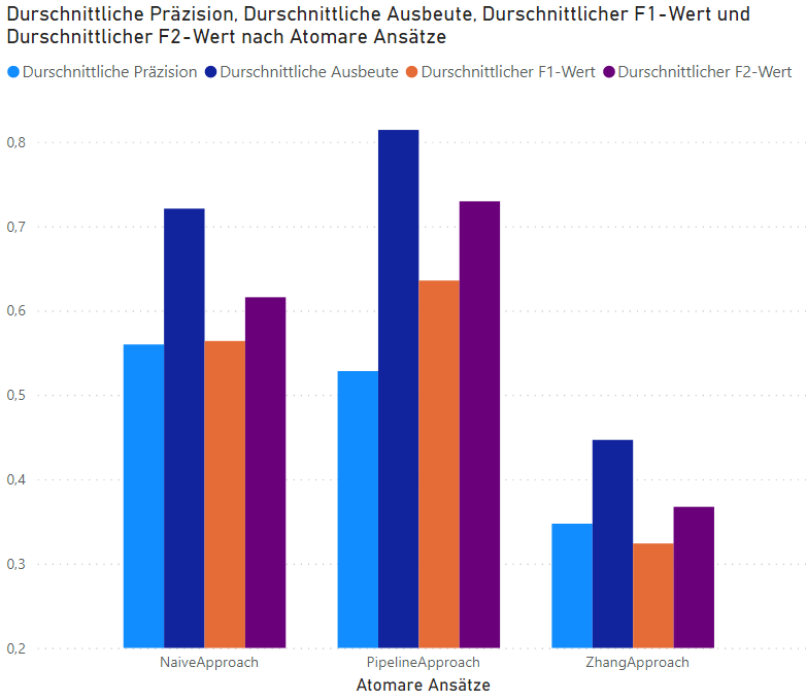


Abbildung 7.1.: Auswertung der atomaren Ansätze nach Ansatz

Die Aussage, dass der Zhang-Ansatz die schlechtesten Ergebnisse erzielt, ist in Abbildung 7.5 noch besser zu sehen. In jedem Projekt liegen dessen Werte unter dem Durchschnitt. Der naive Ansatz hat hingegen im Teammates- und TeaStore-Projekt eine um 0,36 und 0,02 höhere Präzision im Vergleich zum Pipeline-Ansatz (Abbildungen 7.3 und 7.4). Da der naive Ansatz auf einen einfachen Textvergleich setzt, scheinen exakt übereinstimmende Worte in der Identifizierer-Liste und der Dokumentation ein starkes Indiz für einen Trace-Link zu sein. Jedoch werden auch einige Trace-Links nicht erkannt. Bei Teammates beispielsweise liegt die Ausbeute nur bei 0,56, während der Pipeline-Ansatz einen Spitzenwert von 0,96 erreicht.

Die gerade vorgestellten Werte können später verwendet werden, um einen Vergleich von Kompositionen zu ihren Bestandteilen (atomaren Ansätzen) zu ziehen. Außerdem können sie einen Anhaltspunkt geben, warum sich Kompositionen auf eine bestimmte Weise verhalten haben. Im nächsten Abschnitt werden zunächst die Ergebnisse der Kompositionen präsentiert, bevor ein Vergleich zu deren Bestandteilen durchgeführt wird.

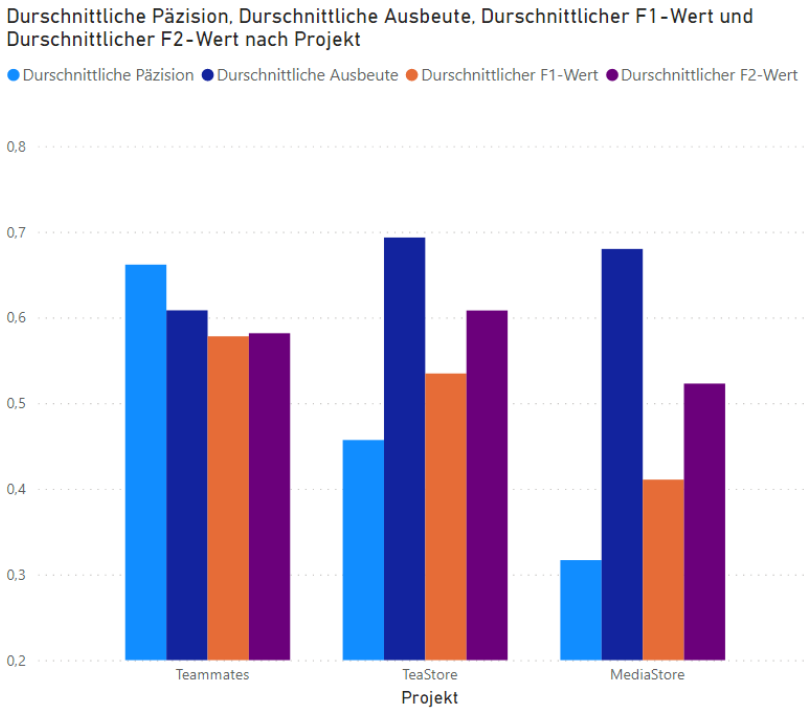


Abbildung 7.2.: Auswertung der atomaren Ansätze nach Projekt

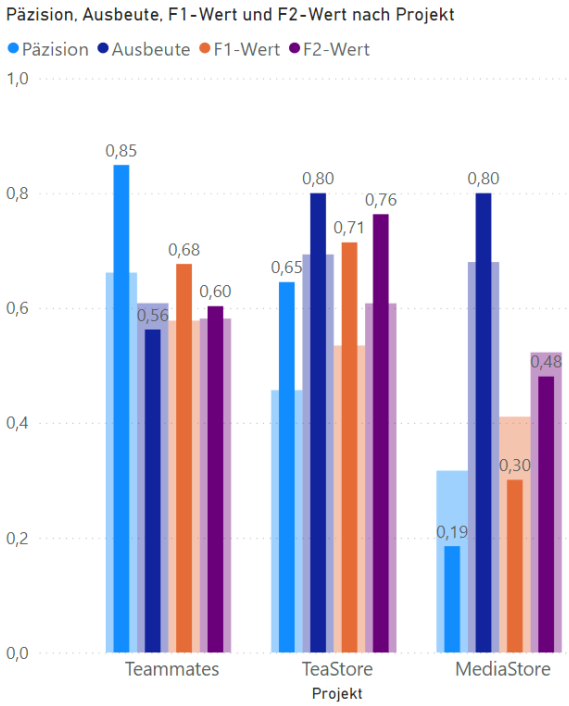


Abbildung 7.3.: Naiver Ansatz nach Projekt im Vergleich zum Durchschnitt über alle Ansätze, wie im Hintergrund dargestellt.

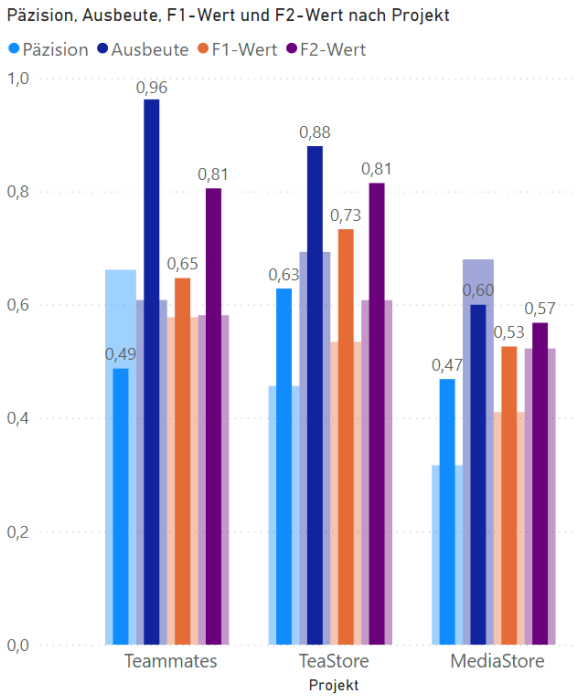


Abbildung 7.4.: Pipeline-Ansatz nach Projekt im Vergleich zum Durchschnitt über alle Ansätze, wie im Hintergrund dargestellt.

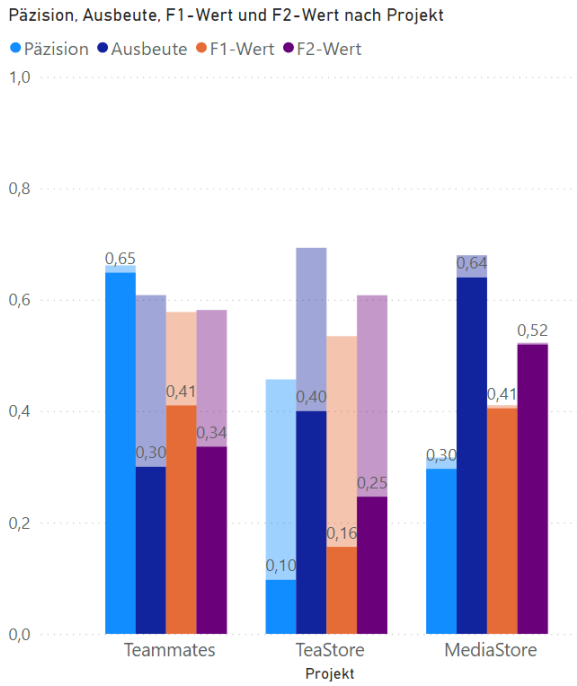


Abbildung 7.5.: Zhang-Ansatz nach Projekt im Vergleich zum Durchschnitt über alle Ansätze, wie im Hintergrund dargestellt.

7.2.2. Kompositionen im Vergleich

Nachdem ich die Ergebnisse der einzelnen atomaren Ansätze vorgestellt habe, will ich nun darauf eingehen, welche Werte die unterschiedlichen Kompositionen erreicht haben. Dazu werde ich zunächst den Begriff der Kompositionskonfiguration einführen. Anschließend werde ich zuerst die Ergebnisse der Kompositionstypen - aufgeteilt nach deren atomaren Bestandteilen - und anschließend die besten zwanzig Kompositionskonfiguration aufzeigen.

Um über einen Vergleich von Kompositionen reden zu können, muss klar sein, was eine Kompositionskonfiguration ist. In den letzten Kapiteln habe ich nur über Kompositionsarten bzw. Kompositionstypen gesprochen. Beispiele hierfür sind die Mengenvereinigungskomposition oder die Maximum-Probability-Komposition. Diese jeweiligen Typen können jedoch mit unterschiedlichen Konfigurationen ausgeführt werden. Ich werde zwischen zwei Konfigurationseigenschaften unterscheiden. Zum Einen ist es sehr wichtig zu wissen, welche atomaren Ansätze der Komposition zu Grunde liegen. Es macht einen deutlichen Unterschied, ob beispielsweise die Mengenvereinigungskomposition mit dem Zhang-Ansatz und dem naiven Ansatz ausführt wird, oder doch mit dem Pipeline-Ansatz zusammen mit dem naiven Ansatz. Die zweite Konfigurationseigenschaft ist der Wert des Wahrscheinlichkeitsfilters. Zum Anderen ist der Filter ein essentieller Teil von wahrscheinlichkeitsbasierten Kompositionen und dessen Wert hat einen großen Einfluss darauf, wie die Ergebnismenge aussieht. Kompositionen ohne Filter können einfach mit einem Filter des Wertes Null dargestellt werden. Ich werde im Folgenden Kompositionskonfigurationen als Tripel der Art *(Kompositionstyp, mitwirkende atomare Ansätze, Wahrscheinlichkeitsfilter-Wert)* schreiben. Für Konfigurationen ohne Filter werde ich den letzten Wert auslassen.

