Karlsruhe Institute of Technology

Wissensanreicherung von Begriffen im Quelltext

Bachelorarbeit
von

Daniel Jungkind

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

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

Bearbeitungszeit: 22.12.2021 – 22.04.2022

Kurzfassung

Anforderungsrückverfolgung spielt im Bereich der Softwarewartung eine große Rolle. Worteinbettungsbasierte Verfahren zur Anforderungsrückverfolgung nutzen Wörter, die in Anforderungen und Quelltext vorkommen, um Rückverfolgbarkeitsverbindungen herzustellen. Semantisch äquivalente aber sprachlich unterschiedliche Formulierungen können dies erschweren. Wissen über derartige semantische Zusammenhänge zwischen verschiedenen Begriffen kann helfen, die Rückverfolgbarkeit zu verbessern. Diese Arbeit hat zum Ziel, in Quelltext vorkommende natürlichsprachliche Begriffe mit Wissen in Form von semantisch verwandten Begriffen anzureichern, um worteinbettungsbasierte Anforderungsrückverfolgung zu verbessern. Hierzu werden zunächst DBpedia-Artikel bestimmt, welche den Bedeutungen der Begriffe im Quelltext entsprechen. Daraufhin werden die Verbindungen dieser DBpedia-Artikel zu weiteren Artikeln dazu genutzt, um Begriffe zu identifizieren, die das gemeinsame Thema der Eingabe beschreiben. Hierzu werden Kategorien- und Oberbegriffsbeziehungen genutzt, um einen DBpedia-Subgraphen aufzubauen und in diesem Zusammenhangskomponenten zu identifizieren. Zentrale Knoten in diesen Zusammenhangskomponenten liefern dabei Kandidaten für die Themenbeschriftung.

Durch das Hinzufügen dieser Themenbeschriftungen konnten auf den Datensätzen eTour und eAnci Verbesserungen der F_1-Werte von bis zu $+9.4\,\%$ für das Bestimmen von Rückverfolgbarkeitsverbindungen erzielt werden. Dabei lagen die Verbesserungen der Präzisionswerte zwischen $+1.5\,\%$ und $+11.5\,\%$.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Anforderungsrückverfolgung spielt eine wichtige Rolle in der Softwareentwicklung und -wartung. Als solche wird das Herstellen von Rückverfolgbarkeitsverbindungen zwischen Anforderungen und Implementierungsartefakten wie Quelltext verstanden [Got+12]. Solche Verbindungen helfen beispielsweise, die Einhaltung der Anforderungen sicherzustellen oder diesbezügliche Fehler zu beheben. Da manuelle Anforderungsrückverfolgung zu aufwendig ist, um verbreitet Anwendung zu finden, existiert ein Bedarf an automatischen Lösungen.

Ein möglicher Ansatz ist die worteinbettungsbasierte Anforderungsrückverfolgung: Hierbei werden aus Anforderungen und Quelltextteilen Wörter extrahiert, welche anschließend in einen Vektorraum eingebettet werden. Beabsichtigt ist, dass Vektoren semantisch verwandter Begriffe eine geringe Distanz zueinander haben. Über Vektordistanzen sollen semantisch verwandte Wörter aus verschiedenen Artefakttypen einander zugeordnet werden können, um die Herstellung von Rückverfolgbarkeitsverbindungen zwischen den Artefakten zu ermöglichen.

Diese Arbeit ist Teil des Projekts INDIRECT [Hey19], das zum Ziel hat, die Rückverfolgbarkeit von Anforderungen im Quelltext über das Verstehen natürlicher Sprache und Programmanalyse zu ermöglichen. Als letzte Stufe des Projekts wird ein solcher worteinbettungsbasierter Ansatz angewendet, um die Rückverfolgbarkeitsverbindungen herzustellen. Die Worteinbettungen werden dabei aus Begriffen, die in den Anforderungen und im Quelltext vorkommen, gebildet.

Betrachtet man nun den beispielhaften Anforderungssatz *„For each request the results are then displayed“*, so ergibt sich die Wortmenge { *request*, *results*, *displayed* }. Eine zugehörige Methode mit der Signatur `RequestManager:displayResult()` enthält analog die Begriffe { `request`, `display`, `result` }. Die Einbettungsvektoren dieser Wortmengen liegen paarweise sehr nah beieinander, da die Wörter gleich bzw. semantisch ähnlich sind. Damit ist eine korrekte Zuordnung einfach. Falls allerdings durch Refaktorierung die Methode in `RequestManager:showResultWindow()` umbenannt wird und eine neue Methode `RequestManager:clearResultDisplay()` hinzugefügt wird, so ergeben sich die Wortmengen { `request`, `show`, `result`, `window` } und { `request`, `clear`, `result`, `display` }. Zwar liegen die Vektoren der beiden Wortmengen aufgrund der semantischen Verwandtschaft nahe bei den Vektoren der Anforderungs-Wortmenge, aber der Vektor für `window` liegt nun weiter entfernt vom *„displayed“*-Vektor der Anforderung als der `display`-Vektor der hinzugefügten „clear“-Methode, weshalb die Gefahr besteht, dass fälschlicherweise aufgrund geringer Vektordistanzen die „clear“-Methode der Anforderung zugeordnet wird.

Abhilfe schaffen kann das Anreichern von Begriffen im Quelltext mit Wissen: Sofern bekannt ist, dass der Begriff `window` im Methodenbezeichner `showResultWindow` dieselbe Bedeutung wie *„display“* trägt, kann die Wortmenge der Methode mit dem Begriff *„display“* angereichert werden.

Dann enthält die Wortmenge die Begriffe { `request`, `show`, `result`, `window`, `display` }, wodurch die Distanz zwischen dem `display`-Vektor der Methode und dem *„displayed“*-Vektor der Anforderung wieder gering genug ist, um die Methode korrekt zuzuordnen.

Semantisch verwandte Begriffe wie Synonyme oder Oberbegriffe können hierbei als Wissen infrage kommen. Damit eine solche Anreicherung möglich wird, müssen zunächst die Bedeutungen der Begriffe im Quelltext aufgelöst werden. Beispielsweise muss im obigen Beispiel erkannt werden, dass `window` ein Anwendungssteuerelement meint und nicht ein Gebäudefenster, da sonst Begriffe wie *„glass“* oder *„house“* angereichert werden. Dafür muss der Kontext, in welchem die Methode sich befindet, betrachtet werden: In der Klasse `RequestManager` befindet sich die Methode `clearResultDisplay` mit dem Begriff `display`, was für `window` die Bedeutung des Steuerelements nahelegt.

1.1. Zielsetzung

Ziel dieser Arbeit ist es, zu untersuchen, wie Wissensanreicherung für Begriffe im Quelltext umgesetzt werden kann, um worteinbettungsbasierte Anforderungsrückverfolgung zu verbessern. Dafür muss untersucht werden, wie Bedeutungsauflösung von Begriffen im Quelltext umgesetzt werden kann. Anschließend sollen verschiedene Strategien zur Anreicherung von Wissen untersucht werden. Dabei soll analysiert werden, wie sich die Anreicherung auf die Ergebnisse der Anforderungsrückverfolgung auswirkt.

1.2. Aufbau der Arbeit

In Kapitel 2 werden Grundbegriffe, die für das Verständnis der Arbeit relevant sind, definiert und erklärt. Kapitel 3 gibt einen Überblick auf das Projekt INDIRECT, dessen Teil diese Arbeit ist. Einen Überblick über verwandte Arbeiten und ähnliche Ansätze gibt es in Kapitel 4. In Kapitel 5 wird die Problemstellung der Arbeit genauer analysiert und die konkrete Zielsetzung der Arbeit herausgearbeitet. Anschließend wird in Kapitel 6 ein Lösungsentwurf vorgestellt und in Kapitel 7 dessen Umsetzung beschrieben. In Kapitel 8 wird untersucht, inwieweit die Ziele dieser Arbeit eingehalten werden konnten. Kapitel 9 fasst die Arbeit noch einmal zusammen und bietet Ansatzpunkte für weitergehende Arbeiten.

2. Grundlagen

In diesem Kapitel soll eine Übersicht über grundlegende Begriffe und Verfahren gegeben werden, welche für die Erarbeitung der Ziele dieser Arbeit nötig sind. Zunächst werden in Abschnitt 2.1 notwendige Schritte und Begriffe zur Vorverarbeitung natürlichsprachlicher Texte erklärt. Anschließend werden in Abschnitt 2.2 und 2.3 Grundlagen zur Wortbedeutungsauflösung und Quelltextanalyse erläutert. Zu guter Letzt werden in Abschnitt 2.4 und 2.5 die Themen Wissensanreicherung und Anforderungsrückverfolgung vorgestellt.

2.1. Verarbeitung natürlicher Sprache

Quelltexte in Programmiersprachen wie Java weisen viele Vorkommen von natürlicher Sprache auf. Einerseits werden z. B. Kommentare im Quelltext oft in der Verkehrssprache der Entwickelnden verfasst, andererseits enthalten viele selbstgewählte Bezeichner wie Variablen-, Klassen- und Methodennamen natürlichsprachliche Begriffe bzw. sind aus ihnen zusammengesetzt. Für eine Analyse von Quelltext im Sinne der Zielsetzung dieser Arbeit ist also die Verarbeitung natürlicher Sprache relevant.

Damit natürliche Sprache maschinell weiterverarbeitet werden kann, müssen einige Vorverarbeitungsschritte durchgeführt werden. Diese Schritte dienen dem Verwerfen von Informationen aus dem zu verarbeitenden natürlichsprachlichen Text, die für die Zielsetzung der Arbeit irrelevant sind, und ebenso dem Beibehalten und der Extraktion von Informationen, die für die Zielsetzung der Arbeit relevant sind. Im Folgenden sollen diese Schritte vorgestellt werden.

2.1.1. Tokenisierung

Um die Bedeutung einzelner Begriffe auflösen zu können, müssen Eingabetexte zunächst in einzelne Begriffe aufgespalten werden. Jurafsky und Martin definieren Tokenisierung (engl. *tokenization*) als das Aufspalten von Text in Textsegmente (engl. *tokens*). Diese Textsegmente stellen die kleinsten Verarbeitungseinheiten dar und können nicht weiter aufgetrennt werden, ohne dass die ermittelte Bedeutung der Segmente sich von der beabsichtigten Bedeutung weiter entfernt. Tokenisierung trennt üblicherweise einzelne sinngebende Satzteile voneinander, was zumeist auf eine Wort-für-Wort-Trennung hinausläuft. Beispiel 2.1 zeigt eine mögliche Tokenisierung eines englischen Satzes.

Beispiel 2.1: Tokenisierung eines englischen Satzes (nach [JM21])

Für die Lesbarkeit sind Leerzeichen als ␣ dargestellt.
Eingabe:
"The␣San␣Francisco-based␣restaurant,"␣they␣said,␣"doesn't␣charge␣$10.50".
Ausgabe (einzelne Segmente werden eingerahmt wiedergegeben):
" | The | San␣Francisco-based | restaurant | , | " | they | said | , | " | does | n't | charge | $ | 10.50 | " | .

Im Beispiel würde eine weitere Auftrennung des Segments San␣Francisco-based in San | Francisco | - | based dazu führen, dass nicht die beabsichtigte Stadt San Francisco ermittelt wird und ferner der Zusammenhang zwischen „San Francisco" und „based" verlorengeht. Genauso kann das Segment 10.50 nicht am Dezimaltrenner in die Segmente 10 und 50 aufgeteilt werden. Allerdings ist es sinnvoll, dass Zusammenziehungen wie „doesn't" in die Segmente does und n't aufgeteilt werden, damit aus „does" die Grundform „do" abgeleitet werden kann (siehe Abschnitt 2.1.3), was wiederum für anschließende Arbeitsschritte hilfreich ist.