Abbildung 7.6.: Auswertung der Kompositionen nach atomaren Bestandteilen

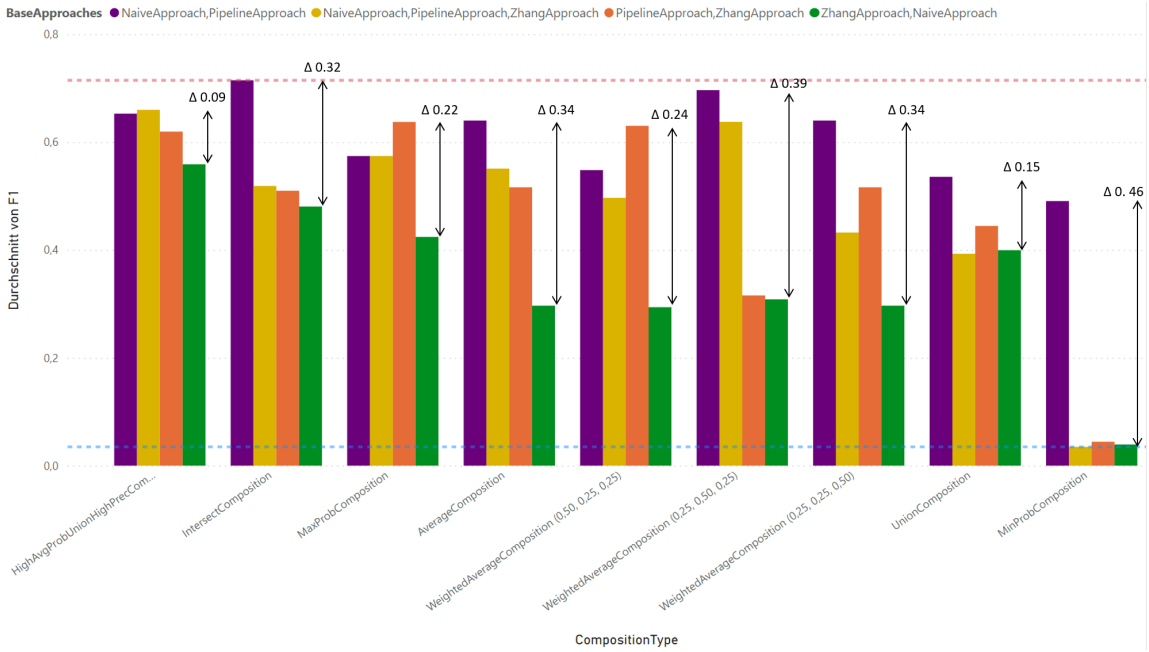


Abbildung 7.7.: Vergleich der F1 Wert unterschiedlicher Kompositionen nach atomaren Bestandteilen. Die rote Linie und die Blaue ist der Minimalwert und der Maximalwert über alle Ergebnisse hinweg.

CompositionType	BaseApproaches	Probability Threshold	Durchschnitt von Precision	Durchschnitt von Recall	Durchschnitt von F1	Durchschnitt von F2
AverageComposition	NaiveApproach,PipelineApproach	0,50	0,69	0,79	0,72	0,75
AverageComposition	NaiveApproach,PipelineApproach,ZhangApproach	0,30	0,69	0,79	0,72	0,75
HighAvgProbUnionHighPrecComposition	NaiveApproach,PipelineApproach	0,50	0,69	0,79	0,72	0,75
HighAvgProbUnionHighPrecComposition	NaiveApproach,PipelineApproach,ZhangApproach	0,30	0,69	0,79	0,72	0,75
WeightedAverageComposition (0,25, 0,25, 0,50)	NaiveApproach,PipelineApproach	0,50	0,69	0,79	0,72	0,75
WeightedAverageComposition (0,25, 0,50, 0,25)	NaiveApproach,PipelineApproach	0,30	0,69	0,79	0,72	0,75
MaxProbComposition	PipelineApproach,ZhangApproach	0,20	0,53	0,83	0,64	0,74
WeightedAverageComposition (0,25, 0,50, 0,25)	NaiveApproach,PipelineApproach,ZhangApproach	0,20	0,53	0,83	0,64	0,74
AverageComposition	NaiveApproach,PipelineApproach,ZhangApproach	0,20	0,51	0,83	0,63	0,73
HighAvgProbUnionHighPrecComposition	NaiveApproach,PipelineApproach,ZhangApproach	0,20	0,51	0,83	0,63	0,73
AtomicApproach	PipelineApproach	0,00	0,53	0,81	0,64	0,73
AverageComposition	NaiveApproach,PipelineApproach	0,40	0,53	0,81	0,64	0,73
AverageComposition	PipelineApproach,ZhangApproach	0,20	0,53	0,81	0,64	0,73
AverageComposition	PipelineApproach,ZhangApproach	0,30	0,53	0,81	0,64	0,73
AverageComposition	PipelineApproach,ZhangApproach	0,40	0,53	0,81	0,64	0,73
HighAvgProbUnionHighPrecComposition	NaiveApproach,PipelineApproach	0,40	0,53	0,81	0,64	0,73
HighAvgProbUnionHighPrecComposition	PipelineApproach,ZhangApproach	0,20	0,53	0,81	0,64	0,73
HighAvgProbUnionHighPrecComposition	PipelineApproach,ZhangApproach	0,30	0,53	0,81	0,64	0,73
HighAvgProbUnionHighPrecComposition	PipelineApproach,ZhangApproach	0,40	0,53	0,81	0,64	0,73
MaxProbComposition	NaiveApproach,PipelineApproach	0,80	0,53	0,81	0,64	0,73

Abbildung 7.8.: Werte der besten 20 Kompositionskonfigurationen. (Die restlichen Daten stehen in Tabelle A.1 im Anhang)

Zunächst möchte ich auf Abbildung 7.6 eingehen. Dort sind verschiedene Kompositionstypen nach deren atomaren Bestandteilen aufgezeigt. Die Werte werden mit dem Durchschnitt über alle Wahrscheinlichkeitsfilter-Werte berechnet. Ich habe Filter mit den Werten von 0,2 bis 0,6 in 0,1-er-Schritten gewählt. Das Erste, was bei Abbildung 7.6 auffällt, sind die drei Werteeinbrüche bei der *MinProbability-Composition*. Dieser Einbruch tritt immer auf, wenn der Zhang-Ansatz mitwirkt. Das liegt daran, dass dieser Ansatz oft

nur sehr geringe Werte (< 0,3) für die Wahrscheinlichkeit angibt. Durch die Minimum-Verknüpfung werden somit viele Trace-Links der anderen Ansätze auf diese geringen Werte gezogen. Die meisten Wahrscheinlichkeitsfilter werden diese Trace-Links sofort entfernen.

Hieran ist ein Problem gut zu erkennen, welches die Komposition verschiedener Ansätze mit sich bringt: Verschiedene Ansätze haben unterschiedliche Normen und müssen gegebenenfalls auf eine einheitliche Stufe gebracht werden. Das heißt, sie müssen aufeinander abgestimmt sein.

Außerdem ist erkennbar, dass die meisten Kompositionen, die nur Zhang und den naiven Ansatz als Grundlage verwenden und den Pipeline-Ansatz nicht beinhalten, meist schlechtere Ergebnisse generieren. Das ist jedoch nicht weiter verwunderlich, da der Pipeline-Ansatz wie in Abschnitt 7.2.1 aufgezeigt, der beste der drei Ansätze ist. Da eine Komposition keine neuen Trace-Links erzeugen kann, die die einzelnen Ansätze nicht gefunden haben, kann die Komposition nicht besser als die Summe der Einzelansätze sein. Dieser Fall tritt jedoch nur ein, wenn diese vollständig orthogonale Ergebnisse liefern, was in der Praxis eher nicht auftritt.

Betrachtet man nun weiter die Werte der einzelnen Kompositionen, ist zu erkennen, dass die im Abschnitt 6.1 zu mengenbasierten Kompositionen aufgestellten Überlegungen zutreffen. Bei der Vereinigungskomposition werden alle Trace-Links vereinigt und somit hat diese Komposition mit Werten zwischen 0,73 und 0,88 die größte Ausbeute. Dafür sinkt die Präzision. Bei der Schnittmengenkomposition ist es genau umgekehrt. Da das Ergebnis nur aus Trace-Links besteht, die alle Ansätze gefunden haben, ist die Ausbeute verhältnismäßig gering. Dafür ist die Präzision zwischen 0,64 und 0,82 jedoch sehr hoch.

Nicht weiter verwunderlich ist außerdem, dass die Komposition, die den gewichteten Durchschnitt der Wahrscheinlichkeiten zusammen mit einem Filter verwendet, einen Werteeinbruch erfährt, wenn das größte Gewicht auf dem Zhang-Ansatz liegt.

Die *MaxProbability-Komposition* erzielt ähnliche Ergebnisverteilungen (hohe Ausbeute geringe Präzision) wie die Mengenvereinigung. Da sie auf die gleiche Weise arbeitet, ist das nicht weiter verwunderlich. Doch die Werte sind trotzdem besser, weil diese Komposition zusätzlich einen Wahrscheinlichkeitsfilter verwendet. Dadurch werden die Trace-Links aussortiert, bei denen sich alle Ansätze unsicher sind. Das ergibt eine erhöhte Präzision und somit auch bessere F1- und F2-Werte.

Die *(High-Average-Probability)-Union-(High-Precision)-Composition* zeichnet sich durch sehr konsistente Ergebnisse aus. Der Unterschied zwischen Ausbeute und Präzision beträgt maximal 0,19. Auch sind die F1- und F2-Werte sehr stabil. In Abbildung 7.7 sind die F1-Werte aufgeteilt nach atomaren Ansätzen je Kompositionen dargestellt. Der Unterschied zwischen dem besten und dem schlechtesten F1-Wert ist bei der *(High-Average-Probability)-Union-(High-Precision)-Composition* nur 0,09, während die meisten anderen Kompositionen größere Schwankungen von bis zu 0,46 aufweisen.

Die besten zwanzig Kompositionskonfigurationen (anhand des F2-Werts) sind in Abbildung 7.8 aufgelistet. Die gesamte Datenliste ist im Anhang in Tabelle A.1 zu finden. Schon die ersten zwei Einträge bergen interessante Informationen. Die ersten zwei Plätze sind jeweils von einer *Average-Probability-Composition* belegt, die sich jedoch in der Zusammensetzung und dem Filterwert unterscheiden. Während der erste Platz nur den Pipeline-

und den naiven Ansatz verwendet, kommt beim zweiten Platz noch der Zhang-Ansatz dazu. Da dieser jedoch sehr geringe Wahrscheinlichkeiten liefert, muss hier der Schwellwert geringer sein, damit nicht zu viele Trace-Links unnötig herausgefiltert werden.

Außerdem ist zu sehen, dass es gleich sechs Konfigurationen mit dem gleichen besten F2-Wert von 0,75 gibt. Dann folgen zwei Konfigurationen mit einem F2-Wert von 0,74. Auf diese folgen 29 Konfigurationen mit einem F2-Wert von 0,73. Daran ist zu sehen, dass es sehr viele Möglichkeiten gibt, die sehr ähnliche Ergebnisse erzielen. Den atomaren Pipeline-Ansatz findet man unter den 29 Konfigurationen mit einem Wert von 0,73. Dessen Ausbeute ist sogar besser als die der ersten sechs Konfigurationen. Diese haben jedoch eine wesentlich höhere Präzision, was den F2-Wert positiv beeinflusst. Somit gibt es 37 Konfigurationen die bessere oder gleich gute Ergebnisse liefern, wie der beste Atomare Ansatz. Jedoch gibt es auch etliche Konfigurationen die miserable F2-Werte von unter 0,1 liefern. Meist sind das Konfigurationen mit sehr hohen Filterwerten von über 0,6 oder Konfigurationen basierend auf der *MinProbability-Composition*.

7.2.3. Kompositionen im Vergleich zu ihren Bestandteilen

Nachdem ich in den letzten Abschnitten die absoluten Ergebnis-Werte der atomaren Ansätze und der Kompositionen dargestellt habe, werde ich nun diese in Relation bringen und untersuchen, ob Kompositionen eine Verbesserung oder eine Verschlechterung herbeigeführt haben.

Um herauszufinden ob die Güte der Trace-Links durch die Komposition zugenommen hat, werde ich als nächstes die Änderung der F1- und F2-Werte der Kompositionen im Vergleich zu deren Bestandteilen betrachten. Der Verbesserungswert bestimmt sich hierbei durch die Differenz des Ergebnisses der Komposition und dem Durchschnitt der Ergebnisse der Basisansätze. Ein positiver Wert zeigt eine Verbesserung durch die Komposition auf, während ein negativer Wert eine Verschlechterung bedeutet. Später werde ich noch darauf eingehen, wie sich die Ergbenisse ändern, wenn der Vergleich nicht mit dem Durchschnitt, sondern mit dem besten Einzelansatz durchgeführt wird.

Beispielsweise würde sich der Verbesserungswert einer Komposition, die alle drei atomaren Ansätze verwendet, mit der Durchschnittsreferenz wie folgt berechnen:

$$Verbesserung\ von\ F1 := F1_{Komposition} - Durchschnitt(F1_{Pipeline},\ F1_{Naive},\ F1_{Zhang})$$
$$Verbesserung\ von\ F2 := F2_{Komposition} - Durchschnitt(F2_{Pipeline},\ F2_{Naive},\ F2_{Zhang})$$

Als erstes werde ich auf die Verbesserung/Verschlechterung durch die einzelnen Kompositionen eingehen. Dazu werde ich wiederum einen Durchschnitt über die Verbesserungswerte dieser Komposition mit unterschiedlichen atomaren Ansätzen berechnen und nach den Filterwerten gruppieren.

In Abbildung 7.9 sind die Kompositionen aufgetragen, die keinen Wahrscheinlichkeitsfilter verwenden. Die Schnittmengenkomposition kann einen um 0,05 verbesserten F1-Wert vorweisen, während der F2-Wert um 0,07 schlechter geworden ist. Da der F2-Wert doppelt

so viel Gewicht auf die Ausbeute legt und die Schnittmengenkomposition eine schlechtere Ausbeute hat, war dies zu erwarten.

Bei der Mengenvereinigung zeichnet sich ein genau umgekehrtes Bild. Hier ist der F1-Wert um 0,06 gesunken und der F2-Wert um 0,02 gestiegen. Sowohl die Verbesserung als auch die Verschlechterung sind hier etwas milder ausgefallen als bei der Schnittmengenkomposition.

Bei den Kompositionen mit Wahscheinlichkeitsfiltern, wie in Abbildungen 7.10, 7.11, 7.12 und 7.13 gezeigt, ist bei der *Minimum-Probability-Composition* immer eine Verschlechterung von zwischen 0,27 und 0,43 beim F1-Wert und zwischen 0,37 und 0,51 beim F2-Wert zu erkennen. Die anderen Kompositionen verzeichnen bis zu einem Filterwert von 0,3 durchgehend eine Verbesserung. Die größte Verbesserung des F1-Werts von 0,14 erzielt die gewichtete Komposition mit dem Gewicht auf dem Pipeline-Ansatz und einem Filterwert von 0,3. Die größte Verbesserung des F2-Wertes von 0,12 erzielt dagegen die *(High-Average-Probability)-Union-(High-Precision)-Composition* ebenfalls beim Filterwert von 0,3.

Beim Filterwert von 0,4 verschlechtern sich nun auch die gewichteten Kompositionen und ab einem Filterwert von 0,5 können nur noch die *(High-AverageProbability)-Union-(High-Precision)-Composition* und die *Maximum-Probability-Composition* Verbesserungen verzeichnen.

Eine alternative Ansicht ist die Betrachtung basierend auf den atomaren Ansätzen, wie sie in Abbildung 7.14 dargestellt ist. Hier ist zu erkennen, dass durchschnittlich nur für die Filterwerte zwischen 0,1 und 0,3 Verbesserungen zu erwarten sind. Bei Kombinationen, in denen der Zhang-Ansatz enthalten ist, ist schon bei kleineren Filterwerten eine Verschlechterung zu erkennen, im Gegensatz zu Kombinationen ohne den Zhang-Ansatz. Somit bestätigt sich auch hier, dass für Kombinationen mit dem Zhang-Ansatz niedrigere Filterwerte nötig sind. Andernfalls müsste die Wahrscheinlichkeitsnormierung des Zhang-Ansatzes angepasst werden.

Außerdem ist die maximale Verbesserung am größten, wenn der Zhang-Ansatz mit dem Pipeline-Ansatz verknüpft wurde. Da der Pipeline-Ansatz immer viel bessere Ergebnisse erzielt als der Zhang-Ansatz, ist es auch nachvollziehbar, dass das Beisteuern des Pipeline-Ansatzes zu einer deutlichen Verbesserung führt.

Bis jetzt habe ich die Kompositionen nur mit dem Durchschnitt der Ergebnisse der zugrundeliegenden Kompositionen verglichen. Dies kann schon einige Erkenntnisse geben, doch in der Praxis will ich die bestmöglichen Werte erreichen. Nur weil eine Komposition besser ist als der Durchschnitt, heißt es nicht, dass diese auch die besten Ergebnisse liefert. Aus diesem Grund habe ich die besten drei Kompositionen nun noch einmal mit dem besten atomaren Ansatz verglichen. In Abbildung 7.15 sind die F2-Werte des Pipeline-Ansatzes in orange zu sehen. Daneben sind die F2-Werte der drei besten Kompositionen aufgetragen, die den Pipeline-Ansatz beinhalten. Es ist zu sehen, dass die drei Kompositionen zunächst einmal alle exakt dieselben Werte haben. Sie haben unabhängig von einander dieselben Trace-Links extrahiert. Das einzige Projekt, indem die Kompositionen eine kleine Verschlechterung zu verbuchen haben, ist Teammates. Dort sind die Kompositionen jedoch nur um 0,004 schlechter. Dafür ist eine kleine Verbesserung von 0,012 beim TeaStore zu

verzeichnen und sogar eine Verbesserung von 0,068 bei MediaStore. Das bedeutet, dass die Kompositionen nicht nur gleichauf mit dem besten atomaren Ansatz waren, sondern sogar in manchen Fällen um einiges besser.

Auch wichtig anzumerken ist, dass es im Allgemeinen vermutlich nicht DEN besten atomaren Ansatz gibt. Während ein Ansatz bei der Gewinnung der Trace-Links aus einem Projekt herausragende Ergebnisse erzielt, sollte das nächste Projekt vielleicht viel besser mit einem anderen bearbeitet werden. Das heißt, eine Komposition mit dem besten Ansatz pro Projekt zu vergleichen ist nicht zielführend. Stattdessen sollte der Ansatz gewählt werden, der im Schnitt die besten Ergebnisse birgt, um die Komposition dann mit diesem zu vergleichen. In diesem Fall macht das jedoch keinen Unterschied, da der Pipeline-Ansatz für diese Projekte immer die besten Werte erzielt. Trotzdem konnte hier der Pipeline-Ansatz durch einige Kompositionen übertroffen werden.

Eine sehr hilfreiche Eigenschaft von einigen Kompositionen ist außerdem, dass sie Fehler, beziehungsweise schlechte Ergebnisse eines Ansatzes, kompensieren können. In Abbildung 7.16 sind beispielsweise die F2-Werte des Zhang-Ansatzes und des naiven Ansatzes nach Projekt aufgetragen (im Bild in orange). Außerdem sind die Werte der *Average-Probability-Composition* und der *(High-Average-Probability)-Union-(High-Precision)-Composition* mit ihnen als Basis aufgetragen (im Bild in blau). Im Falle von MediaStore sind sehr ähnliche Werte über alle vier Recovery-Ansätze hinweg zu sehen. Bei Teammates und TeaStore ist ein deutlich niedrigerer Wert beim Zhang-Ansatz zu erkennen. In den Kompositionen spiegelt sich dies jedoch überhaupt nicht wieder. Die Werte der Kompositionen sind exakt so gut wie die des naiven Ansatzes. Diese Kompositionen konnten somit die schlechten Werte des Zhang-Ansatzes auf diesen Projekten abfedern.