2.1.2. Stoppwort-Entfernung

In natürlichsprachlichen Texten kommen manche Wörter besonders häufig vor, die wenig zur Bedeutung des restlichen Textes beitragen, wie zum Beispiel „the", „a", „and" oder „of". Diese Wörter werden Stoppwörter genannt. Je nach Anwendungsfall werden diese Wörter in der maschinellen Sprachverarbeitung nach der Tokenisierung herausgefiltert, um semantisch gewichtigere Wörter wegen ihrer geringeren Häufigkeit bei der Sprachverarbeitung nicht zu benachteiligen. Dies geschieht üblicherweise über vorgefertigte Listen, sogenannte Stopplisten. [JM21]

2.1.3. Lemmatisierung

Für die Auflösung der Bedeutungen einzelner Wörter ist es notwendig, Unterschiede zwischen verschiedenen Formen derselben Wörter zu ignorieren, um die sonst nötige Behandlung von diesbezüglichen Redundanzen zu vermeiden. Dafür werden Wörter ihren Stammformen zugeordnet.

Die Grund- bzw. Stammform eines Wortes wird Lemma dieses Wortes genannt. Lemmatisierung bezeichnet somit das Finden der Stammformen einzelner Wörter. Das Lemma der englischen Wörter *am*, *is* und *are* ist *be*, das Lemma von *dinner* und *dinners* ist *dinner*. Die Lemmatisierung des Satzes *He is reading detective stories* ist also *He be read detective story*. Für eine korrekte Lemmatisierung sind allerdings komplexe morphologische Analyseverfahren notwendig. Zur Vereinfachung wird daher oft das sogenannte Stemming gewählt, eine naive Variante der morphologischen Analyse, bei der Wortsuffixe vom Wortstamm abgeschnitten werden. Beispiel 2.2 zeigt hierzu die Anwendung des sog. Porter-Stemmers. [JM21]

Beispiel 2.2: Lemmatisierung mit dem Porter-Stemmer [JM21]

Eingabe:
This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.

Ausgabe:
Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note

2.1.4. Wortarten und Wortarterkennung

Durch Klassifikation nach grammatischen Merkmalen erhalten Wörter eine Einteilung in Kategorien wie Substantiv, Verb, Adjektiv oder Präposition. Die zu einem Wort gehörende Kategorie wird auch als Wortart dieses Wortes bezeichnet (engl. *part of speech, POS*) [Buß02]. Je nach Wortart können Wörter unterschiedliche Bedeutungen tragen wie bspw. das Substantiv „fly“ (dt. *Fliege*) und das Verb „(to) fly“ (dt. *fliegen*). Um die Bedeutung eines Wortes zu bestimmen, muss also der Eingabetext analysiert und eine Wortarterkennung (engl. *part-of-speech tagging*) durchgeführt werden. Dabei ist Wortarterkennung selbst bereits eine Bedeutungsauflösungs-Aufgabe. State-of-the-Art-Algorithmen erreichen 97 % Genauigkeit in Bezug auf die korrekt bestimmten Wortarten. [JM21]

Beispiel 2.3: Nach Wortarterkennung annotierter Beispielsatz

Eingabe:
Thumper's fur is the fluffiest.

Für die Annotation der Wortarten wird die folgende Tabelle verwendet.

Wortartkürzel	Beschreibung
DT	Artikelwort
JJS	Adjektiv (Superlativ)
NN	Substantiv (Singular)
NNP	Eigenname (Singular)
POS	Genitivendung (z. B. „'s“)
VBZ	Verb (3. Pers. Singular)

Abbildung 2.1.: Auszug aus der Penn-Treebank-Tabelle der Wortartkürzel [JM21]

Somit ergibt sich folgende Wortart-Annotation.
Ausgabe:
Thumper/NNP 's/POS fur/NN is/VBZ the/DT fluffiest/JJS.

2.1.5. Semantische Relationen

In diesem Abschnitt sollen einige Begriffe erläutert werden, mit welchen in der Linguistik Beziehungen zwischen Wörtern bzw. Wortbedeutungen benannt werden.

Synonyme sind Wörter, die gleiche oder fast gleiche Bedeutungen haben. Ein Synonym von „couch“ ist bspw. „sofa“. **Antonyme** sind Wörter, die zueinander gegensätzliche Bedeutungen haben. Beispiele für Antonym-Paare sind „up“/„down“ und „long“/„short“. Ein **Hyperonym** eines Wortes ist ein Oberbegriff dieses Wortes (engl. *hypernym* (sic!), *superordinate*). Somit ist „building“ ein Hyperonym von „house“. Hingegen ist ein **Hyponym** eines Wortes ein Unterbegriff dieses Wortes (engl. *hyponym, subordinate*). Das Wort „dog“ ist Hyponym von „animal“. **Meronymie** beschreibt eine Teil-von-Beziehung. Demnach ist „wheel“ ein Meronym von „car“, ebenso ist „leg“ ein Meronym von „chair“. Die Umkehrung dieser Relation ist die **Holonymie**. Beispielsweise ist „car“ ein Holonym von „wheel“. [JM21]

Homonyme sind mehrdeutige Wörter, d. h., Wörter, die gleich geschrieben werden, aber unterschiedliche Bedeutungen haben, welche nicht miteinander verbunden sind. Das Wort „Bank“ im Sinne des Kreditinstitutes und das Wort „Bank“ im Sinne der Sitzgelegenheit sind Homonyme. Davon abzugrenzen sind **Polyseme**, welche zwar ebenso gleich geschrieben werden, aber unterschiedliche Bedeutungen haben, welche miteinander verwandt sind. Somit sind das Wort

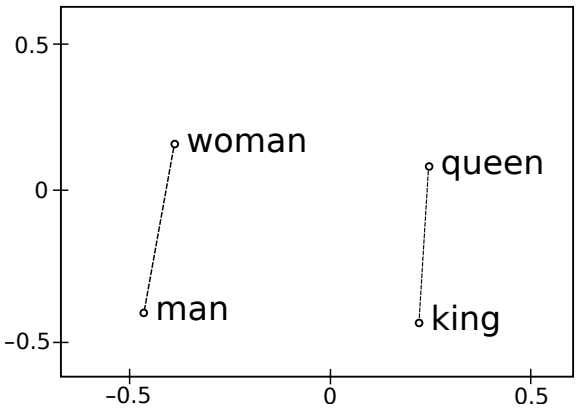


Abbildung 2.2.: Zweidimensionale Projektion einer Beispieleinbettung (nach [JM21])

„Bank“ im Sinne des Kreditinstitutes und das Wort „Bank“ im Sinne des Kreditinstitut-Gebäudes Polyseme. Zum Beispiel ist im Satz *„Biegen Sie einfach bei der Bank rechts ab“* mit „Bank“ das Gebäude gemeint. [Buß02]

2.1.6. Repräsentation von Wortbedeutungen

In Bußmanns *Lexikon der Sprachwissenschaft* werden verschiedene Möglichkeiten aufgezählt, den Begriff der Wortbedeutung zu definieren. Nach der für diese Arbeit passendsten Definition ist die Bedeutung eines sprachlichen Ausdrucks allein durch seine Stellung und Position im Sprachsystem festgelegt. Anders ausgedrückt ist die Bedeutung eines Wortes durch seine Beziehungen zu anderen Wörtern bereits eindeutig gegeben. Beispielsweise ist die Bedeutung des englischen Wortes „mouse“, das „mammal“ als Oberbegriff hat, eine andere als die des englischen Wortes „mouse“, das Unterbegriff von „input device“ ist. Offensichtlich erschließt sich somit die Bedeutung eines Wortes innerhalb eines Textes nur durch den Kontext, in welchem das Wort verwendet wird. Im Textbeispiel *„Other than a rabbit, a mouse usually grows a hairless tail“* wird das Wort „mouse“ im Kontext des Wortes „rabbit“ benutzt, wobei beide jeweils „mammal“ als Oberbegriff haben, was nahelegt, dass hier die zuvor erstgenannte Bedeutung von „mouse“ gemeint ist.

Lexika geben die unterschiedlichen Bedeutungen einzelner Wörter üblicherweise als Beschreibungen unter Verwendung derjenigen Wörter an, mit denen das zu definierende Wort in Verbindung steht. Im Folgenden werden zwei Arten betrachtet, wie Wortbedeutungen in der Informatik repräsentiert werden können, um eine weitere Verarbeitung zu ermöglichen. Die erste Möglichkeit verwendet reellwertige Vektoren als Repräsentation, was besonders im Fachbereich des Maschinellen Lernens als beliebtes Datenmodell gilt. Die zweite Möglichkeit versucht, die obige Definition von Wortbedeutungen in Datenstrukturen zu übersetzen.

Worteinbettungen

Worteinbettungen sind Abbildungen von Wörtern in einen reellen Vektorraum. Unterschieden werden hierbei statische Einbettungen, welche jedem Wort in einem Vokabular einen festen Vektor zuordnen, von kontextuellen Einbettungen, mit welchen jedes Wort je nach auftretendem Kontext entsprechend unterschiedliche Vektoren zugeordnet bekommt. Um verschiedene Bedeutungen einzelner Wörter zu repräsentieren, eignen sich hiernach nur kontextuelle Einbettungen, welche auch Bedeutungseinbettungen (engl. *sense embeddings*) genannt werden, da statische Einbettungen nur eine Repräsentation pro Wort erlauben.

Ein naiver Ansatz für Worteinbettungen heißt Bag-of-Words und ordnet jeder Wortbedeutung eine eigene Vektorraumdimension zu. Dies führt allerdings zu überwiegend dünnbesetzten Vektoren[1],

[1] Vektoren mit mehrheitlich Null-Einträgen

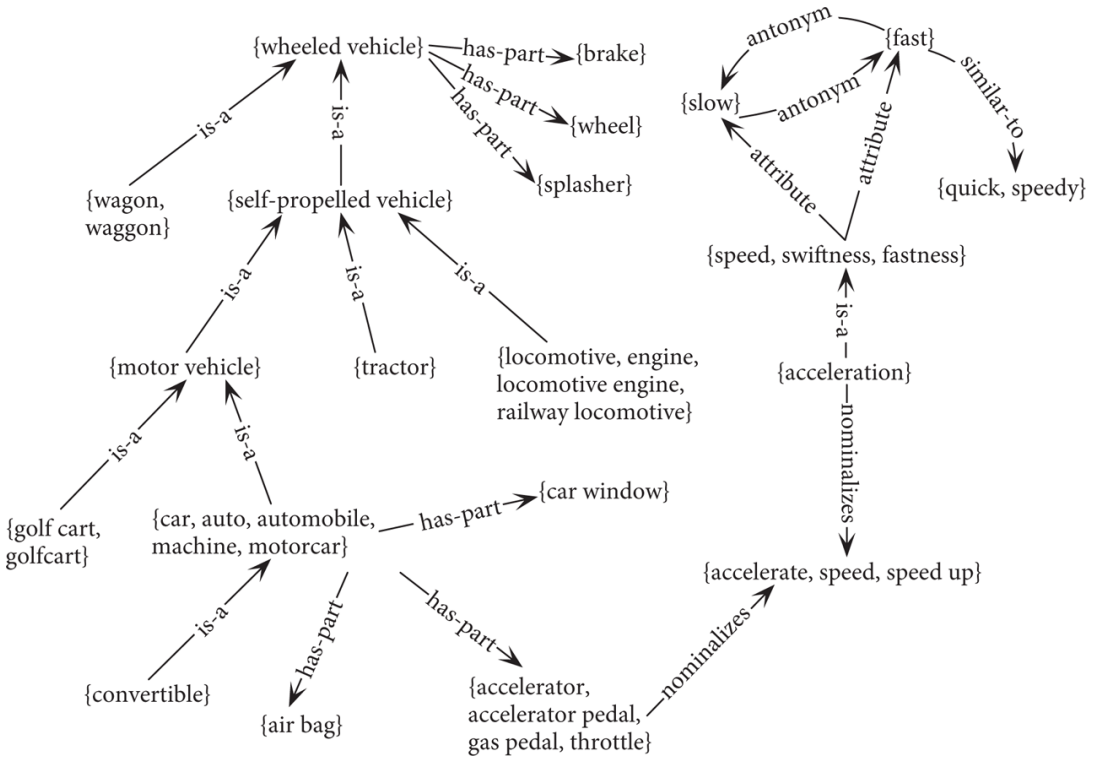


Abbildung 2.3.: Ausschnitt aus dem WordNet-Graphen [JM21]