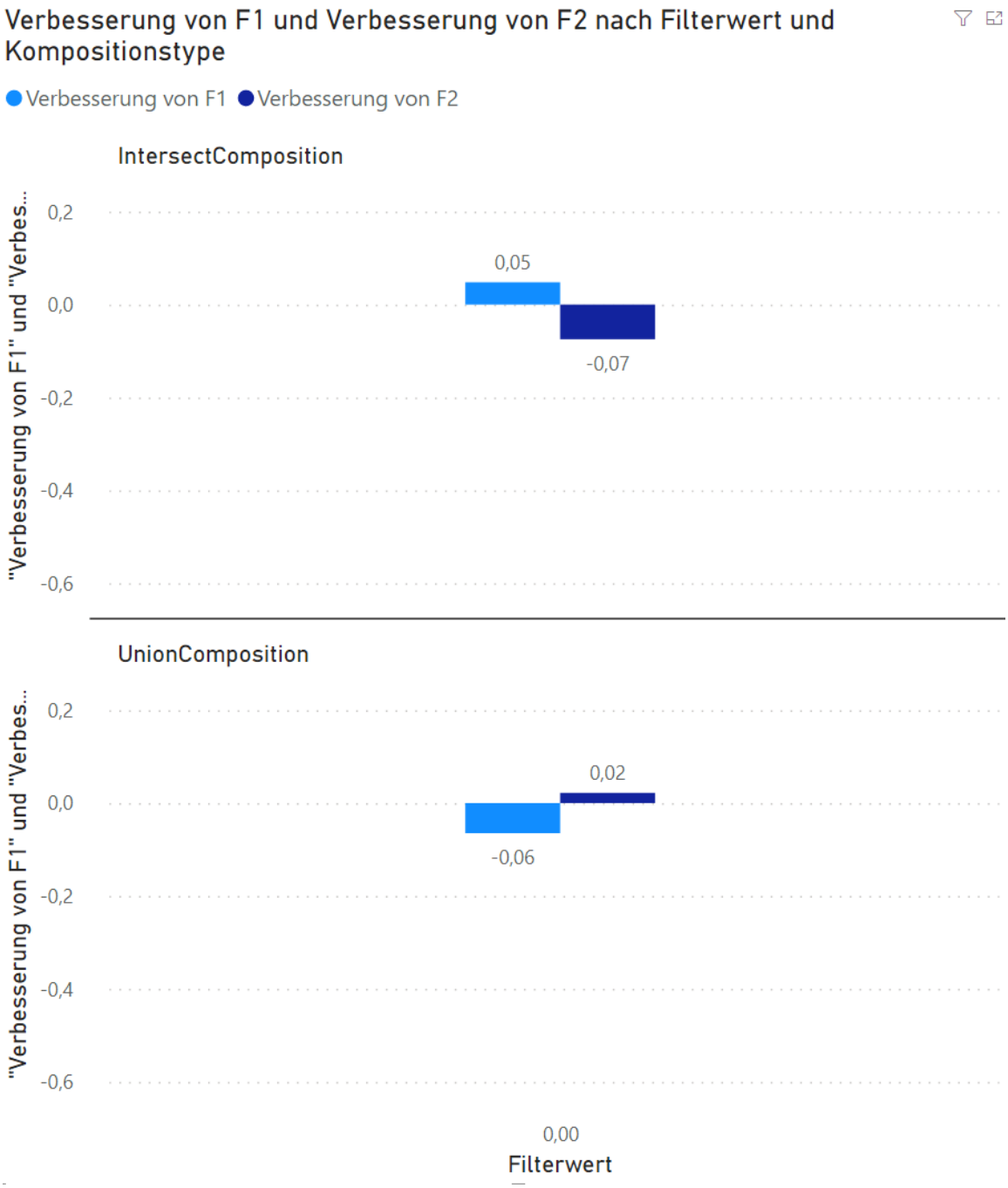


Abbildung 7.9.: Verbesserung des F1- und F2-Wertes der Kompositionen ohne Filter

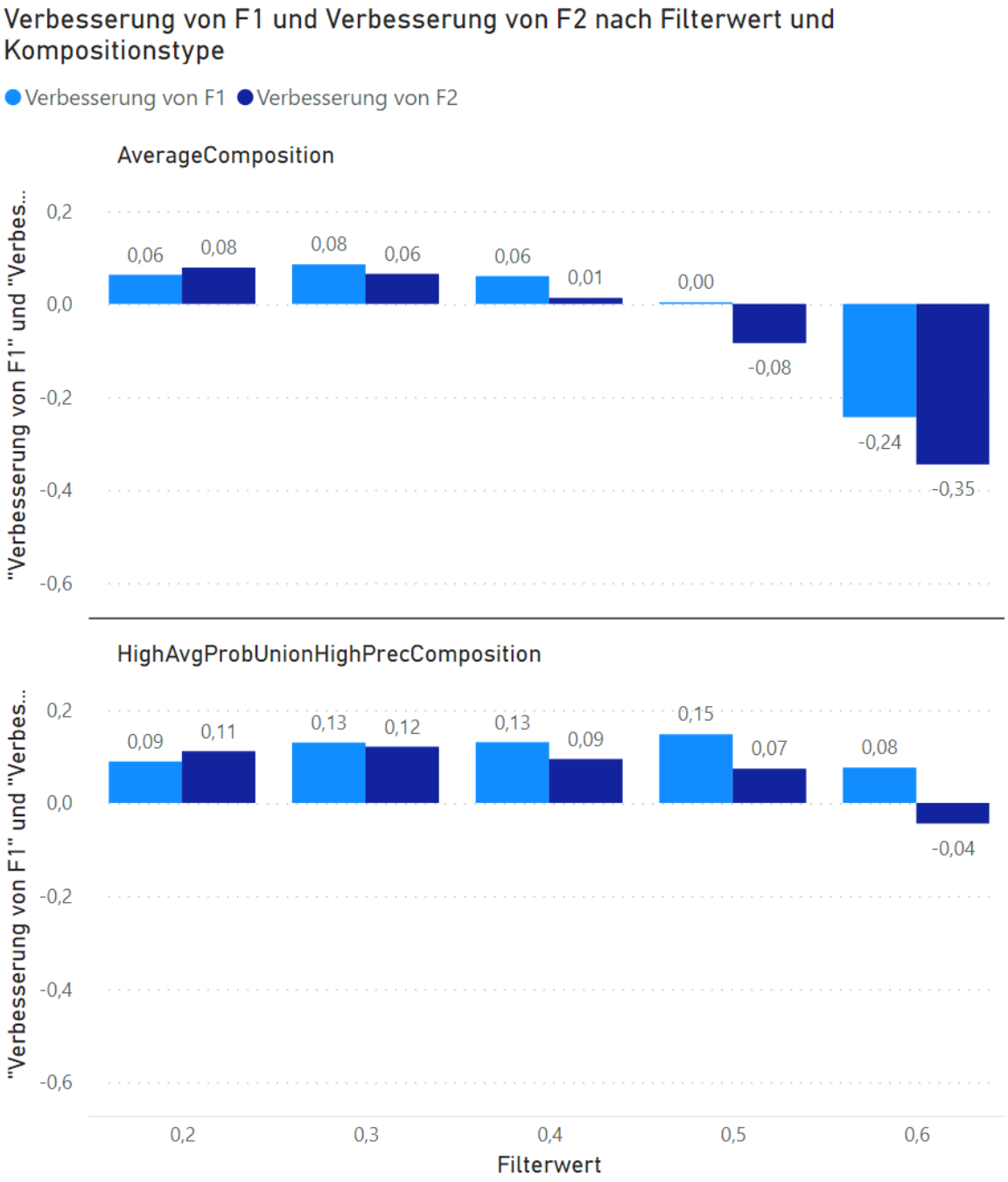


Abbildung 7.10.: Verbesserung des F1- und F2-Wertes der *Average-Probability-Composition* und der *(High-AverageProbability)-Union-(High-Precision)-Composition*

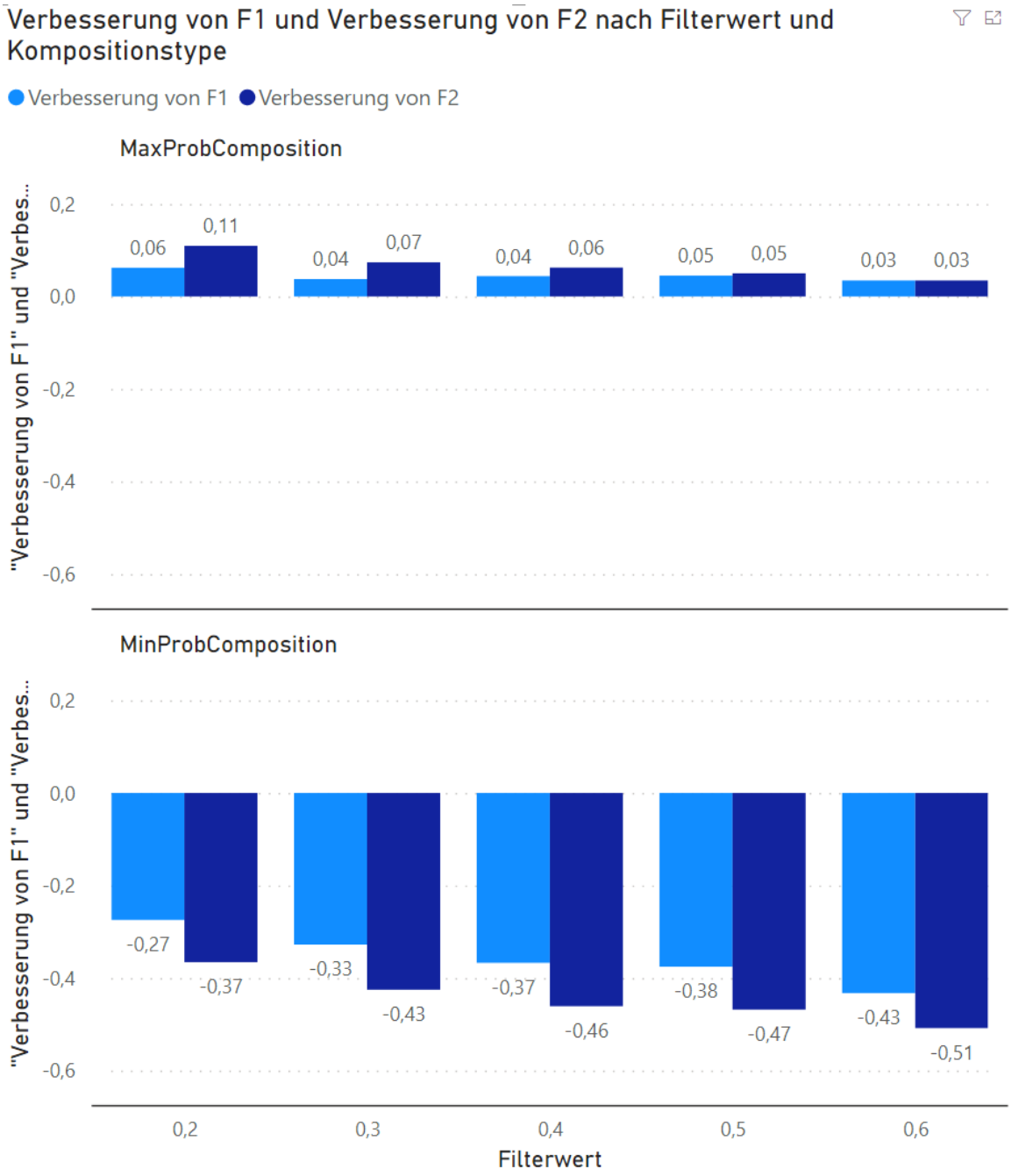


Abbildung 7.11.: Verbesserung des F1- und F2-Wertes der *Minimum-Probability-Composition* und der *Maximum-Probability-Composition*

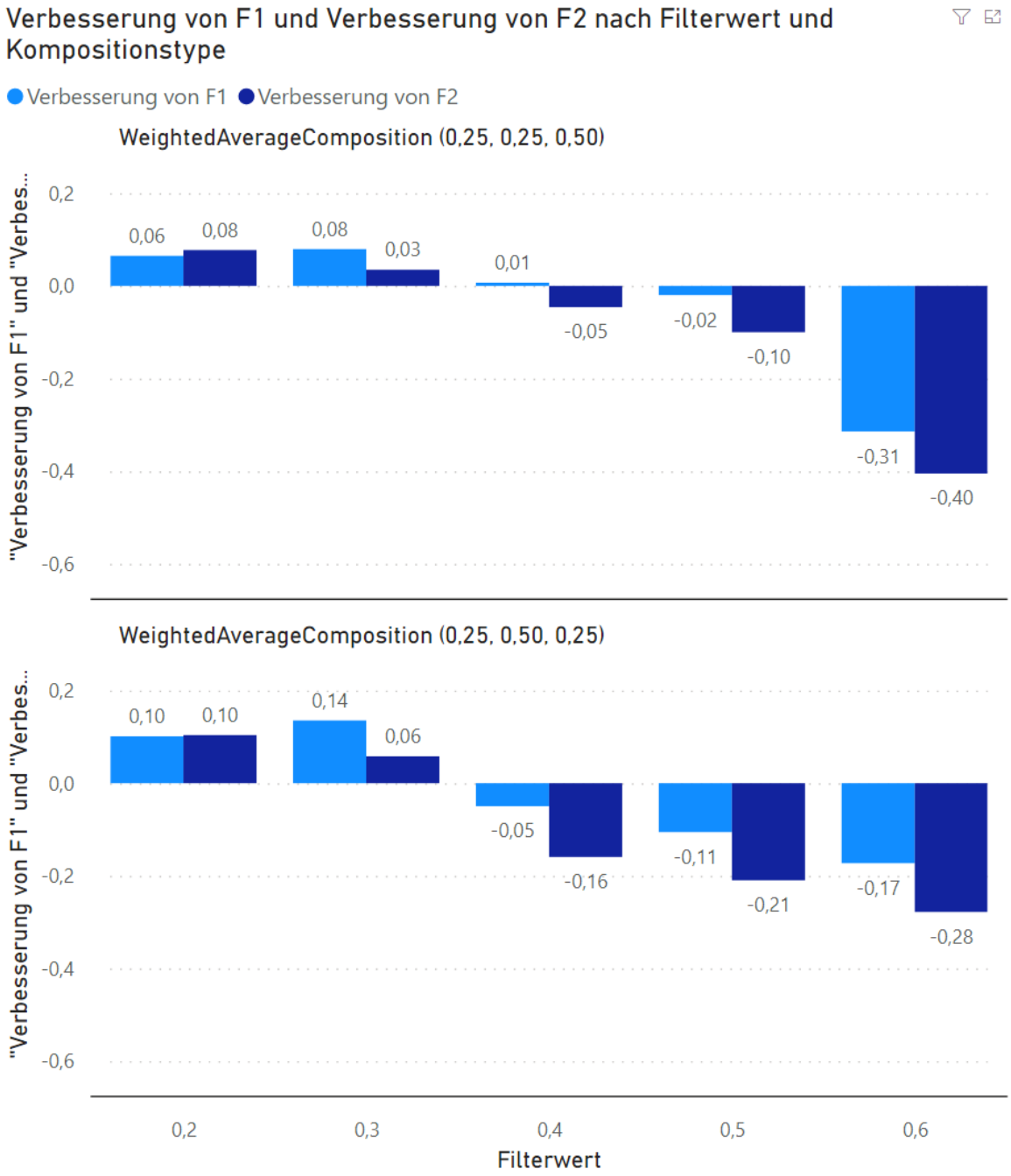


Abbildung 7.12.: Verbesserung des F1- und F2-Wertes der *Weighted-Average-Probability-Composition*

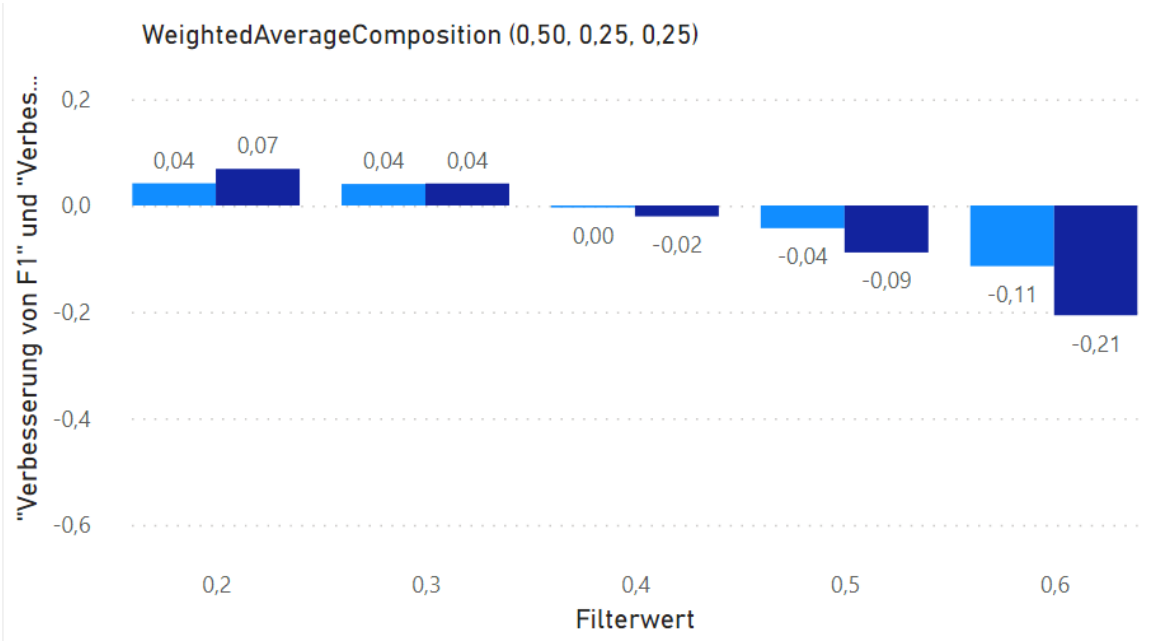


Abbildung 7.13.: Verbesserung des F1- und F2-Wertes der *Weighted-Average-Probability-Composition*

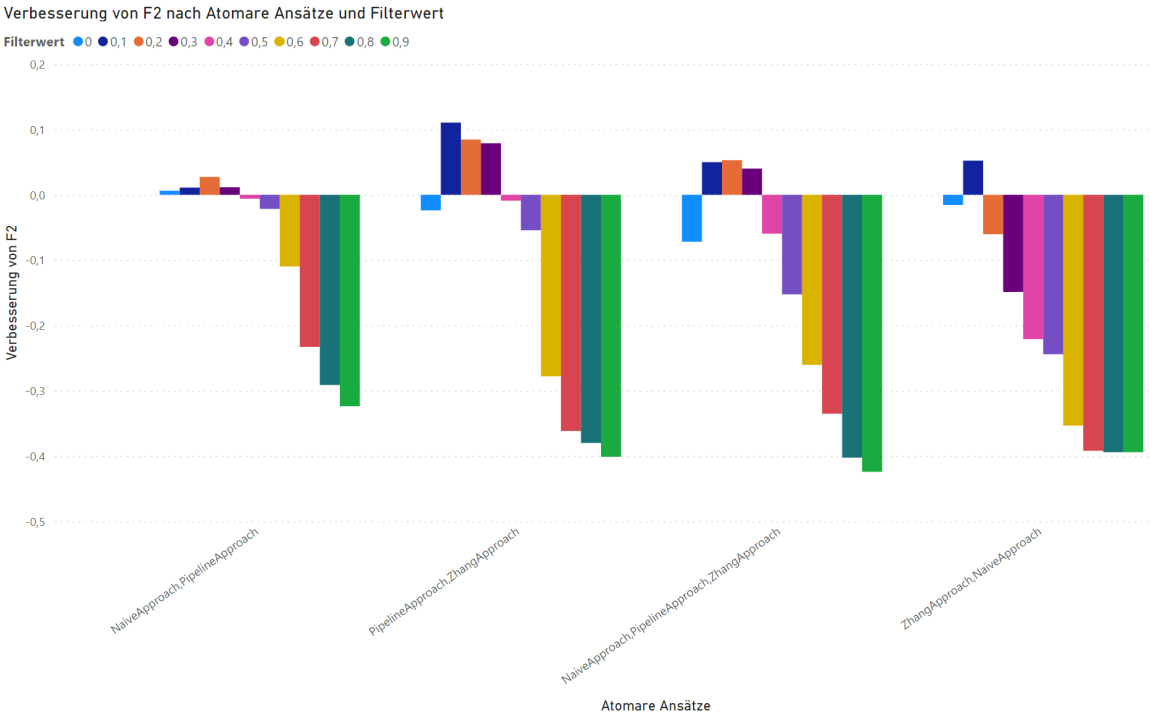


Abbildung 7.14.: Verbesserung des F2-Wertes nach atomaren Ansätzen und Filterwerten

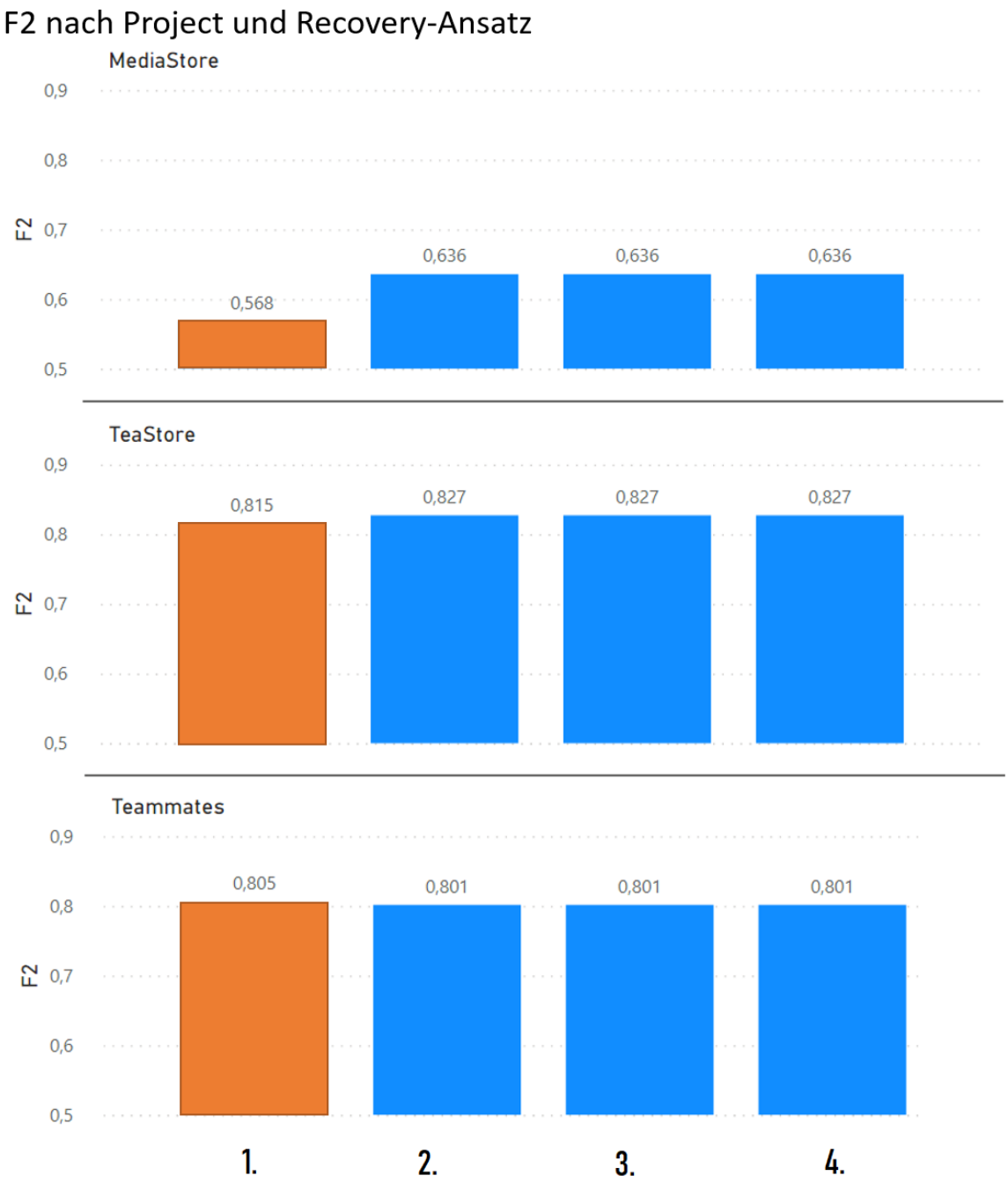


Abbildung 7.15.: Die besten drei Kompositionen im Vergleich zum besten atomaren Ansatz

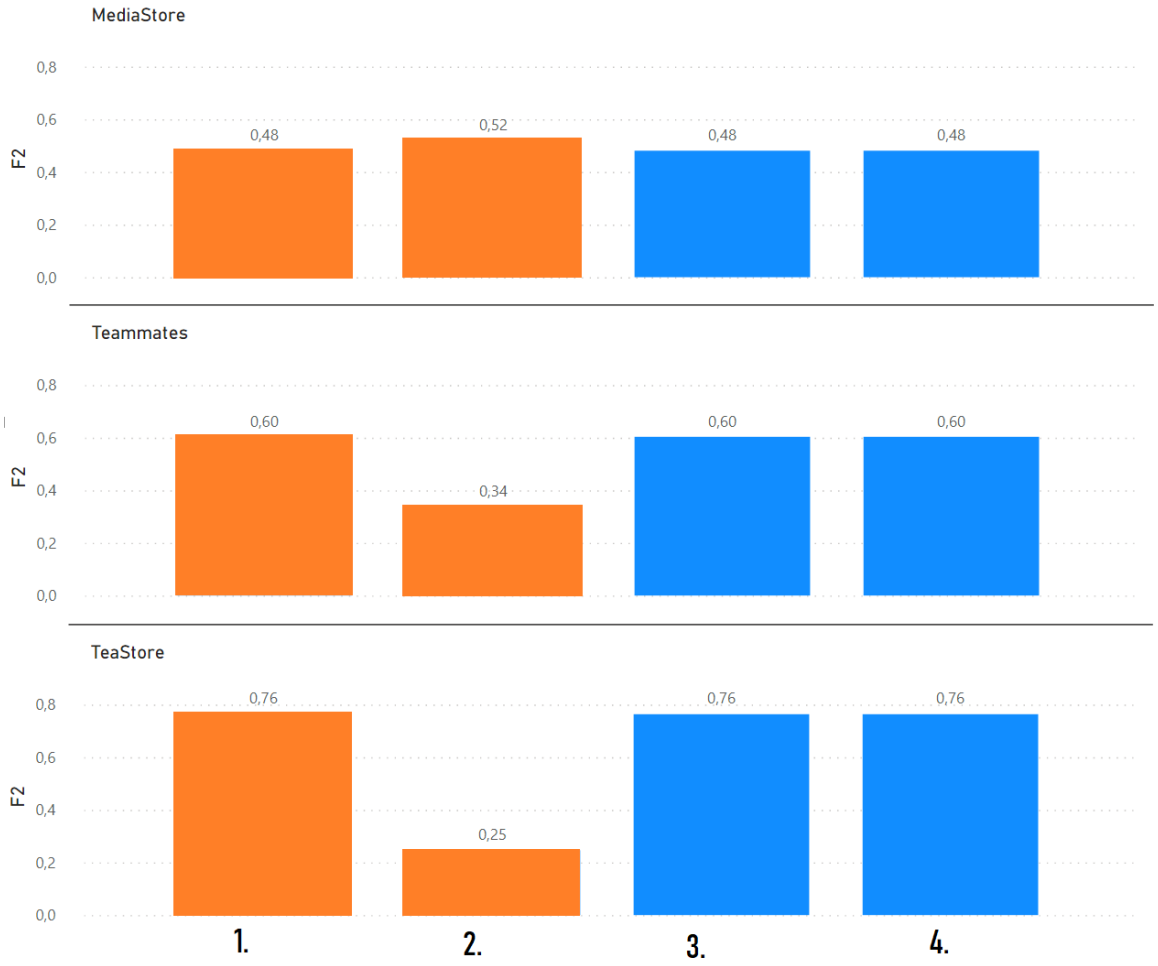


Abbildung 7.16.: Zhang- und naiver Ansatz im Vergleich zu der *Average-Probability-Composition* und der *(High-AverageProbability)-Union-(High-Precision)-Composition* mit Zhang- und naivem Ansatz als Basis.

7.3. Aufwand der Komposition

Um die untergeordnete zweite Forschungsfrage, wie effizient eine Komposition in der Implementierung und Ausführung im Vergleich zu atomaren Ansätzen ist, zu beantworten, werde ich nun noch etwas über den zeitlichen Aufwand in der Entwicklung und der Ausführung der Kompositionen eingehen.

Ich persönlich habe nur den naiven Ansatz implementiert und dieser konnte sehr viele Codeabschnitte von den anderen Ansätzen wiederverwenden. Beispielsweise die Logik, die das PCM-Modell ausliest und dort die Identifizierter extrahiert. Deswegen kann ich keine genau Zeitangabe für die Implementierungsdauer eines Recovery-Ansatzes geben. Jedoch ist das auch nicht nötig, da diese von so vielen schwankenden Faktoren abhängt. Was jedoch wichtig ist, ist die Größenordnung des Aufwands. Die Implementierungsdauer eines atomaren Ansatzes überschattet die einer Komposition bei Weitem. Auch wenn die Angabe von Zeilen Programmcode oft kein sinnvoller Maßstab ist, gibt es hier immerhin einen Anhaltspunkt. Der Zhang-Ansatz hat beispielsweise ca. 1500 Zeilen Code. Dahingegen hat meine größte Komposition nicht einmal 100.

Auch was die Ausführungszeit angeht, sind die Kompositionen um einiges leichtgewichtiger. Während auf meinem Computer (Intel Core I7-Prozessor und Java im Debug-Modus) das Ausführen der atomaren Ansätze zwischen 40 Sekunden (naiver Ansatz) und 15 Minuten (Zhang- und Pipeline-Ansatz) dauerte, konnte die darauf folgende Komposition in wenigen Millisekunden berechnet werden. Um jedoch die Komposition erst einmal durchführen zu können, mussten die atomaren Ansätze vorher die Trace-Links berechnet haben, auf deren Basis die Komposition stattfindet.