zwischen denen kaum Abhängigkeit besteht, was der tatsächlichen semantischen Verknüpftheit von Wortbedeutungen nicht gerecht wird. Daher wird die Dimension eines solchen Vektorraums üblicherweise deutlich kleiner als die Anzahl verschiedener Wortbedeutungen gewählt, sodass eher dichtbesetzte Vektoren für die Einbettung genutzt werden, was auch einer Anwendung im Bereich des Maschinellen Lernens zugutekommt. Zudem sollen semantisch verwandte Wörter auch als Vektoren im Vektorraum eine geringe Distanz zueinander haben. Bei entsprechender Konstruktion lassen sich mathematische Eigenschaften nutzen, wie in Abbildung 2.2 zu sehen ist: Die Beziehungen zwischen Wörtern mit der Eigenschaft „männlich“ und Wörtern mit der Eigenschaft „weiblich“ wurden als nahezu konstante Vektortranslation repräsentiert. Somit liegt der Vektor $\left(\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}}\right)$ nahe beim Vektor $\overrightarrow{\text{queen}}$, was semantisch insofern nachvollziehbar ist, als dass beim Begriff „king“ das Subtrahieren der Männlichkeitseigenschaft und Addieren der Weiblichkeitseigenschaft tatsächlich zum Begriff „queen“ führt. [JM21]

Wissensnetz

Ein Wissensnetz ist ein Graph, der Wortbedeutungen als Knoten und semantische Beziehungen zwischen Wörtern als Kanten darstellt. Verschiedene Bedeutungen eines einzelnen Wortes erhalten jeweils einen eigenen Knoten. Die Kanten eines Wissensnetzes können Sinnverwandtschaft (z. B. engl. *couch/sofa*), Gegensätzlichkeit (z. B. engl. *long/short, up/down*) oder auch taxonomische Beziehungen (*furniture* ist ein Oberbegriff von *chair*, *mango* ist ein Unterbegriff von *fruit*) anzeigen. Weitere Informationen wie z. B. textuelle Beschreibungen der Bedeutung (auch Glosse genannt) oder Beispielsätze können den Knoten zusätzlich zugeordnet werden. [JM21]

Ein solches Wissensnetz ist WordNet mit Einträgen für über 150 000 Wörter in Version 3.0. Wörter werden hier über ihr Lemma repräsentiert, um Redundanzen zu vermeiden. Als Repräsentation von Wortbedeutungen werden in WordNet sogenannte Synsets (engl. *synonym sets*, dt. *Synonymmengen*) verwendet, welche mehrere bedeutungsgleiche Lemma-Einträge zu einer Einheit zusammenschließen. Die semantischen Relationen sind somit zumeist zwischen Synsets ausgedrückt. Abbildung 2.3 zeigt einen Ausschnitt aus dem WordNet-Graphen. Es lassen sich

Beziehungen ablesen, z. B. sind die Synsets {fast} und {quick, speedy} zueinander „similar“ (dt. *ähnlich*) oder die Synsets {slow} und {fast} zueinander „antonym“ (gegensätzlich). [JM21]

Das Projekt DBpedia hat zum Ziel, Wissensnetze automatisiert aus Wikipedia zu extrahieren [Leh+15]. Das Thema eines einzelnen Wikipedia-Artikels ergibt hierbei einen Bedeutungsknoten im Wissensnetz, da ein einzelner Wikipedia-Artikel üblicherweise ein Konzept bzw. eine Thematik behandelt und auch verschiedene Bedeutungen eines Begriffs üblicherweise jeweils eigene Artikel erhalten. Zudem werden Wikipedia-Artikelinhalte sowie Artikel-Metadaten wie Verlinkungen zwischen Artikeln oder Artikelkategorien miteinbezogen, um semantische Beziehungen zwischen den Artikelthemen als Kanten im Wissensnetz zu erfassen. Diese Beziehungen werden in Form von sogenannten RDF[2]-Tripeln angegeben. RDF-Tripel bestehen aus den Bestandteilen Subjekt, Prädikat und Objekt und stellen so einfache faktische Aussagen dar [JM21]. Beispiel 2.4 zeigt eine einfache Ortsangabe und die zugehörige natürlichsprachliche Aussage.

Beispiel 2.4: Beispiel für ein RDF-Tripel (nach [JM21])

Subjekt	**Prädikat**	**Objekt**
Golden Gate Park	located-in	San Francisco

Dieses Tripel repräsentiert den englischen Satz *„Golden Gate Park is located in San Francisco“*.

Es ist zu beachten, dass Wikipedia-Artikel üblicherweise mit Substantiven oder Eigennamen betitelt sind [Wik], sodass DBpedia-Wissensnetze im Gegensatz zu WordNet hauptsächlich Einträge für Substantive oder Eigennamen enthalten.

2.2. Wortbedeutungsauflösung

Als Wortbedeutungsauflösung (engl. *word sense disambiguation*, im Folgenden auch *WSD*) wird die Aufgabe bezeichnet, Wörtern anhand des Kontextes, in welchem sie auftreten, ihre korrekte Bedeutung zuzuordnen. Was hierbei konkret als Repräsentation für Wortbedeutungen verwendet wird, hängt vom verwendeten Ansatz ab (siehe Abschnitt 2.1.6). Je nach Anwendungsfall unterscheidet man bei WSD zwei Aufgabentypen: Beim sogenannten **lexical sample task** wird lediglich eine Vorauswahl an Wörtern aufgelöst, wohingegen beim **all-words task** alle Wörter im Eingabetext aufgelöst werden sollen. Verschiedene Lösungsansätze sollen im Folgenden vorgestellt werden. [JM21]

2.2.1. Lernbasierte Wortbedeutungsauflösung

Lernbasierte WSD-Verfahren basieren auf Maschinellem Lernen, welches zum Ziel hat, menschliche Lernprozesse maschinell nachzubilden. Dabei werden üblicherweise kontextuelle Worteinbettungen als Repräsentation benutzt. Man unterscheidet sogenannte überwachte von unüberwachten Lernverfahren.

Überwachte Wortbedeutungsauflösung

Überwachte Bedeutungsauflösung (engl. *supervised word sense disambiguation*) nutzt ein vorheriges Training mit annotierten Beispieltexten, um die korrekte Zuordnung von Wörtern zu ihrer jeweiligen Bedeutung zu erlernen. Dies erfordert die Bereitstellung bzw. Aufbereitung von ausreichend Trainingsmaterial, das die Domäne der künftigen Eingabetexte abdeckt.

Beispiel 2.5 zeigt eine WordNet-kompatible Annotation eines Satzes. An den verschiedenen Einträgen für „fruit“ ist ersichtlich, dass nur die Annotation „$\text{fruit}^{1}_{\text{n}}$“ die Bedeutung im Satzkontext

[2] Resource Description Framework

Beispiel 2.5: Beispiel für einen annotierten Text (nach [JM21])

Der folgende Satz ist aus dem SemCor-Korpus entnommen und zeigt hinter ausgewählten Wörtern die Bedeutungsannotation.

„You will find$^{9}_{v}$ that avocado$^{1}_{n}$ is$^{1}_{v}$ unlike$^{1}_{j}$ other$^{1}_{j}$ fruit$^{1}_{n}$ you have ever$^{1}_{r}$ tasted$^{2}_{v}$."

Die hochgestellten Ziffern zeigen die WordNet-Bedeutungsnummerierung an, die tiefgestellten Buchstaben die Wortart. Für „fruit" finden sich in WordNet u. a. beispielsweise folgende Einträge.

Eintrag	Glosse
fruit$^{1}_{n}$	the ripened reproductive body of a seed plant
fruit$^{2}_{n}$	yield; an amount of a product
fruit$^{3}_{n}$	the consequence of some effort or action

korrekt widerspiegelt. Das Training eines überwachten Bedeutungsauflösers mit diesen und ähnlichen Beispielsätzen hat zum Ziel, dass in künftigen Eingabetexten Vorkommen von „fruit" im Kontext mit „avocado" und/oder „(to) taste" genauso annotiert werden. [JM21]

Unüberwachte Wortbedeutungsauflösung

Unüberwachte Bedeutungsauflösung (engl. *unsupervised word sense disambiguation,* auch *word sense induction*) verzichtet auf Training mittels annotierter Textkorpora. Stattdessen werden für die im Eingabetext vorkommenden Wörter ihre verschiedenen Bedeutungen automatisch aus dem Kontext ermittelt und dabei die entsprechenden Zuordnungen gelernt. Üblicherweise werden hierfür kontextuelle Worteinbettungen anhand einer Distanzmetrik in Cluster eingeteilt, deren Mittelpunkte die unterschiedlichen Bedeutungsvektoren bilden. Diese ermittelten Bedeutungsrepräsentationen sind verfahrensbedingt namenlos und werden daher über eine Nummerierung identifiziert. Gegenüber überwachter Wortbedeutungsauflösung haben diese Ansätze den Vorteil, dass die aufwendige Annotation domänenspezifischer Trainingstexte wegfällt. [JM21]

2.2.2. Wissensbasierte Wortbedeutungsauflösung

Wissensbasierte Bedeutungsauflösung (engl. *knowledge-based word sense disambiguation*) verwendet Wissensnetze als Datenquelle, benötigt allerdings keine annotierten Textkorpora und wird auch nicht im Vorfeld trainiert. Stattdessen werden für ein gegebenes Wort aus dem Eingabetext die verschiedenen zugehörigen Bedeutungen im Wissensnetz untersucht und unter Betrachtung des Wortkontextes die passendste Bedeutung anhand ihrer Nachbarknoten im Wissensnetz oder ihrer Begleittexte ermittelt. [JM21]

Ein möglicher Ansatz, welcher auch Simplified-Lesk-Algorithmus genannt wird, berechnet beispielsweise für ein einzelnes Wort eines Satzes, wie viele Wörter aus dessen unmittelbarem Kontext in Glossen und Beispielsätzen der passenden Bedeutungsknoten vorkommen. Derjenige Bedeutungsknoten, der hier die größte Überschneidung aufweist, wird dem Wort zugeordnet. [Les86]

Beispiel 2.6: Beispiel für wissensbasierte Bedeutungsauflösung (nach [JM21])

Im folgenden Satz soll das Wort „bank“ (fett) aufgelöst werden:

„The **bank** can guarantee that *deposits* will eventually cover future tuition costs because it invests in adjustable-rate *mortgage* securities.“

In WordNet gibt es für „bank“ unter anderem folgende Bedeutungseinträge. Bedeutungstragende Begriffe in Glossen und Beispielsätzen, welche im Kontext des Wortes vorkommen, sind kursiv markiert.

bank[1]	Glosse:	a financial institution that accepts *deposits* and channels the money into lending activities
	Beispiele:	„that bank holds the *mortgage* on my home“
bank[2]	Glosse:	sloping land (especially the slope beside a body of water)
	Beispiele:	„they pulled the canoe up on the bank“

Beispiel 2.6 zeigt das Vorgehen an einem Beispielsatz: Die Bedeutung bank[1] hat zwei Wörter, die im Kontext des Beispielsatzes auftauchen („deposits“ und „mortgage“), während bank[2] keine solche Überschneidung aufweist. Somit wird bank[1] als Ergebnis gewählt.