Wenn man die Ausführung der atomaren Ansätze sequentiell ablaufen lässt, dann steigt die Ausführungszeit natürlich auch mit der Anzahl an atomaren Ansätzen. Jedoch hängen die einzelnen atomaren Ansätze nicht von einander ab und somit können sie völlig parallel ausgeführt werden. Mit den nötigen Hardware-Ressourcen kann dann die Ausführungszeit auf die des langsamsten atomaren Ansatzes heruntergebrochen werden (plus einige Millisekunden für die Komposition).

8. Folgerung

Nachdem ich im letzten Kapitel die Ergebnisse der Auswertung dargestellt habe, werde ich nun auf die beiden Forschungsfragen eingehen. Als erstes darauf, wie sich die Güte der Recovery-Ansätze durch Komposition verändert. Danach betrachte ich die Frage, wie effizient sich Kompositionen implementieren und ausführen lassen. Abschließend werde ich darauf eingehen, was daraus ableitbar ist und was dies für die Trace-Link-Recovery bedeuten kann.

8.1. Veränderung der Güte durch Kompositionen

1. Wie verändert sich die Güte der Trace-Link-Recovery durch verschiedene Kompositionen von Trace-Link-Recovery-Ansätzen im Vergleich zu den Einzelansätzen?

Viele Kompositionen haben eine deutliche Verschlechterung des F2-Wertes herbeigeführt. Jedoch waren diese meist mit schlichtweg zu extremen Filterwerten versehen und nur der Vollständigkeit wegen aufgeführt. Im Kontrast dazu gab es einige Kompositionen, die selbst die Güte (F2-Wert) des besten atomaren Ansatzes (dem Pipeline-Ansatz) um bis zu 0,02 übertroffen haben. Darunter z.B. die *Average-Probobability-Composition*, die *(High-Average-Probability)-Union-(High-Precision)-Composition*, die *Maximum-Probobability-Composition* oder die *Weighted-Average-Composition.*

Vor allem jedoch können Fehler bzw. schlechte Ergebnisse eines einzelnen atomaren Ansatzes abgefedert werden. Wenn ein atomarer Ansatz auf einem Projekt aus irgendeinem Grund schlechte Ergebnisse liefert, können die anderen Ansätze diese Schwäche ausgleichen.

Auch beim Vergleich der Kompositionen mit dem Durchschnitt der ihnen zugrundeliegenden atomaren Ansätze konnten manche Kompositionen, wie beispielsweise die *(High-Average-Probability)-Union-(High-Precision)-Composition* eine Verbesserung des F2-Wertes von bis zu 0,12 herbeiführen.

Es ist außerdem möglich, durch eine Komposition gezielt die Ausgabe zu beeinflussen. Wenn keine besonders hohe Ausbeute gefordert ist, stattdessen aber eine hohe Präzision, dann kann durch Anpassen der Filterwerte und verwenden von beispielsweise der Schnittmengenvereinigung auf einfache Weise eine hohe Präzision erreicht werden.

8.2. Effizienz von Kompositionen

2. Wie effizient ist eine Komposition in der Implementierung und der Ausführung im Vergleich zu den einzelnen Ansätzen?

Auch wenn Kompositionen sowohl in der Ausführung als auch in der Implementierung viel leichtgewichtiger sind als atomare Ansätze, ist klar, dass sie diese nicht ersetzen können. Es werden zwangsläufig immer noch atomare Ansätze benötigt, die komponiert werden können. Jedoch können Kompositionen als leichtgewichtige Verbesserung verwendet werden. Das gilt vor allem, wenn die atomaren Ansätze parallel ausgeführt werden.

Jedoch muss man leider erst einmal mehrere atomare Ansätze haben, um sie verwenden zu können. Das stellt potentiell eine Hürde dar, da die Implementierung von gleich mehreren atomaren Ansätzen auch dementsprechend aufwendiger ist. Wenn diese jedoch schon zur Verfügung stehen, ist die Komposition ein durchaus sinnvoller Schritt.

8.3. Bedrohung der Validität

Bei der Evaluation und den Beobachtungen existieren verschiedene Bedrohungen der Validität, die ich im folgenden auflisten werde.

In dieser Arbeit werden Kompositionen nur mit drei konkreten atomaren Ansätzen und an drei Software-Projekten evaluiert. Die Ergebnisse sind somit nicht allgemeingültig. Jedoch ergeben sich Hinweise darauf, ob ein solches Verfahren die Trace-Link-Gewinnung auch bei anderen Ansätzen und Software-Projekten verbessern könnte.

Außerdem können Kompositionen natürlich auch beliebig Komplex werden, sodass meine Aussagen zur effizienten Implementierung und Ausführung gesprengt werden. Doch auch diese Aussage sollte für die meisten praktikablen Kompositionen zutreffen.

9. Zusammenfassung

Das Erstellen von Trace-Links die beispielsweise Dokumentation mit Entwurfsmodellen verknüpfen ist ein wertvoller Bestandteil der Softwareentwicklung. Da ein manuelles Herauslesen der Trace-Links oft nicht praktikabel ist, sollte dieser Prozess automatisiert werden. In dieser Arbeit habe ich mich damit befasst, verschiedene Recovery-Ansätze durch Kompositionen miteinander zu verknüpfen. Ich habe die drei atomaren Recovery-Ansätze, Pipeline, Zhang- und naiver Ansatz auf verschiedene Weisen verknüpft. Zum Einen anhand der beiden Mengenoperationen, Schnittmenge und der Mengenvereinigung und zum Anderen durch die auf den Wahrscheinlichkeitswerten der Trace-Links basierenden Operationen, Minimum, Maximum und Durchschnitt. Als letztes habe ich noch eine Komposition aus anderen Kompositionen aufgebaut.

Anschließend habe ich jeweils von all diese Kompositionen und atomaren Ansätze Trace-Links dreier Software-Projekte ermitteln lassen. Diese Trace-Links wurden mit einem manuell erarbeiteten Goldstandard verglichen um die Werte Ausbeute, Präzision, F1 und F2 zu erhalten, die als Güte gedient haben.

Zu sehen war, dass der Pipeline-Ansatz von den drei atomaren Ansätzen durchweg die besten Ergebnisse lieferte. Der Zhang Ansatz hingegen Schnitt bei zwei Projekten eher schlecht ab.

Bei den Komposition gab es einige, die eine Verbesserung im Vergleich zu sowohl dem Durchschnitt über deren Bestandteile als auch zum Pipeline-Ansatz herbeiführen konnten. Darunter waren z.B. die *Average-Probability-Composition*, die *(High-AverageProbability)-Union-(High-Precision)-Composition* oder die *Weighted-Average-Probability-Composition*. Diese Kompositionen waren außerdem in der Lage wenn ein atomarer Ansatz bei einem Projekt nur schlechte Ergebnisse lieferte dies durch die anderen beiden Ansätze zu kompensieren.

Bei wahrscheinlichkeitsbasierten Kompositionen fiel auf, dass es wichtig ist, dass alle Ansätze eine ähnliche Größenordnung der Wahrscheinlichkeiten haben. Da der Zhang-Ansatz meist nur sehr geringe Wahrscheinlichkeiten zurücklieferte war es schwer einen sinnvollen Filterwert zu finden.

Die Eigenschaft dass Kompositionen Fehler einzelner Ansätze mitigieren können, könnte vor allem auch im Bezug auf Trace-Link-Recovery mittels Deep-Learning Vorteile bergen. Es wäre beispielsweise möglich einen untrainierten Deep-Learning-Ansatz in Kombination mit anderen Recovery-Ansätzen zu verwenden. Der untrainierte Ansatz kann dann über einige Projekte hinweg lernen, ohne das Gesamtergebnis zu sehr zu stören. Dann kann den Aussagen des Deep-Learning-Ansatzes Stück für Stück mehr Gewicht zugeordnet werden.

Außerdem wäre es sehr wichtig das Verhalten der Kompositionen auf noch mehr Software-Projekten zu analysieren um einen fundierteren Datensatz zu erhalten, der mehr Aussagekraft besitzt. Gleichermaßen interessant wäre es auch noch weitere Kompositionen zu untersuchen und diese auch mit anderen atomaren Ansätzen auszuführen.

Literatur

[1] URL: https://github.com/ArDoCo/SWATTR.

[2] *ArDoCoWiki*. URL: https://github.com/ArDoCo/Core/wiki/.

[3] Algirdas Avizienis. „The methodology of n-version programming“. In: *Software fault tolerance* 3 (1995), S. 23–46.

[4] Liming Chen und Algirdas Avizienis. „N-version programming: A fault-tolerance approach to reliability of software operation“. In: *Proc. 8th IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-8)*. Bd. 1. 1978, S. 3–9.

[5] David Lorge Parnas; Paul C. Clements. *A rational design process: How and why to fake it*. 1986. DOI: 10.1109/TSE.1986.6312940. URL: https://ieeexplore.ieee.org/abstract/document/6312940.

[6] V.N. Gudivada u. a. „Information retrieval on the World Wide Web“. In: *IEEE Internet Computing* 1.5 (1997), S. 58–68. DOI: 10.1109/4236.623969.

[7] Jin Guo, Jinghui Cheng und Jane Cleland-Huang. „Semantically Enhanced Software Traceability Using Deep Learning Techniques“. In: *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. 2017, S. 3–14. DOI: 10.1109/ICSE.2017.9.

[8] RENATE HAGEDORN, FRANCISCO J. DOBLAS-REYES und T. N. PALMER. „The rationale behind the success of multi-model ensembles in seasonal forecasting – I. Basic concept“. In: *Tellus A* 57.3 (2005), S. 219–233. DOI: https://doi.org/10.1111/j.1600-0870.2005.00103.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1600-0870.2005.00103.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1600-0870.2005.00103.x.

[9] J.H. Hayes, A. Dekhtyar und S.K. Sundaram. „Advancing candidate link generation for requirements tracing: the study of methods“. In: *IEEE Transactions on Software Engineering* 32.1 (2006), S. 4–19. DOI: 10.1109/TSE.2006.3.

[10] Jan Keim, Yves Schneider und Anne Koziolek. „Towards Consistency Analysis between Formal and Informal Software Architecture Artefacts“. In: *2019 IEEE/ACM 2nd International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE)*. 2019, S. 6–12. DOI: 10.1109/ECASE.2019.00010.

[11] Jan Keim u. a. „Trace Link Recovery for Software Architecture Documentation“. In: *Software Architecture*. Hrsg. von Stefan Biffl u. a. Cham: Springer International Publishing, 2021, S. 101–116. ISBN: 978-3-030-86044-8.

[12] Kevin Moran u. a. „Improving the Effectiveness of Traceability Link Recovery Using Hierarchical Bayesian Networks“. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. ICSE '20. Seoul, South Korea: Association for Computing Machinery, 2020, S. 873–885. ISBN: 9781450371216. DOI: 10.1145/3377811.3380418. URL: https://doi.org/10.1145/3377811.3380418.

[13] Rocco Oliveto u. a. „On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery“. In: *2010 IEEE 18th International Conference on Program Comprehension*. 2010, S. 68–71. DOI: 10.1109/ICPC.2010.20.

[14] *Palladio*. URL: https://www.palladio-simulator.com/home/.

[15] Danissa V. Rodriguez und Doris L. Carver. „Multi-Objective Information Retrieval-Based NSGA-II Optimization for Requirements Traceability Recovery“. In: *2020 IEEE International Conference on Electro Information Technology (EIT)*. 2020, S. 271–280. DOI: 10.1109/EIT48999.2020.9208233.

[16] Douglas R Smith. „The design of divide and conquer algorithms“. In: *Science of Computer Programming* 5 (1985), S. 37–58.

[17] Janek Speit. 2021. URL: https://git.scc.kit.edu/i43/stud/janekspeit/-/blob/master/documentation/Dokumentation_Praktikum.pdf.

[18] Yuchen Zhang, Chengcheng Wan und Bo Jin. „An empirical study on recovering requirement-to-code links“. In: *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 2016. DOI: 10.1109/SNPD.2016.7515889.