Zwar liefern überwachte Lernansätze bessere Ergebnisse, jedoch können wissensbasierte Ansätze in sämtlichen Domänen eingesetzt werden, für die Lexika oder Wissensnetze verfügbar sind, selbst wenn keine bedeutungsannotierten Textkorpora existieren. [JM21]

2.3. Quelltextanalyse

In diesem Abschnitt soll Quelltext als natürlichsprachliche Textquelle betrachtet werden. Dafür werden Entitäten im Quelltext beleuchtet, die diesbezüglich geeignet sind. Anschließend sollen einige Grundbegriffe der Quelltextanalyse eingeführt werden, die für die Ziele dieser Arbeit relevant sind. Die Begriffe beziehen sich gemäß dem Rahmen dieser Arbeit auf Java-Quelltexte.

2.3.1. Natürliche Sprache in Kommentaren

Ein Kommentar ist ein Quelltextabschnitt, der die Programmübersetzung und -ausführung nicht beeinflusst und gesondert als solcher gekennzeichnet ist. Kommentare enthalten üblicherweise natürlichsprachliche Textinformation, die Menschen beim Lesen, Verstehen und/oder Bearbeiten des Quelltextes unterstützen soll. Zeilenkommentare in Java werden mit der Zeichenfolge `//` eingeleitet und mit einem Zeilenende-Zeichen oder dem Dateiende beendet, Blockkommentare beginnen mit `/*` und enden mit `*/`. Davon abgesehen unterliegt der Kommentarinhalt keinerlei Einschränkungen. [Orac, Kapitel 3]

Eine Besonderheit stellen sogenannte Javadoc-Dokumentationskommentare dar, die einzelnen Quelltextelementen zugeordnet sind. Diese Blockkommentare folgen einem eigenen Schema und enthalten maschinenlesbar strukturierten natürlichsprachlichen Text zur Beschreibung des zugehörigen Quelltextelements. Dadurch kann aus solchen Javadoc-Kommentaren automatisch Quelltextdokumentation generiert werden. Beispiel 2.7 zeigt, wie mittels Javadoc-Kommentar eine Beschreibung des Methodenverhaltens, der Methodenparameter und des Methodenrückgabewertes angegeben werden kann. [Orab]

2.3.2. Natürliche Sprache in Bezeichnern

Quelltextentitäten wie Klassen, Methoden, Methodenparameter, lokale sowie Instanz- und Klassenvariablen werden im Quelltext über selbstgewählte Bezeichner (engl. *identifier*) identifiziert.

Beispiel 2.7: Javadoc-Kommentar zu einer Methode

```java
/**
 * Calculates logical implication between two booleans.
 *
 * @param first the first boolean
 * @param second the second boolean
 * @return true if "first implies second", false otherwise.
 */
private boolean implies(boolean first, boolean second) {
    return !first || second;
}
```

Diese Bezeichner enthalten oft natürlichsprachliche Begriffe, um die Bedeutung des benannten Quelltextelements an lesende oder bearbeitende Menschen zu kommunizieren.

Da in Bezeichnern keine Leerzeichen vorkommen dürfen, werden aus mehreren Wörtern zusammengesetzte Bezeichner z. B. mittels Binnenmajuskel-Schreibweise (engl. *camel case*) gebildet, bei welcher die ersten Buchstaben angehängter Wörter großgeschrieben werden (z. B. `writeToFile`). Alternativ können Wörter auch unverändert mit Unterstrichen als Trennzeichen aneinandergereiht werden (z. B. `write_to_file`). Dies soll in dieser Arbeit als *Binnenunterstrich-Schreibweise* bezeichnet werden.

Schlüsselwörter der Programmiersprache sind unverändert als Bezeichner nicht erlaubt, weshalb alternativ u. a. Abkürzungen (wie `dbl` statt `double`) oder abgewandelte Schreibungen (wie `clazz` statt `class`) gewählt werden. Vor allem lokale Variablen werden ebenfalls oft mit Abkürzungen (wie `id` statt `identifier`) oder gar einbuchstabig (wie `int i` oder `c` statt `counter`) benannt. All dies stellt die linguistische Verarbeitung von Bezeichnern in Quelltexten vor Herausforderungen. [Ull20]

2.3.3. Abstrakter Syntaxbaum

Ein abstrakter Syntaxbaum (engl. *abstract syntax tree*, im Folgenden auch *AST*) eines Quelltextes ist eine hierarchische Repräsentation der syntaktischen Struktur dieses Quelltextes. Der Quelltext wird beim Übersetzungsvorgang zerlegt und anhand der Grammatik der Quelltext-Programmiersprache analysiert. Der dabei entstandene AST wird anschließend für weitere Übersetzungsprozesse wie die Maschinencode-Erzeugung verwendet. [AA07]

In Beispiel 2.8 wird ein Quelltextabschnitt und ein zugehöriger AST gezeigt. Der `while`-Befehl enthält die Teilbäume von Ausführbedingung und Schleifenrumpf als Kinder. Diese sind wiederum zerlegt in einzelne Ausdrücke aus Operatoren und Operanden. Der $+$-Knoten enthält einen $*$-Teilbaum als Kind, was die Operatorrangfolge im Teilausdruck `a + 2 * i` widerspiegelt.

Ein AST eignet sich zur Extraktion von verschiedenen Abhängigkeitsinformationen, welche in den folgenden Abschnitten beschrieben werden.

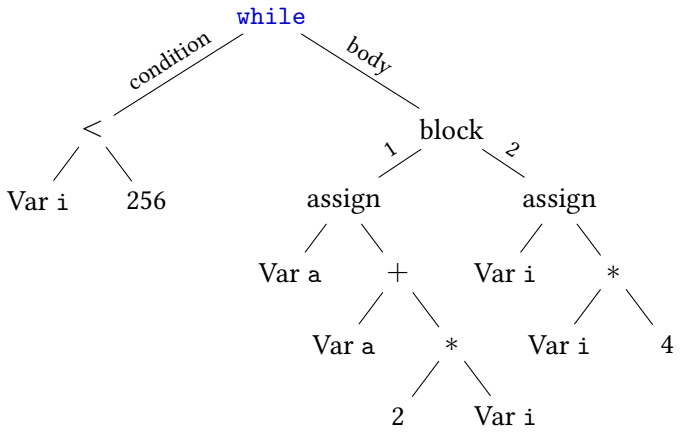
2.3.4. Methodenaufruf-Abhängigkeit

Eine Methode A ist von einer Methode B aufrufabhängig, wenn Methode A Methode B aufruft. Diese Relation ist gerichtet. Zwischen A und B besteht dann eine Methodenaufruf-Abhängigkeit (engl. *method call dependency*) oder Aufrufabhängigkeit (engl. *call dependency*).

Ein Methodenaufruf-Graph (engl. *method call graph*) oder Aufrufgraph (engl. *call graph*) eines Quelltextes ist ein gerichteter Graph mit der Menge aller Methoden im Quelltext als Knotenmenge und sämtlichen Aufrufabhängigkeiten zwischen den Methoden als Kantenmenge. [Kua+12]

Beispiel 2.8: Quelltext und zugehöriger AST

```
while (i < 256) {
    a = a + 2 * i;
    i = i * 4;
}
```



Beispiel 2.9: Quelltext mit Methodenaufrufen und Datenzugriffen

```
1  class A {
2      private int x;
3
4      public A() {
5          this.x = 42;
6          B b = new B(this.x);
7          b.m();
8      }
9  }
```

```
10 class B {
11     public B(int number) {
12         this.init();
13     }
14
15     private void init() { }
16
17     public void m() { }
18 }
```

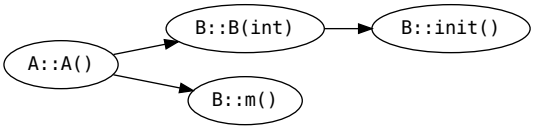


Abbildung 2.4.: Der zu Beispiel 2.9 gehörende Aufrufgraph.

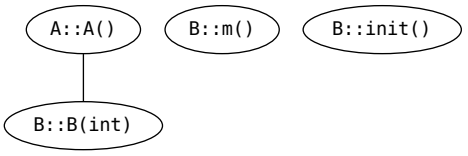


Abbildung 2.5.: Der zu Beispiel 2.9 gehörende Datenabhängigkeitsgraph.

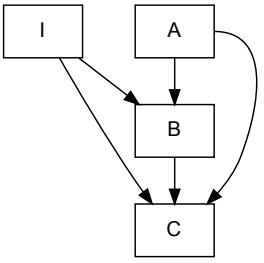
Beispiel 2.10: Deklaration abhängiger Referenztypen, Vererbungsgraph

```
interface I { }

class A { }

class B extends A implements I { }

class C extends B { }
```



Im Quelltext von Beispiel 2.9 ruft der Konstruktor von `A` den Konstruktor von `B` sowie die Methode `m` des `B`-Objektes auf. Der Konstruktor von `B` ruft zudem die Methode `init` der Klasse auf. Abbildung 2.4 zeigt diese Abhängigkeiten als Graph.

2.3.5. Aufrufgeschwister

Zwei verschiedene Methoden heißen Aufrufgeschwister (engl. *call siblings*), wenn beide innerhalb einer Methode aufgerufen werden. Es besteht dann eine sogenannte Aufrufgeschwisterschaft zwischen den Methoden.

2.3.6. Daten-Abhängigkeit

Zwei verschiedene Methoden sind datenabhängig, wenn beide Methoden auf dieselben Daten (lokale, Instanz- oder Klassenvariablen) zugreifen. Diese Relation ist ungerichtet. Zwischen zwei solchen Methoden besteht dann eine Datenabhängigkeit (engl. *data dependency*).

Ein Datenabhängigkeits-Graph (engl. *data dependency graph*) eines Quelltextes ist ein ungerichteter Graph mit der Menge aller Methoden im Quelltext als Knotenmenge und sämtlichen Datenabhängigkeiten zwischen den Methoden als Kantenmenge. [Kua+12]

Abbildung 2.5 zeigt den zu Beispiel 2.9 gehörigen Datenabhängigkeitsgraph. Es gibt im Quelltext genau drei Datenquellen: `A.x`, die lokale Variable `b` in `A::A()` und der Parameter `number` in `B::B(int)`. Auf `A.x` wird von `A::A()` und `B::B(int)` zugegriffen (Zeilen 5 und 6). Daraus ergibt sich eine Datenabhängigkeit zwischen `A::A()` und `B::B(int)`. Auf `b` wird nur in `A::A()` zugegriffen. Der Parameter `number` wird gar nicht benutzt.

2.3.7. Vererbungsbeziehungen

Eine Klasse A ist direkte Oberklasse (engl. *direct superclass*) einer Klasse B, wenn Klasse B mit einer Klausel `extends A` deklariert wurde. In diesem Fall ist B auch direkte Unterklasse (engl. *direct subclass*) von A.

Die transitive Hülle der Relation „direkte Oberklasse“ ist die Relation „Oberklasse“. Also sind alle direkten Oberklassen von A auch Oberklassen von A und falls A Oberklasse von B und B Oberklasse von C ist, so ist auch A Oberklasse von C. Analog verhält es sich mit der Relation „Unterklasse“. Somit ist B genau dann Unterklasse von A, wenn A Oberklasse von B ist. [Orac, Kapitel 8]

Ein Schnittstellentyp I ist direkte Oberschnittstelle (engl. *direct superinterface*) eines Schnittstellentyps J oder einer Klasse C, wenn J mit einer Klausel `extends I` [Orac, Kapitel 9] bzw. C mit einer Klausel `implements I` [Orac, Kapitel 8] deklariert wurde. Die transitive Hülle der Relation „direkte Oberschnittstelle“ ist die Relation „Oberschnittstelle“. Ist I eine Oberschnittstelle einer Schnittstelle J, so heiße J analog Unterschnittstelle von I.

Zusammengefasst heiße ein Typ T Obertyp / Untertyp (engl. *supertype / subtype*) eines Typs S, wenn T Oberklasse / Unterklasse oder Oberschnittstelle / Unterschnittstelle von S ist.

Eine Klasse C hängt direkt von einem Typ T ab (engl. *directly depends on*), wenn T direkte Oberklasse oder Oberschnittstelle von C ist. Die transitive Hülle der Relation „hängt direkt ab“ ist die Relation „hängt ab“ (engl. *depends on*). Also hängt C von T ab, falls C von T direkt abhängt oder falls C von B abhängt und B von T abhängt. [Orac, Kapitel 8]

Ein Vererbungsgraph eines Quelltextes ist ein gerichteter Graph mit der Menge aller Referenztypen im Quelltext als Knotenmenge, wobei für zwei Typen T_1 und T_2 die Kante (T_1, T_2) genau dann Kante im Graphen ist, wenn T_2 von T_1 abhängt.

Beispiel 2.10 zeigt die Deklaration einiger abhängiger Referenztypen und den dazugehörigen Vererbungsgraphen. Beachte z. B. die Kante von I nach C, da C von I aufgrund Transitivität über B abhängt.

2.4. Wissensanreicherung

In diesem Abschnitt sollen Konzepte vorgestellt werden, die für die Anreicherung von Wörtern in einer natürlichsprachlichen Eingabe mit zusätzlichem Wissen wie bspw. semantischen Relationen relevant sind.

2.4.1. Informationsgehalt

Der Informationsgehalt (engl. *information content*) oder Überraschungswert einer Aussage S ist definiert als

$$I(S) := -\log \mathbb{P}(S)\,,$$

wobei $\mathbb{P}(S)$ die Auftrittswahrscheinlichkeit der Aussage S bezeichnet [Lin98]. Dies deckt sich mit der Intuition, dass unwahrscheinliche Aussagen mehr Information enthalten als Aussagen, die häufiger auftreten. Beispielsweise wird mit der Aussage *„Klopfer ist mein Haustier“* deutlich weniger Information transportiert als mit der Aussage *„Klopfer ist mein Löwenkopf-Zwergkaninchen“*: Die Wahrscheinlichkeit für das Auftreten des Wortes „Haustier“ ist deutlich größer als die des Wortes „Löwenkopf-Zwergkaninchen“, da letzteres viel seltener vorkommt.

2.4.2. Semantische Ähnlichkeit

Die semantische Ähnlichkeit (engl. *semantic similarity*) zwischen zwei Aussagen oder Wörtern A und B wird als $\mathrm{sim}(A, B)$ notiert und ist ein Wert zwischen 0 und 1. Zwei Aussagen oder Wörter sind zueinander semantisch umso ähnlicher zueinander, je ähnlicher ihre Bedeutungen sind. Es gilt stets $\mathrm{sim}(A, A) = 1$. Ferner ist für zwei Aussagen A und B, die keinerlei semantische Gemeinsamkeit haben, $\mathrm{sim}(A, B) = 0$. Im Folgenden soll eine Definition für sim nach Lin [Lin98] erklärt werden.

Bezeichne $\mathrm{description}(A, B, ...)$ eine Aussage, die die Bedeutungen der Aussagen/Wörter A, B etc. beschreibt. Sei weiterhin $\mathrm{common}(A, B)$ eine Aussage, die die Gemeinsamkeiten zwischen den Aussagen/Wörtern A und B beschreibt. Beispielsweise ist für die Aussagen $A =$ *„x ist ein Apfel“* und $B =$ *„y ist eine Orange“* die Aussage $\mathrm{common}(A, B) =$ *„x ist eine Frucht und y ist eine Frucht“*. Die Aussage $\mathrm{common}(A, B)$ ist im Allgemeinen das Wiederholen der Gemeinsamkeiten C für jeweils A und B. Für $C =$ *„... ist eine Frucht“* ist demnach der Informationsgehalt der vorherigen Aussage

$$\begin{aligned} I(\mathrm{common}(A, B)) &= I(C \wedge C) \\ &= -\log(\mathbb{P}(C \wedge C)) \\ &= -\log(\mathbb{P}(C) \cdot \mathbb{P}(C)) \\ &= -2\log \mathbb{P}(C)\,. \end{aligned} \tag{2.1}$$

Lin definiert somit die Ähnlichkeit zwischen A und B wie folgt:

$$\begin{aligned} \mathrm{sim}_{\mathrm{Lin}}(A, B) &:= \frac{I\left(\mathrm{common}(A, B)\right)}{I\left(\mathrm{description}(A, B)\right)} \\ &\overset{2.1}{=} \frac{I(C \wedge C)}{I\left(\mathrm{description}(A, B)\right)} \\ &= \frac{\log \mathbb{P}(C \wedge C)}{\log \mathbb{P}(\mathrm{description}(A, B))} \\ &= \frac{\log(\mathbb{P}(C) \cdot \mathbb{P}(C))}{\log(\mathbb{P}(\mathrm{description}(A)) \cdot \mathbb{P}(\mathrm{description}(B)))} \\ &= \frac{2\log \mathbb{P}(C)}{\log \mathbb{P}(\mathrm{description}(A)) + \log \mathbb{P}(\mathrm{description}(B))} \end{aligned}$$

Dies lässt sich analog für Wissensnetze und Bedeutungsknoten formulieren. Der *Lowest common subsumer* (LCS) zweier Bedeutungen ist der spezifischste gemeinsame Oberbegriff dieser Bedeutungen. Somit gilt für zwei Bedeutungsknoten A und B sowie $C = \mathrm{LCS}(A, B)$:

$$\mathrm{sim}_{\mathrm{Lin}}(A, B) = \frac{2 \log \mathbb{P}(C)}{\log \mathbb{P}(A) + \log \mathbb{P}(B)}, \qquad \text{[Lin98]}$$

wobei hier $\mathbb{P}(A)$ die durchschnittliche Auftrittswahrscheinlichkeit der Wortbedeutung A bezeichnet.

2.5. Anforderungsrückverfolgung

Anforderungsrückverfolgung (engl. *requirements traceability*) kann laut IEEE definiert werden als „die Identifikation und Dokumentation [...] eines Zuordnungspfades von Anforderungen in der Anforderungshierarchie“ bzw. als „erkennbare Zuordnung zwischen einer Anforderung und zugehörigen [...] Implementierungen [...]“ [ISO10]. Damit ist gemeint, dass Zusammenhänge zwischen Anforderungen aus Anforderungsdokumenten und anderen Artefakttypen festgestellt und dokumentiert werden sollen. Als Implementierungsartefakte können z. B. Entwurfsdokumente sowie Produkt- und Testquelltexte verstanden werden. Beispielsweise sollte eine mögliche Pflichtenheft-Anforderung *„If the patient is unknown, the reception creates a new record“* im Produktquelltext der Methode `Reception::createRecord(Patient)` zugeordnet werden [Hey+21]. Die Zuordnung solcher Stellen in Artefakten wird auch *traceability link recovery* (kurz *TLR*) genannt. Die Rückverfolgung von Anforderungen ist vor allem im Bereich der Software-Wartung und -Qualitätssicherung von großer Bedeutung.

3. INDIRECT

In diesem Kapitel soll das Projekt INDIRECT vorgestellt werden, in welches die Ergebnisse dieser Arbeit eingebunden werden sollen.

In der Arbeit *„INDIRECT: Intent-Driven Requirements-to-Code Traceability“* [Hey19] präsentiert Hey einen Entwurf für ein System zur automatisierten Anforderungsrückverfolgung (engl. *traceability link recovery, TLR*). Hierbei werden sowohl für die Anforderungen als auch den Quelltext sog. Absichtsmodelle (engl. *intent models*) generiert und anschließend Zuordnungen zwischen diesen beiden Absichts-Graphen ermittelt.

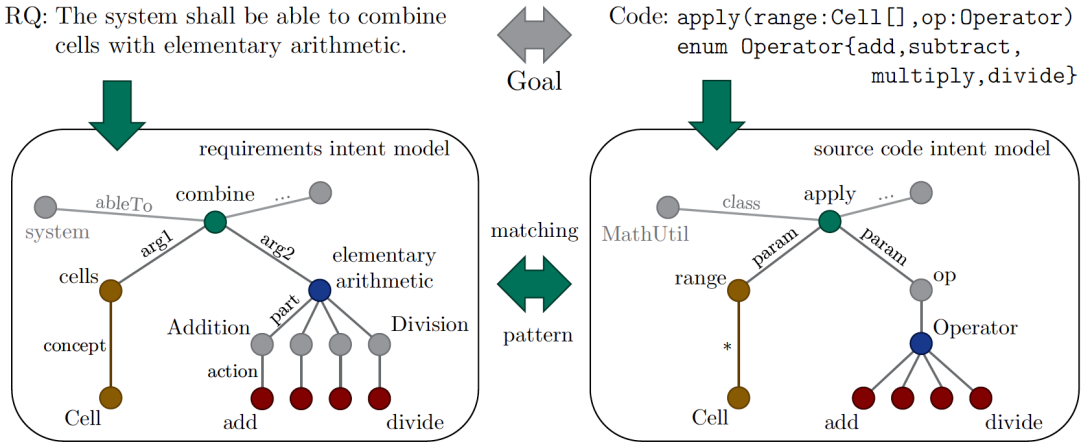


Abbildung 3.1.: Beispielhafte Abbildung der Absichtsmodelle von INDIRECT [Hey19]. Links eine Anforderung, rechts das entsprechende Quelltextelement, beide mit zugeh. Absichtsmodellen. Einander zugeordnete Knoten sind farblich markiert.

Diese Zuordnungen dienen dann als Basis für die TLR. Für die Generierung der Absichtsmodelle werden die Anforderungstexte und der Quelltext zerlegt und mithilfe unterschiedlicher Verfahren mit Wissen über zugrundeliegende Konzepte und mit semantischen Informationen angereichert. Ein wissensbasierter Ansatz soll hier im Vergleich zu alternativen Ansätzen eine höhere Trefferquote sowie größere Domänen-Flexibilität bieten. Abbildung 3.1 zeigt das Prinzip an einem Beispiel.

3.1. Projektarchitektur

Die Architektur von INDIRECT basiert auf einem Agenten-System, angelehnt an das PARSE-Projekt [WT15]. Eine Vorverarbeitungs-Pipeline aus mehreren Agenten baut dabei die initiale Graphenstruktur auf und erzeugt Knoten für einzelne Anforderungen sowie einen abstrakten Syntaxbaum für den Quelltext. Anschließend reichern die Agenten der Hauptverarbeitungs-Pipeline die jeweiligen Teilgraphen weiter an. Eine Nachverarbeitungs-Pipeline stellt anschließend die Zuordnungen zwischen den Teilgraphen her.

Ein Agent der Hauptverarbeitungs-Pipeline kann jeweils den Teilgraphen der Anforderungs- oder Quelltextseite manipulieren, beispielsweise semantische Beziehungen zwischen Knoten erfassen, Knoten mit semantischen Informationen anreichern oder mehrere Knoten zu einem übergeordneten Knoten zusammenfassen. Alle Agenten dieser Pipeline werden solange nacheinander eingesetzt, bis sie keine Arbeit mehr verrichten können.