A. Anhang

Kompositionstyp	Basis	Filter	Präz.	Ausb.	F1	F2
AverageComposition	N,P	0,50	0,69	0,79	0,72	0,75
AverageComposition	N,P,Z	0,30	0,69	0,79	0,72	0,75
HighAvgProbUnionHighPrecComposition	N,P	0,50	0,69	0,79	0,72	0,75
HighAvgProbUnionHighPrecComposition	N,P,Z	0,30	0,69	0,79	0,72	0,75
WeightedAverageComposition (0,25 0,25 0,50)	N,P	0,50	0,69	0,79	0,72	0,75
WeightedAverageComposition (0,25 0,50 0,25)	N,P	0,30	0,69	0,79	0,72	0,75
MaxProbComposition	P,Z	0,10	0,54	0,84	0,65	0,75
AverageComposition	P,Z	0,10	0,53	0,83	0,64	0,74
HighAvgProbUnionHighPrecComposition	P,Z	0,10	0,53	0,83	0,64	0,74
MaxProbComposition	P,Z	0,20	0,53	0,83	0,64	0,74
WeightedAverageComposition (0,25 0,25 0,50)	P,Z	0,10	0,53	0,83	0,64	0,74
WeightedAverageComposition (0,25 0,50 0,25)	N,P,Z	0,20	0,53	0,83	0,64	0,74
WeightedAverageComposition (0,50 0,25 0,25)	P,Z	0,10	0,53	0,83	0,64	0,74
AverageComposition	N,P,Z	0,20	0,51	0,83	0,63	0,73
HighAvgProbUnionHighPrecComposition	N,P,Z	0,20	0,51	0,83	0,63	0,73
AtomicApproach	P	0,00	0,53	0,81	0,64	0,73
AverageComposition	N,P	0,40	0,53	0,81	0,64	0,73
AverageComposition	P,Z	0,20	0,53	0,81	0,64	0,73
AverageComposition	P,Z	0,30	0,53	0,81	0,64	0,73
AverageComposition	P,Z	0,40	0,53	0,81	0,64	0,73
HighAvgProbUnionHighPrecComposition	N,P	0,40	0,53	0,81	0,64	0,73
HighAvgProbUnionHighPrecComposition	P,Z	0,20	0,53	0,81	0,64	0,73
HighAvgProbUnionHighPrecComposition	P,Z	0,30	0,53	0,81	0,64	0,73
HighAvgProbUnionHighPrecComposition	P,Z	0,40	0,53	0,81	0,64	0,73
MaxProbComposition	N,P	0,80	0,53	0,81	0,64	0,73
MaxProbComposition	N,P,Z	0,80	0,53	0,81	0,64	0,73
MaxProbComposition	P,Z	0,30	0,53	0,81	0,64	0,73
MaxProbComposition	P,Z	0,40	0,53	0,81	0,64	0,73
MaxProbComposition	P,Z	0,50	0,53	0,81	0,64	0,73
MaxProbComposition	P,Z	0,60	0,53	0,81	0,64	0,73
MaxProbComposition	P,Z	0,70	0,53	0,81	0,64	0,73
MaxProbComposition	P,Z	0,80	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,25 0,25 0,50)	N,P	0,40	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,25 0,25 0,50)	N,P,Z	0,20	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,25 0,25 0,50)	P,Z	0,20	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,25 0,25 0,50)	P,Z	0,30	0,53	0,81	0,64	0,73

WeightedAverageComposition (0,25 0,25 0,50)	P,Z	0,40	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,25 0,50 0,25)	P,Z	0,10	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,25 0,50 0,25)	P,Z	0,20	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,50 0,25 0,25)	P,Z	0,20	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,50 0,25 0,25)	P,Z	0,30	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,50 0,25 0,25)	P,Z	0,40	0,53	0,81	0,64	0,73
WeightedAverageComposition (0,50 0,25 0,25)	P,Z	0,50	0,53	0,81	0,64	0,73
AverageComposition	N,P	0,30	0,50	0,81	0,62	0,72
HighAvgProbUnionHighPrecComposition	N,P	0,30	0,50	0,81	0,62	0,72
MaxProbComposition	N,P	0,60	0,50	0,81	0,62	0,72
MaxProbComposition	N,P	0,70	0,50	0,81	0,62	0,72
MaxProbComposition	N,P,Z	0,60	0,50	0,81	0,62	0,72
MaxProbComposition	N,P,Z	0,70	0,50	0,81	0,62	0,72
WeightedAverageComposition (0,25 0,25 0,50)	N,P	0,30	0,50	0,81	0,62	0,72
WeightedAverageComposition (0,25 0,50 0,25)	N,P	0,20	0,50	0,81	0,62	0,72
WeightedAverageComposition (0,50 0,25 0,25)	N,P	0,40	0,50	0,81	0,62	0,72
WeightedAverageComposition (0,50 0,25 0,25)	N,P,Z	0,30	0,50	0,81	0,62	0,72
AverageComposition	N,P	0,20	0,45	0,85	0,58	0,71
HighAvgProbUnionHighPrecComposition	N,P	0,20	0,45	0,85	0,58	0,71
MaxProbComposition	N,P	0,40	0,45	0,85	0,58	0,71
MaxProbComposition	N,P,Z	0,40	0,45	0,85	0,58	0,71
WeightedAverageComposition (0,25 0,25 0,50)	N,P	0,20	0,45	0,85	0,58	0,71
WeightedAverageComposition (0,25 0,25 0,50)	N,P,Z	0,10	0,45	0,85	0,58	0,71
WeightedAverageComposition (0,25 0,50 0,25)	N,P,Z	0,10	0,45	0,85	0,58	0,71
WeightedAverageComposition (0,50 0,25 0,25)	N,P,Z	0,20	0,45	0,85	0,58	0,71
WeightedAverageComposition (0,25 0,50 0,25)	P,Z	0,30	0,71	0,72	0,70	0,71
MaxProbComposition	N,P	0,50	0,47	0,81	0,59	0,70
MaxProbComposition	N,P,Z	0,50	0,47	0,81	0,59	0,70
WeightedAverageComposition (0,50 0,25 0,25)	N,P	0,30	0,47	0,81	0,59	0,70
HighAvgProbUnionHighPrecComposition	P,Z	0,50	0,67	0,73	0,68	0,70
AverageComposition	N,P,Z	0,10	0,42	0,88	0,55	0,70
HighAvgProbUnionHighPrecComposition	N,P,Z	0,10	0,42	0,88	0,55	0,70
MaxProbComposition	N,P	0,30	0,42	0,88	0,55	0,70
MaxProbComposition	N,P,Z	0,30	0,42	0,88	0,55	0,70
WeightedAverageComposition (0,25 0,50 0,25)	N,P	0,10	0,42	0,88	0,55	0,70
WeightedAverageComposition (0,50 0,25 0,25)	N,P	0,20	0,42	0,88	0,55	0,70
AverageComposition	N,P	0,10	0,41	0,88	0,54	0,68
HighAvgProbUnionHighPrecComposition	N,P	0,10	0,41	0,88	0,54	0,68
MaxProbComposition	N,P	0,10	0,41	0,88	0,54	0,68
MaxProbComposition	N,P	0,20	0,41	0,88	0,54	0,68
MaxProbComposition	N,P,Z	0,10	0,41	0,88	0,54	0,68
MaxProbComposition	N,P,Z	0,20	0,41	0,88	0,54	0,68
UnionComposition	N,P	0,00	0,41	0,88	0,54	0,68
WeightedAverageComposition (0,25 0,25 0,50)	N,P	0,10	0,41	0,88	0,54	0,68
WeightedAverageComposition (0,50 0,25 0,25)	N,P	0,10	0,41	0,88	0,54	0,68

WeightedAverageComposition (0,50 0,25 0,25)	N,P,Z	0,10	0,41	0,88	0,54	0,68
HighAvgProbUnionHighPrecComposition	N,P	0,60	0,81	0,65	0,71	0,68
HighAvgProbUnionHighPrecComposition	N,P	0,70	0,81	0,65	0,71	0,68
HighAvgProbUnionHighPrecComposition	N,P	0,80	0,81	0,65	0,71	0,68
HighAvgProbUnionHighPrecComposition	N,P	0,90	0,81	0,65	0,71	0,68
IntersectComposition	N,P	0,00	0,81	0,65	0,71	0,68
MinProbComposition	N,P	0,10	0,81	0,65	0,71	0,68
MinProbComposition	N,P	0,20	0,81	0,65	0,71	0,68
WeightedAverageComposition (0,25 0,50 0,25)	N,P	0,40	0,81	0,65	0,71	0,68
WeightedAverageComposition (0,25 0,50 0,25)	N,P	0,50	0,81	0,65	0,71	0,68
WeightedAverageComposition (0,25 0,50 0,25)	N,P	0,60	0,81	0,65	0,71	0,68
WeightedAverageComposition (0,25 0,50 0,25)	N,P,Z	0,30	0,81	0,65	0,71	0,68
WeightedAverageComposition (0,25 0,50 0,25)	N,P,Z	0,40	0,81	0,65	0,71	0,68
WeightedAverageComposition (0,25 0,25 0,50)	N,P,Z	0,30	0,84	0,60	0,70	0,64
HighAvgProbUnionHighPrecComposition	N,P,Z	0,40	0,82	0,60	0,69	0,63
UnionComposition	P,Z	0,00	0,31	0,87	0,44	0,62
AtomicApproach	N	0,00	0,56	0,72	0,56	0,62
AverageComposition	Z,N	0,10	0,56	0,72	0,56	0,62
HighAvgProbUnionHighPrecComposition	Z,N	0,10	0,56	0,72	0,56	0,62
MaxProbComposition	Z,N	0,10	0,56	0,72	0,56	0,62
MaxProbComposition	Z,N	0,20	0,56	0,72	0,56	0,62
WeightedAverageComposition (0,25 0,25 0,50)	Z,N	0,10	0,56	0,72	0,56	0,62
AverageComposition	P,Z	0,50	0,70	0,63	0,61	0,61
WeightedAverageComposition (0,25 0,25 0,50)	P,Z	0,50	0,70	0,63	0,61	0,61
WeightedAverageComposition (0,50 0,25 0,25)	P,Z	0,60	0,70	0,63	0,61	0,61
AverageComposition	N,P,Z	0,40	0,83	0,57	0,67	0,61
WeightedAverageComposition (0,25 0,50 0,25)	N,P,Z	0,50	0,83	0,57	0,67	0,61
MaxProbComposition	N,P	0,90	0,69	0,62	0,59	0,60
MaxProbComposition	N,P,Z	0,90	0,69	0,62	0,59	0,60
MaxProbComposition	P,Z	0,90	0,69	0,62	0,59	0,60
WeightedAverageComposition (0,50 0,25 0,25)	Z,N	0,10	0,55	0,69	0,55	0,60
AverageComposition	N,P	0,60	0,83	0,55	0,65	0,58
WeightedAverageComposition (0,25 0,25 0,50)	N,P	0,60	0,83	0,55	0,65	0,58
WeightedAverageComposition (0,25 0,50 0,25)	N,P	0,70	0,83	0,53	0,64	0,57
HighAvgProbUnionHighPrecComposition	Z,N	0,40	0,66	0,56	0,59	0,57
HighAvgProbUnionHighPrecComposition	Z,N	0,50	0,66	0,56	0,59	0,57
UnionComposition	N,P,Z	0,00	0,27	0,88	0,39	0,57
WeightedAverageComposition (0,50 0,25 0,25)	N,P	0,50	0,44	0,63	0,51	0,57
WeightedAverageComposition (0,50 0,25 0,25)	N,P,Z	0,40	0,46	0,61	0,51	0,56
HighAvgProbUnionHighPrecComposition	Z,N	0,30	0,64	0,56	0,58	0,56
HighAvgProbUnionHighPrecComposition	Z,N	0,20	0,60	0,60	0,55	0,56
HighAvgProbUnionHighPrecComposition	N,P,Z	0,50	0,84	0,51	0,63	0,55
HighAvgProbUnionHighPrecComposition	N,P,Z	0,60	0,84	0,51	0,63	0,55
WeightedAverageComposition (0,25 0,50 0,25)	Z,N	0,10	0,55	0,61	0,52	0,55
WeightedAverageComposition (0,25 0,50 0,25)	Z,N	0,20	0,62	0,49	0,54	0,51