Für die abschließende TLR kommt ein worteinbettungsbasiertes Verfahren zum Einsatz, das in der Arbeit *„Improving Traceability Link Recovery Using Fine-grained Requirements-to-Code Relations“* [Hey+21] beschrieben wird. Der mit dem Namen *„Fine-grained Traceability Link Recovery“* (kurz *FTLR*) vorgestellte Ansatz stellt für jedes zuzuordnende Artefaktelement sogenannte Bags-of-Embeddings zusammen. Diese Bags-of-Embeddings sind Mengen von statischen Worteinbettungen aller zu verarbeitenden Wörter des Artefakt-Elements. Solche Bags-of-Embeddings verschiedener Artefakttypen werden dann über die Word-Mover's-Distance-Ähnlichkeitsmetrik einander zugeordnet und damit Zuordnungen zwischen den Artefakten bestimmt. Die Agenten der Hauptverarbeitungs-Pipeline von INDIRECT haben zur Aufgabe, Absichtsmodelle aufzubauen, deren angereicherte Informationen dann für die Erstellung dieser Bags-of-Embeddings verwendet werden.

3.2. INDIRECT in dieser Arbeit

Für die Anforderungsseite wurden in bisherigen Arbeiten Ansätze zur Wortbedeutungsauflösung erarbeitet [HKT21]. Auf Quelltextseite fehlt dies bislang. Zwar wurden bereits Verfahren für semantische Quelltextanalyse erarbeitet wie z. B. semantisches Clustering von Quelltextteilen [Eur20] oder Auflösen von Abkürzungen im Quelltext [Gro21], jedoch wird auch für Quelltext eine Wortbedeutungsauflösung und weitere Wissensanreicherung benötigt, um wie auf Anforderungsseite potentielle Mehrdeutigkeiten zu eliminieren und die Zuordnung der Bags-of-Embeddings treffsicherer zu machen.

Somit sollen im Zuge dieser Arbeit Agenten für das INDIRECT-Projekt implementiert werden, welche Wortbedeutungsauflösung und Wissensanreicherung für Begriffe im Quelltext durchführen.

4. Verwandte Arbeiten

In diesem Kapitel sollen Arbeiten vorgestellt werden, die für diese Arbeit relevante Teilprobleme lösen oder interessante Ansätze vorstellen, die später bei der Erarbeitung eines Entwurfs dienlich sind.

Um Wissen zu Begriffen im Quelltext anzureichern, müssen zunächst die Bedeutungen der Begriffe korrekt ermittelt werden. Dafür werden in Abschnitt 4.1 Verfahren zur Wortbedeutungsauflösung vorgestellt. Wie bereits in Abschnitt 2.1.6 beschrieben, ergibt sich die Bedeutung eines Wortes nur aus dessen Kontext im Quelltext. Abschnitt 4.2 zeigt daher Ansätze, wie Informationen im Quelltext zur weiteren Analyse von Bedeutungen, Themenbereichen und Konzepten herangezogen werden können. Mit den korrekt ermittelten Bedeutungen kann nun zusätzliches Wissen angereichert werden. Abschnitt 4.3 stellt Ansätze vor, wie mithilfe eines Wissensnetzes zu bestehenden Bedeutungszuordnungen verwandte Bedeutungen wie Hyperonyme oder Meronyme ausgewählt werden können, um eine weitere semantische Verarbeitung zu ermöglichen.

4.1. Wortbedeutungsauflösung

Im Folgenden wird ein Überblick des Forschungsstands bezüglich WSD auf natürlichsprachlichen Eingaben gegeben. Dabei wird zwischen wissensbasierten, lernbasierten und Hybridverfahren unterschieden. Eine Übersicht der erreichten F_1-Werte ist in Tabelle 5.3 zu finden.

4.1.1. Wissensbasierte WSD

Die Arbeit *„Random Walks for Knowledge-Based Word Sense Disambiguation“* [ALS14] von Agirre et al. beschreibt, wie mithilfe von Personalized PageRank auf WordNet Wortbedeutungen aufgelöst werden können. Dabei wird WordNet als ungerichteter Graph betrachtet.

Der PageRank-Random-Walk-Algorithmus ordnet jedem Knoten ein Gewicht (engl. *rank*) zu, das davon abhängt, wie viele andere Knoten mit ihm verbunden sind und welches Gewicht jene Knoten selbst haben. Anschaulich kann das Gewicht eines Knotens i als die Wahrscheinlichkeit betrachtet werden, dass ein ausreichend langer Random Walk (dt. *Zufallsweg*) durch den Graphen bei i endet. In der ursprünglichen PageRank-Fassung von Brin et al. [BP98] wird ein Bias-Vektor definiert, der für jeden Knoten eine geringe Wahrscheinlichkeit angibt, mit der ein Random Walk zufällig auf diesen Knoten springt, ohne etwaige fehlende Kantenverbindungen zu berücksichtigen. In jener Arbeit wird für diese Wahrscheinlichkeiten eine Gleichverteilung angenommen, jeder Knoten kann somit gleich wahrscheinlich zufällig angesprungen werden. Agirre et al. nennen diese originale Variante des Algorithmus *Static PageRank* und unterscheiden sie vom sogenannten

Personalized PageRank, bei welchem ein modifizierter Bias-Vektor genutzt wird. Falls man bspw. einem Knoten i einen besonders hohen Bias-Wert zuweist, wird ein zufälliger Weg oft zu diesem Knoten zurückspringen. Die dann resultierenden PageRank-Gewichte aller Knoten können somit als die Relevanz dieser Knoten aus der Perspektive von i betrachtet werden.

Dieses Prinzip lässt sich nun auf WSD übertragen: Zu allen Wörtern einer aufzulösenden Textpassage, die Einträge in WordNet haben, werden deren Bedeutungsknoten mit gleichverteilt hohen Bias-Wahrscheinlichkeiten versehen. Alle anderen Knoten bekommen den Bias-Wert null. Die Anwendung von Personalized PageRank (im Folgenden auch PPR) wird die vergleichsweise hohen Gewichte der Bedeutungsknoten nach und nach auf deren Nachbarschaft verteilen. Der kontextuelle Zusammenhang der Eingabe wird dabei diejenigen Bedeutungsknoten der im Text vorkommenden Lemmata bevorzugen, die nah beieinander liegen. Am Ende wird von allen konkurrierenden Bedeutungen eines Wortes jeweils die gewichtsstärkste gewählt.

Eine Variante dieses Verfahrens ist *Personalized PageRank word-to-word* (kurz PPR_{w2w}), bei welcher eine Texteingabe nicht als Ganzes, sondern Wort für Wort aufgelöst wird. Hierbei werden wie oben die in der Eingabe vorkommenden Wörter mit Bias-Werten versehen, allerdings nicht das aktuell aufzulösende Wort. Dadurch werden die resultierenden Gewichtswerte der Bedeutungsknoten des Wortes ausschließlich durch den Kontext des Wortes bestimmt. PPR_{w2w} liefert bessere Ergebnisse, ist aber langsamer als PPR.

Für die Evaluation wurden die Aufgabensätze von SemEval-2013 [NJV13] und -2015 [MN15] verwendet. Dabei wurden F_1-Werte von 68.8 % bzw. 70.3 % durch die Verwendung von PPR_{w2w} erreicht [ALS18]. Die Autoren haben eine Implementierung ihres Ansatzes, die UKB-Werkzeugsammlung, veröffentlicht.

In der Arbeit *„Entity Linking meets Word Sense Disambiguation"* [MRN14] von Moro et al. wird das Projekt Babelfy vorgestellt. Als Wissensbasis wird BabelNet 1.1.1 verwendet, ein mehrsprachiges Wissensnetz, das aus einer automatischen Zusammenführung von Wikipedia und WordNet hervorgegangen ist. Die Autoren beschreiben ein dreischrittiges Verfahren zur Bedeutungsauflösung.

Zuerst bekommt jeder Knoten im Graph eine Menge an verwandten Knoten, die sogenannte semantische Signatur, zugeordnet. Diese wird über ein abgewandeltes Personalized-PageRank-Verfahren[1] bestimmt, bei welchem u. a. die Erreichbarkeit der Knoten durch die Anzahl an Dreieckszyklen, in denen sie vorkommen, gegeben ist. Als Zweites werden im Eingabetext alle aufzulösenden Fragmente bestimmt. Dabei werden auch Multiwort-Ausdrücke bis zu einer bestimmten Wortanzahl erlaubt, allerdings muss jedes Fragment ein Substantiv oder einen Eigennamen enthalten. Zudem erhält jedes Fragment die Menge seiner potentiellen Bedeutungskandidaten zugeordnet. Im dritten Schritt werden nun die Bedeutungen aufgelöst. Dafür wird ein Graph aufgebaut, in welchem fragmentübergreifend jeder Kandidat Kanten zu denjenigen Kandidaten bekommt, die auch in seiner semantischen Signatur vorkommen. Aus diesem Graphen wird mittels einer Heuristik ein dichter Subgraph ausgewählt und unter dessen Knoten über eine Gewichtung die passendste Bedeutung ermittelt.

Für die SemEval-2013-Aufgaben [NJV13] werden F_1-Werte von 69.2 % für BabelNet und 65.9 % für WordNet angegeben.

In der Arbeit *„Word Sense Disambiguation: A comprehensive knowledge exploitation framework"* [WWF20] kombinieren Wang et al. verschiedene Ansätze, zu welchen auch unüberwachte lernbasierte Schritte gehören, wobei jedoch alles auf Wissens- bzw. Textquellen aufbaut.

Es werden von Wikipedia und anderen Korpora basierend auf Wortvorkommen-Überlappung Textquellen abgerufen, die zu der Eingabe potentiell domänenverwandt sind. Auf diesen Textquellen werden dann Worteinbettungen mittels LSA (latent semantic analysis) erlernt. Die Wörter der vorverarbeitete Eingabe werden mit Glossen aus WordNet-Synset-Kandidaten und solchen

[1] dort als „Random Walk with Restart" bezeichnet

aus verwandten Synsets (Hyperonymen etc.) angereichert. Für diese erweiterten Glossen werden mit dem vorher erlernten Modell Worteinbettungen errechnet und zwischen diesen und den Vektoren aus den Textquellen die Ähnlichkeit errechnet. Aus den drei Glossen mit den höchsten Ähnlichkeitswerten wird anschließend ein Subgraph von WordNet gebildet basierend auf den Glossenwörtern und den zugehörigen WordNet-Relationen. Mittels PageRank wird dann die Signifikanz der Knoten ermittelt, welche dann zum Anpassen der obigen Ähnlichkeitswerte benutzt wird. Zuletzt werden über diese Werte dann die jeweils ähnlichsten Wortbedeutungen bestimmt.

Dieses Vorgehen liefert in der Evaluation die F_1-Werte 72.3 % für die SemEval-2015- [MN15] und 68.4 % für die SemEval-2013-Aufgaben [NJV13].

4.1.2. Lernbasierte WSD

Die Arbeit *„Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations“* [HNG19] von Hadiwinoto et al. zeigt die Verwendung von BERT [Dev+19] als transformerbasiertes Sprachmodell für kontextuelle Worteinbettungen, welches auf Texten mit mehreren Milliarden Wörtern vortrainiert worden ist. Es werden mehrere Techniken vorgestellt, mit diesem Modell Bedeutungen aufzulösen.

Ein einfacher Ansatz ist das sogenannte Nearest-Neighbor-Matching (dt. *Nächste-Nachbar-Klassifikation*), bei welchem für alle Trainingswortbedeutungen die Einbettungsvektoren berechnet werden und daraus diejenige Bedeutung bestimmt wird, dessen Vektor die geringste Distanz zum Vektor des jeweiligen Eingabewortes und dessen Kontext hat. Eine mögliche Alternative dazu ist, eine Linearprojektion der Ergebnisvektoren zu berechnen und die Vektorkomponenten zu normalisieren.

Die Autoren stellen als zusätzliche Alternative die Berechnung einer sog. Gated Linear Unit vor. Bei diesem Ansatz wird aus bestimmten Vektorkomponenten ein Gate-Vektor mit Werten zwischen 0 und 1 berechnet, der dann komponentenweise mit anderen Vektorkomponenten multipliziert wird. Anschaulich betrachtet bestimmt der Gate-Vektor also, welche Vektorkomponenten das Gate „ungehindert passieren dürfen“.

Die besten Ergebnisse erzielt letztlich das Gated-Linear-Unit-Verfahren mit F_1-Werten von 71.1 % beim SemEval-2013- [NJV13] und 76.2 % beim SemEval-2015-Aufgabensatz [MN15].

4.1.3. Hybridverfahren

In diesem Abschnitt soll ein Hybrid-Ansatz vorgestellt werden, der sowohl vortrainierte Sprachmodelle als auch Wissensnetze zur Wortbedeutungsauflösung einsetzt.

Das Projekt EWISER, das in der Arbeit *„Breaking Through the 80% Glass Ceiling“* [BN20] von Bevilacqua et al. vorgestellt wird, verwendet ebenfalls BERT für kontextuelle Einbettungen und kombiniert diese mit WordNet-Informationen. Die Einbettungsvektoren aus BERT werden mit einer Gewichtungsmatrix multipliziert, um eine Wahrscheinlichkeitsverteilung über alle Synsets zu erhalten, woraus dann das wahrscheinlichste Synset gewählt wird, das gleichzeitig ein mögliches Synset für das aufzulösende Wort ist. Beim Training werden die Koeffizienten dieser Gewichtungsmatrix optimiert: Falls ein falsches Synset die höchste Wahrscheinlichkeit zugewiesen bekommt, werden die Koeffizienten entsprechend korrigiert. Nun soll das System unterscheiden können, wann ein falsches Synset zumindest mit dem korrekten verwandt ist und wann nicht. Hierfür werden WordNet-Beziehungen zwischen den Bedeutungsknoten genutzt, um in der Wahrscheinlichkeitsverteilung verwandte Synsets stärker zu gewichten. Die Autoren experimentieren mit verschiedenen Beziehungsarten wie verschiedene Kombinationen von Hyperonym- und Hyponym-Beziehungen sowie deren transitiven Hüllen.

Beste Ergebnisse werden bei der Einbeziehung von Hyperonym-Beziehungen und bedeutungsannotierten WordNet-Glossen und WordNet-Beispielsätzen erzielt mit F_1-Werten von 80.7 % und 81.8 % bei SemEval-2013 [NJV13] und -2015 [MN15].

4.2. Quelltextanalyse

In diesem Abschnitt sollen Arbeiten vorgestellt werden, die Quelltextanalyse nutzen, um natürlichsprachliche Informationen aus dem Quelltext semantisch weiterzuverarbeiten. Dabei werden Arbeiten zur Quelltext-Beschreibung, Anforderungs- und Konzeptrückverfolgung beschrieben. Der Fokus soll hier auf den Quelltextelementen liegen, die als Informationsquelle herangezogen werden.

Das Feld der Quelltext-Beschreibung (engl. *code summarization*) erforscht Wege, das aufwendige manuelle Untersuchen und Verstehen von Quelltext maschinengestützt zu vereinfachen, um Menschen bei der Software-Entwicklung zu helfen. Die Möglichkeiten reichen dabei von automatischer Javadoc-Kommentargenerierung bis zum Generieren kurzer Stichpunktlisten, um Ergebnisse semantischer Suchanfragen ansprechend aufzubereiten.

Die Arbeit *„Automatic generation of natural language summaries for Java classes“* [Mor+13] von Moreno et al. zeigt, wie Beschreibungen für Java-Klassen generiert werden können. Um eine einzelne Klasse zu beschreiben, werden u. a. ihre implementierten Schnittstellen, Oberklassen, innere Klassen sowie ihre relevantesten Methoden bestimmt und daraus teils mehrere natürlichsprachliche Sätze generiert. Dafür wird der UML-Stereotyp einer Klasse bestimmt, der angibt, ob es sich bspw. um eine Kontrollklasse, eine Fabrikklasse oder eine Hilfsklasse handelt. Ausgehend davon ist auch die Relevanz einzelner Methoden in Abhängigkeit vom vorliegenden Stereotyp gegeben, wobei zusätzlich die Methoden-Sichtbarkeiten miteinbezogen werden.

Eine Nutzerstudie auf insgesamt 40 Klassen entnommen von zwei mittelgroßen bis großen Open-Source-Projekten ergab, dass in 69 % der generierten Beschreibungen keine wichtigen Informationen fehlen und 96 % der Beschreibungen prägnant, lesbar und verständlich sind.

In der Arbeit *„Automatic Source Code Summarization of Context for Java Methods“* [MM16] von McBurney et al. wird ein Verfahren zur Beschreibungsgenerierung für Methoden vorgestellt. Dabei werden Methodenname, Rückgabetyp und Parameternamen für eine Kurzzusammenfassung verwendet. Zusätzlich wird der Kontext der Methode miteinbezogen, indem mittels PageRank im Aufrufgraphen die wichtigsten aufrufenden und aufgerufenen Methoden bestimmt werden und für die beiden Gruppen jeweils zwei beschreibende Sätze generiert werden. Außerdem werden noch Beispiele zur Verwendung der Methode im gesamten Quelltext ausfindig gemacht und eine Auswahl präsentiert.

Es wurde auf 24 Methoden von sechs kleinen bis großen Open-Source-Projekten einer Nutzerstudie durchgeführt. Neben zwei Freitextfragen wurden sechs Fragen zur Qualität der generierten Beschreibungen gestellt, die mit Werten von 4 („stimme sehr zu“) bis 1 („stimme gar nicht zu“) beantwortet werden sollten. Für die Aussage „Grundsätzlich finde ich die Beschreibung angemessen“ wurde als Mittelwert 2.635 (Varianz 0.558) für die generierten Beschreibungen ermittelt, für die manuell erstellten Beschreibungen als Mittelwert 3.032 (Varianz 0.676). Der Vergleich zwischen den generierten Beschreibungen und Beschreibungen, die mit State-of-the-Art-Ansätzen generiert worden waren, lieferte für dieselbe Frage als Mittelwert 3.015 für den Ansatz von McBurney et al. und 2.390 für die Konkurrenz (Varianz beide 0.863). Verglichen mit State-of-the-Art-Ansätzen erreichen McBurney et al. also eine bessere Qualität, sie sind jedoch manuell erstellten Beschreibungen unterlegen.

In der Arbeit *„Do data dependencies in source code complement call dependencies for understanding requirements traceability?“* [Kua+12] von Kuang et al. wird untersucht, inwiefern die Berücksichtigung von Datenabhängigkeiten zusätzlich zu Aufrufabhängigkeiten die Anforderungsrückverfolgung verbessern kann. Um Daten- und Aufrufabhängigkeiten zu bestimmen, werden zur Laufzeit über JVM[2]-Inspektion Methodenaufrufe und JVM-Speicher-Zugriffe überwacht und mitprotokolliert. Die Autoren haben sich bewusst für eine dynamische Erfassung der Abhängigkeiten

[2] Java Virtual Machine

entschieden, da vor allem bei Datenabhängigkeiten statische Analyseverfahren ein tatsächlich benutztes Objekt als mehrere verschiedene Entitäten verarbeiten können u. a. aufgrund von Schwierigkeiten bei der statischen Auflösung dynamischer Bindung. Es werden also konkrete Speicherzugriffe protokolliert und daraus dann Datenabhängigkeitsinformationen ermittelt. All diese Abhängigkeitsinformationen werden genutzt, um basierend auf gegebenen Zuordnungen zwischen Anforderungen und Eingabequelltext neue Zuordnungen vorzuschlagen.

In einer Nutzerstudie wurden für drei kleine bis große Open-Source-Projekte unterschiedlicher Domänen Goldstandards für Zuordnungen von 50 Anforderungen ermittelt. Gegen diese Musterlösungen wurde dann das System getestet, wobei F_1-Werte von 63 % bis 85 % erreicht wurden. Dabei konnte verglichen mit bloßem Einsatz von Aufrufabhängigkeiten durch zusätzlichen Einsatz von Datenabhängigkeiten die Fehlerrate um 1.6 % gesenkt werden.

Die Arbeit *„Facilitating program comprehension with call graph multilevel hierarchical abstractions“* [AGL21] von Alanazi et al. hat zum Ziel, Methoden-Aufrufgraphen in weniger feingliedrige Repräsentationen verschiedenen Detailgrades umzuwandeln, um Menschen mithilfe dieser gröberen Visualisierungen das Quelltextverstehen zu erleichtern. Als einzige Datenquelle dient der Methoden-Aufrufgraph des Eingabe-Projektes, um unter Verwendung von Clustering-Verfahren Klassen- und Paket-Aufrufgraphen zu erstellen. Dabei besteht zwischen zwei Klassen eine Aufrufabhängigkeit, wenn eine Methode der ersten Klasse eine Methode der zweiten Klasse aufruft. Paket-Aufrufabhängigkeiten werden analog über Klassen formuliert. Verschiedene Einstellmöglichkeiten sollen die Benutzung des Visualisierers erleichtern.

In einer Nutzerstudie wurde der Quelltext des Open-Source-Projektes „Sweet Home 3D“ genutzt, um Aufgaben zum Quelltextverständnis unter der Verwendung des vorgestellten Visualisierungswerkzeugs zu stellen. Das System erreichte hierbei eine Gebrauchstauglichkeit (engl. *system usability score*) von 72.6 %, was über der Schwelle für „gute Tauglichkeit“ von 68 % liegt.

In der Arbeit *„Improving topic model source code summarization“* [McB+14] von McBurney et al. wird ein Themenmodell (engl. *topic model*) aus Quelltext erstellt, um einzelnen Methoden Schlüsselwörter zuordnen zu können, die deren Funktion und Absicht möglichst gut beschreiben. Methoden werden dabei in einem sogenannten Hierarchischen Dokument-Themenmodell (engl. *hierarchical document topic model, HDTM*) angeordnet. Dafür wird ein Aufrufgraph mittels PPR in eine Baumstruktur überführt, sodass Methoden mit grober Thematik über Methoden mit feinerer Thematik stehen. Damit ein tatsächliches Themenmodell entsteht, werden jeder Methode alle Wörter, die in Bezeichnern in ihrem Quelltextabschnitt vorkommen, initial zugeordnet und während der PPR-Ausführung zwischen den Methoden in der Hierarchie auf- und abwärts geschoben, sodass semantisch grobe Schlüsselwörter weiter oben und semantisch feinere Schlüsselwörter weiter unten landen. Als Ergebnis erhält man eine Hierarchie der Methoden des Eingabequelltextes basierend auf ihrer semantischen Feingliedrigkeit und zusätzlich zu jeder Methode Schlüsselwörter, die als Oberbegriffe für den Methodeninhalt und die darunter liegenden Methoden dienen.

In einer Nutzerstudie auf 15 Methoden aus sechs kleinen bis großen Open-Source-Projekten wurde in 76.9 % der Fälle als Bewertung der errechneten Schlüsselwörter mindestens „einigermaßen zutreffend“, in 43.6 % der Fälle sogar „sehr zutreffend“ angegeben.