UnionComposition	Z,N	0,00	0,36	0,73	0,40	0,50
MinProbComposition	N,P	0,30	0,78	0,43	0,55	0,47
MaxProbComposition	Z,N	0,30	0,49	0,49	0,44	0,45
WeightedAverageComposition (0,50 0,25 0,25)	N,P,Z	0,50	0,73	0,46	0,48	0,45
WeightedAverageComposition (0,50 0,25 0,25)	N,P	0,60	0,68	0,46	0,47	0,45
HighAvgProbUnionHighPrecComposition	Z,N	0,60	0,64	0,44	0,48	0,45
HighAvgProbUnionHighPrecComposition	Z,N	0,70	0,64	0,44	0,48	0,45
HighAvgProbUnionHighPrecComposition	Z,N	0,80	0,64	0,44	0,48	0,45
HighAvgProbUnionHighPrecComposition	Z,N	0,90	0,64	0,44	0,48	0,45
IntersectComposition	Z,N	0,00	0,64	0,44	0,48	0,45
HighAvgProbUnionHighPrecComposition	P,Z	0,60	0,73	0,39	0,51	0,43
HighAvgProbUnionHighPrecComposition	P,Z	0,70	0,73	0,39	0,51	0,43
HighAvgProbUnionHighPrecComposition	P,Z	0,80	0,73	0,39	0,51	0,43
HighAvgProbUnionHighPrecComposition	P,Z	0,90	0,73	0,39	0,51	0,43
IntersectComposition	P,Z	0,00	0,73	0,39	0,51	0,43
HighAvgProbUnionHighPrecComposition	N,P,Z	0,70	0,82	0,39	0,52	0,43
HighAvgProbUnionHighPrecComposition	N,P,Z	0,80	0,82	0,39	0,52	0,43
HighAvgProbUnionHighPrecComposition	N,P,Z	0,90	0,82	0,39	0,52	0,43
IntersectComposition	N,P,Z	0,00	0,82	0,39	0,52	0,43
AverageComposition	Z,N	0,20	0,64	0,44	0,44	0,43
WeightedAverageComposition (0,25 0,25 0,50)	Z,N	0,20	0,64	0,44	0,44	0,43
WeightedAverageComposition (0,50 0,25 0,25)	Z,N	0,20	0,55	0,44	0,44	0,42
MaxProbComposition	Z,N	0,40	0,68	0,39	0,42	0,39
WeightedAverageComposition (0,25 0,50 0,25)	Z,N	0,30	0,77	0,35	0,44	0,38
WeightedAverageComposition (0,25 0,25 0,50)	N,P,Z	0,40	0,95	0,33	0,46	0,37
MinProbComposition	N,P	0,40	0,90	0,33	0,46	0,37
AtomicApproach	Z	0,00	0,35	0,45	0,32	0,37
WeightedAverageComposition (0,25 0,50 0,25)	N,P,Z	0,60	0,84	0,32	0,44	0,36
MaxProbComposition	Z,N	0,50	0,69	0,32	0,40	0,34
MinProbComposition	N,P	0,50	0,95	0,31	0,43	0,34
AverageComposition	Z,N	0,30	0,80	0,31	0,40	0,34
WeightedAverageComposition (0,25 0,25 0,50)	Z,N	0,30	0,80	0,31	0,40	0,34
AverageComposition	N,P,Z	0,50	0,95	0,29	0,41	0,33
AverageComposition	N,P	0,70	0,84	0,27	0,37	0,30
WeightedAverageComposition (0,25 0,25 0,50)	N,P	0,70	0,84	0,27	0,37	0,30
WeightedAverageComposition (0,25 0,50 0,25)	N,P	0,80	0,84	0,27	0,37	0,30
WeightedAverageComposition (0,50 0,25 0,25)	Z,N	0,30	0,69	0,27	0,35	0,29
WeightedAverageComposition (0,50 0,25 0,25)	N,P	0,70	0,95	0,25	0,34	0,28
AverageComposition	N,P,Z	0,60	0,95	0,24	0,32	0,27
AverageComposition	Z,N	0,40	0,95	0,24	0,32	0,27
WeightedAverageComposition (0,25 0,25 0,50)	N,P,Z	0,50	0,95	0,24	0,32	0,27
WeightedAverageComposition (0,25 0,25 0,50)	Z,N	0,40	0,95	0,24	0,32	0,27
AverageComposition	N,P	0,80	0,95	0,23	0,30	0,25
AverageComposition	Z,N	0,50	0,95	0,23	0,30	0,25
MinProbComposition	N,P	0,60	0,95	0,23	0,30	0,25

WeightedAverageComposition (0,25 0,25 0,50)	N,P	0,80	0,95	0,23	0,30	0,25
WeightedAverageComposition (0,25 0,25 0,50)	Z,N	0,50	0,95	0,23	0,30	0,25
WeightedAverageComposition (0,25 0,50 0,25)	N,P,Z	0,70	0,95	0,23	0,30	0,25
WeightedAverageComposition (0,50 0,25 0,25)	N,P,Z	0,60	0,95	0,23	0,30	0,25
MaxProbComposition	Z,N	0,60	0,66	0,23	0,30	0,25
AverageComposition	N,P	0,90	0,62	0,21	0,27	0,23
MaxProbComposition	Z,N	0,70	0,62	0,21	0,27	0,23
MaxProbComposition	Z,N	0,80	0,62	0,21	0,27	0,23
MaxProbComposition	Z,N	0,90	0,62	0,21	0,27	0,23
MinProbComposition	N,P	0,70	0,62	0,21	0,27	0,23
MinProbComposition	N,P	0,80	0,62	0,21	0,27	0,23
MinProbComposition	N,P	0,90	0,62	0,21	0,27	0,23
WeightedAverageComposition (0,25 0,25 0,50)	N,P	0,90	0,62	0,21	0,27	0,23
WeightedAverageComposition (0,25 0,50 0,25)	N,P	0,90	0,62	0,21	0,27	0,23
WeightedAverageComposition (0,50 0,25 0,25)	N,P	0,80	0,62	0,21	0,27	0,23
WeightedAverageComposition (0,50 0,25 0,25)	N,P	0,90	0,62	0,21	0,27	0,23
WeightedAverageComposition (0,50 0,25 0,25)	N,P,Z	0,70	0,62	0,21	0,27	0,23
WeightedAverageComposition (0,25 0,50 0,25)	Z,N	0,40	0,56	0,17	0,26	0,20
MinProbComposition	Z,N	0,10	0,83	0,17	0,25	0,20
WeightedAverageComposition (0,50 0,25 0,25)	Z,N	0,40	0,73	0,16	0,25	0,19
WeightedAverageComposition (0,50 0,25 0,25)	Z,N	0,50	0,72	0,15	0,24	0,18
MinProbComposition	N,P,Z	0,10	0,83	0,15	0,23	0,17
MinProbComposition	P,Z	0,10	0,83	0,15	0,23	0,17
WeightedAverageComposition (0,50 0,25 0,25)	Z,N	0,60	0,63	0,12	0,19	0,14
WeightedAverageComposition (0,25 0,50 0,25)	Z,N	0,50	0,44	0,11	0,16	0,12
WeightedAverageComposition (0,25 0,50 0,25)	Z,N	0,60	0,27	0,09	0,14	0,11
WeightedAverageComposition (0,25 0,50 0,25)	P,Z	0,40	0,83	0,08	0,14	0,10
WeightedAverageComposition (0,50 0,25 0,25)	P,Z	0,70	0,83	0,08	0,14	0,10
MinProbComposition	Z,N	0,20	0,67	0,05	0,09	0,06
AverageComposition	P,Z	0,60	0,67	0,04	0,07	0,04
MinProbComposition	N,P,Z	0,20	0,67	0,04	0,07	0,04
MinProbComposition	P,Z	0,20	0,67	0,04	0,07	0,04
WeightedAverageComposition (0,25 0,25 0,50)	P,Z	0,60	0,67	0,04	0,07	0,04
MinProbComposition	N,P,Z	0,30	0,67	0,03	0,06	0,04
MinProbComposition	P,Z	0,30	0,67	0,03	0,06	0,04
MinProbComposition	Z,N	0,30	0,67	0,03	0,06	0,04
WeightedAverageComposition (0,25 0,50 0,25)	P,Z	0,50	0,67	0,03	0,06	0,04
MinProbComposition	P,Z	0,40	0,33	0,03	0,05	0,03
MinProbComposition	P,Z	0,50	0,33	0,03	0,05	0,03
WeightedAverageComposition (0,25 0,50 0,25)	P,Z	0,60	0,33	0,03	0,05	0,03
WeightedAverageComposition (0,25 0,25 0,50)	N,P,Z	0,60	0,67	0,02	0,04	0,03
AverageComposition	P,Z	0,70	0,33	0,01	0,03	0,02
MinProbComposition	N,P,Z	0,40	0,33	0,01	0,03	0,02
MinProbComposition	N,P,Z	0,50	0,33	0,01	0,03	0,02
MinProbComposition	Z,N	0,40	0,33	0,01	0,03	0,02

MinProbComposition	Z,N	0,50	0,33	0,01	0,03	0,02
WeightedAverageComposition (0,25 0,25 0,50)	P,Z	0,70	0,33	0,01	0,03	0,02
WeightedAverageComposition (0,25 0,50 0,25)	P,Z	0,70	0,33	0,01	0,03	0,02
WeightedAverageComposition (0,50 0,25 0,25)	P,Z	0,80	0,33	0,01	0,03	0,02
AverageComposition	N,P,Z	0,70	0,33	0,01	0,02	0,02
WeightedAverageComposition (0,25 0,50 0,25)	Z,N	0,70	0,33	0,01	0,02	0,02
AverageComposition	Z,N	0,60	0,33	0,01	0,02	0,01
WeightedAverageComposition (0,25 0,25 0,50)	Z,N	0,60	0,33	0,01	0,02	0,01
WeightedAverageComposition (0,25 0,50 0,25)	N,P,Z	0,80	0,33	0,01	0,02	0,01
WeightedAverageComposition (0,50 0,25 0,25)	N,P,Z	0,80	0,33	0,01	0,02	0,01
AverageComposition	N,P,Z	0,80	0,00	0,00	0,00	0,00
AverageComposition	N,P,Z	0,90	0,00	0,00	0,00	0,00
AverageComposition	P,Z	0,80	0,00	0,00	0,00	0,00
AverageComposition	P,Z	0,90	0,00	0,00	0,00	0,00
AverageComposition	Z,N	0,70	0,00	0,00	0,00	0,00
AverageComposition	Z,N	0,80	0,00	0,00	0,00	0,00
AverageComposition	Z,N	0,90	0,00	0,00	0,00	0,00
MinProbComposition	N,P,Z	0,60	0,00	0,00	0,00	0,00
MinProbComposition	N,P,Z	0,70	0,00	0,00	0,00	0,00
MinProbComposition	N,P,Z	0,80	0,00	0,00	0,00	0,00
MinProbComposition	N,P,Z	0,90	0,00	0,00	0,00	0,00
MinProbComposition	P,Z	0,60	0,00	0,00	0,00	0,00
MinProbComposition	P,Z	0,70	0,00	0,00	0,00	0,00
MinProbComposition	P,Z	0,80	0,00	0,00	0,00	0,00
MinProbComposition	P,Z	0,90	0,00	0,00	0,00	0,00
MinProbComposition	Z,N	0,60	0,00	0,00	0,00	0,00
MinProbComposition	Z,N	0,70	0,00	0,00	0,00	0,00
MinProbComposition	Z,N	0,80	0,00	0,00	0,00	0,00
MinProbComposition	Z,N	0,90	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,25 0,50)	N,P,Z	0,70	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,25 0,50)	N,P,Z	0,80	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,25 0,50)	N,P,Z	0,90	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,25 0,50)	P,Z	0,80	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,25 0,50)	P,Z	0,90	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,25 0,50)	Z,N	0,70	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,25 0,50)	Z,N	0,80	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,25 0,50)	Z,N	0,90	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,50 0,25)	N,P,Z	0,90	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,50 0,25)	P,Z	0,80	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,50 0,25)	P,Z	0,90	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,50 0,25)	Z,N	0,80	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,25 0,50 0,25)	Z,N	0,90	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,50 0,25 0,25)	N,P,Z	0,90	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,50 0,25 0,25)	P,Z	0,90	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,50 0,25 0,25)	Z,N	0,70	0,00	0,00	0,00	0,00

WeightedAverageComposition (0,50 0,25 0,25)	Z,N	0,80	0,00	0,00	0,00	0,00
WeightedAverageComposition (0,50 0,25 0,25)	Z,N	0,90	0,00	0,00	0,00	0,00

Tabelle A.1.: Ergebnisse aller ausgewerteter Konfigurationen. Die Spalte „Basis“ gibt die atomaren Bestandteile an, wobei „P“, „N“ und „Z“ für den Pipeline-, Naiven- und Zhang-Ansatz stehen. Die Spalte „Filter“ gibt den Wahrscheinlichkeitsfilterwert an, der verwendet wurde. Darauf folgen Präzision, Ausbeute, F1- und F2-Werte.