4.3. Wissensanreicherung

In diesem Abschnitt sollen Arbeiten vorgestellt werden, die Wörter in einer natürlichsprachlichen Eingabe mit semantisch verwandten Begriffen anreichern. Das Ziel ist hierbei, die Verarbeitung natürlicher Sprache zu verbessern. In vielen Ansätzen werden dafür semantische Relationen in Wissensnetzen als Quelle genutzt.

Die Arbeit *„Context Model Acquisition from Spoken Utterances“* [WHT17] von Weigelt et al. zeigt einen Ansatz, wie Begriffe in gesprochener Sprache mit Wissen aus WordNet angereichert werden

können. Das vorgestellte Verfahren kann beispielsweise dazu benutzt werden, die Verarbeitung natürlichsprachlicher Befehle zur Roboter-Fernsteuerung zu verbessern. Im Beispielsatz *„Open the fridge, grab the milk and close the door"* bezieht sich „door" auf die Tür des Kühlschranks. Das System kann dies erkennen, sofern es mittels Zugriff auf Weltwissen folgern kann, dass hier „door" ein Meronym von „fridge" ist, also eine Teil-Ganzes-Beziehung vorliegt. Ebenso ist in der Befehlsfolge *„Get the tea and orange juice from the kitchen. Put the beverages on the dining table"* mit „beverages" zuvor genannter „tea" und „orange juice" gemeint. Auch hier ist „beverage" ein Hyperonym von „orange juice" und „tea". Mit diesem Wissen kann der Wortbezug aufgelöst werden.

Um einzelne Wörter mit Wissen anzureichern, werden über alle möglichen Bedeutungsknoten der Eingabe hinweg gemeinsame Oberbegriffe in WordNet gesucht. Laut Weigelt et al. wäre ein naiver Ansatz, zu je zwei Bedeutungsknoten den Lowest Common Subsumer zu wählen, was aber dazu führen würde, dass für zwei sinnfremde Wortbedeutungen ungeeignete WordNet-Wurzelknoten wie „Entity" oder „Artifact" bestimmt werden. Um also zu gewährleisten, dass gefundene Oberbegriffe hinreichend spezifisch sind, wird die Ähnlichkeitsmetrik von Lin [Lin98] (siehe Abschnitt 2.4.2) verwendet: Falls für zwei Bedeutungsknoten a und b gilt, dass $\mathrm{sim}_{\mathrm{Lin}}(a, b) \geqslant 0.7$ ist, so wird deren LCS als Oberbegriffs-Elternknoten für a und b eingetragen. Falls hierbei a ein Oberbegriff von b ist, so kann direkt a als Elternknoten von b gewählt werden. Dieser Prozess wird iterativ wiederholt und so eine Hierarchie aufgebaut, bis keine neuen Oberbegriffe mehr hinzugefügt werden können. Es ist zu beachten, dass zu einem einzelnen Knoten keine Oberbegriffe angereichert werden. Anschließend werden Teil-Ganzes-Beziehungen zwischen den initialen Bedeutungsknoten aus WordNet gesucht und ebenfalls in den Graphen eingetragen.

In einer Nutzerstudie wurden zu zwei Szenarien Roboter-Instruktionen aufgezeichnet und transkribiert. Dazugehörige Musterlösungen enthielten über 200 WordNet-Beziehungen. Es wurden F_1-Werte von 78.6 % für die Oberbegriffsbestimmung und 92.7 % für die Teil-Ganzes-Beziehungsbestimmung erreicht, wobei die zugehörigen Präzisionswerte bei 68.0 % bzw. 89.7 % lagen. Die niedrigen Präzisionswerte wurden damit erklärt, dass im beschriebenen Ansatz vor der Wissensanreicherung keine Wortbedeutungsauflösung erfolgt.

Die Arbeit *„Unsupervised Multi-Topic Labeling for Spoken Utterances"* [Wei+19] von Weigelt et al. beschreibt einen Ansatz, um auf Spracheingaben eine sogenannte Themen-Beschriftung (engl. *topic labeling*), die Zuordnung von Themenüberschriften zu Gruppen thematisch zusammengehöriger Begriffe, zu bestimmen. Nach einem Wortbedeutungsauflösung-Schritt wird ein Themengraph (engl. *topic graph*) gebaut, aus welchem mehrere Hauptthemen identifiziert werden, welche dann als Themen-Beschriftungen verwendet werden. Dafür wird wie folgt vorgegangen: Aus den initialen Bedeutungsknoten aus DBpedia, die durch WSD bestimmt worden sind, werden Subgraphen gebaut, indem zusätzlich alle Knoten aus DBpedia hinzugefügt werden, die von den initialen Knoten aus innerhalb von zwei Kantenschritten erreichbar sind. Diese Subgraphen werden nun als ein großer Graph betrachtet, der womöglich nicht zusammenhängend ist. Es wird nun angenommen, dass jede Zusammenhangskomponente dieses Graphen einen separaten Themenbereich repräsentiert. Um nun für jeden Themenbereich Beschriftungs-Kandidaten zu finden, gibt es zwei mögliche Herangehensweisen. Die erste (genannt *top connectitvity strategy*) wählt für einen Themenbereich diejenigen Knoten, die am stärksten mit den initialen Bedeutungsknoten verbunden sind[3], wobei im Falle des Gleichstands Gewichte berücksichtigt werden, die durch PPR auf dem Graphen ermittelt worden sind. Die zweite (genannt *max coverage strategy*) versucht hingegen, diejenigen Knoten zu wählen, die am meisten noch unerreichte initiale Bedeutungsknoten abdecken, sodass möglichst viele Bedeutungen eine Beschriftung zugeordnet haben. Im Falle des Gleichstands werden auch hier wieder PPR-Gewichte hinzugezogen. Falls alle Bedeutungen abgedeckt sind, wird zur *top connectivity strategy* gewechselt. Am Ende werden für beide Strategien die Beschriftungen nach PPR-Gewichten ihrer Knoten sortiert.

[3] Gegeben durch die Anzahl an Subgraphen, in denen ein Knoten vorhanden ist

Für eine Nutzerstudie wurden 16 Sprachaufnahmen transkribiert und die Ergebnisse der anschließenden Themen-Beschriftung von sechs Individuen bewertet. In 90.9 % der Fälle wurde mindestens eine der vorgeschlagenen Themen-Beschriftungen unter den ersten vier als eine gute Wahl bewertet. Ferner wurde in 77.2 % der Fälle die zuerst vorgeschlagene Beschriftung als „gut, aber zu allgemein“ bewertet. Die *max coverage strategy* erzielt dabei leicht bessere Ergebnisse als die *top connectivity strategy*.

In der Arbeit *„Knowledge-based graph document modeling“* [SP14] von Schuhmacher et al. wird ein Ansatz zur Themengliederung von Textdokumenten vorgestellt, welche über Themen-Beschriftung realisiert wird. Nach einem Wortbedeutungsauflösungs-Schritt wird dafür zunächst ein DBpedia-Subgraph aus den initialen Bedeutungsknoten eines Textausschnittes gebildet. Anders als beim vorherigen Ansatz werden alle Pfade im gesamten DBpedia-Graphen der Länge 2 hinzugefügt, die zwei initiale Bedeutungsknoten miteinander verbinden. Anschließend werden die Kanten nach einer Informationsgehalts-Metrik gewichtet (siehe Abschnitt 2.4.1). Vorgestellt wird unter anderem die sogenannte „Combined-Information-Content“-Metrik, welche einer Kante $e = (\text{subj}, \text{pred}, \text{obj})$ das Gewicht $w_{\text{CombIC}}(e) = \text{I}(\text{pred}) + \text{I}(\text{obj})$ zuordnet. Die vorgestellten Metriken können als Maß für die Relevanz einer Kante verstanden werden. Für einen Ausgangsknoten u im Subgraphen wird nun für alle übrigen Knoten v gemäß der Kantengewichte die Relevanz des relevantesten Pfades von u nach v bestimmt (wobei die Richtung der Kanten ignoriert wird). Nach dieser Relevanz werden die übrigen Knoten sortiert und daraus eine Beschriftungs-Kandidatenliste für den Ausgangsknoten bestimmt.

Für die Evaluation wurden die KORE-Entity-Ranking-Datensätze [Hof+12] verwendet und der Spearman'sche Rangkorrelationskoeffizient ρ berechnet. Im Durchschnitt liegt dieser bei 0.624, wobei die CombIC-Metrik die besten Ergebnisse erzielt.

In der Arbeit *„Automatic Labelling of Topics with Neural Embeddings“* [BLB16] von Bhatia et al. wird ein lernbasierter Ansatz für Themen-Beschriftung beschrieben. Dafür werden die Sprachmodelle doc2vec und alternativ word2vec mit englischsprachigen Wikipedia-Artikeln trainiert mit dem Ziel, Worteinbettungen für Themen-Beschriftungen aus natürlichsprachlichen Texten zu generieren. Die so generierten Beschriftungs-Kandidaten werden nach ihrer Kosinus-Ähnlichkeit zu Worteinbettungen der wichtigsten Eingabewörter sortiert. Diese Rangfolge wird in einem zweiten Schritt weiter verfeinert: Ein überwachtes Rangordnungs-Modell (engl. *learn-to-rank model*) kombiniert mehrere Merkmale wie PageRank-Gewichte der Wikipedia-Artikel, Wortzahl der Beschriftungs-Kandidaten oder Trigramm-Ähnlichkeit mit den wichtigsten Eingabewörtern und optimiert die Rangfolge der ausgegebenen Themen-Beschriftungen.

In einer Nutzerstudie wurden aus insgesamt über 200 Themengebieten von vier verschiedenen Domänen die besten 19 Beschriftungs-Kandidaten von je 10 Teilnehmenden bewertet. Als Bewertungsskala wurden Werte von 0–3 („sehr unangemessene Beschriftung“ bis „sehr gute Beschriftung“) gewählt. Für die vier Domänen wurden als Bewertungen für die ranghöchsten Kandidaten Mittelwerte zwischen 1.99 und 2.02 ermittelt.

5. Analyse

Diese Arbeit hat zum Ziel, Begriffe in Quelltexten mit Wissen anzureichern, um Verfahren zur Anforderungsrückverfolgung (engl. *traceability link recovery, TLR*) zu verbessern. Dabei soll der Fokus auf Verfahren zu TLR liegen, die Zuordnungen über Worteinbettungen der Begriffe in den jeweiligen Software-Artefakten herstellen. Die Ergebnisse meiner Arbeit sollen anschließend in das FTLR-Projekt von Hey et al. [Hey+21] (siehe Kapitel 3) integriert werden, weshalb in diesem Kapitel einige Annahmen zugunsten dieser Integration getroffen werden, was an den zugehörigen Stellen erwähnt wird. Quelltexte seien im Folgenden kompilierbar und in Java verfasst.

TLR kommt bei der Entwicklung und Wartung von Software zum Einsatz. Solche Software-Projekte decken beliebige Domänen ab. Domänenbeschränkte Ansätze zur TLR können zielgerichtet maßgeschneidert und optimiert werden, können dafür allerdings nur in Domänenbereichen, für die sie entwickelt worden sind, zuverlässig gute Ergebnisse erzielen. Domänenflexible Ansätze hingegen müssen so konzipiert sein, dass sie auch mit Eingaben aus unbekannten Domänen gute Ergebnisse erzielen können. Aufgrund der grundsätzlich domänenflexiblen Natur der Software-Entwicklung liegt es daher nahe, für diese Arbeit nur die domänenflexible TLR und zugehörige Wissensanreicherungs-Ansätze zu betrachten.

Als Nächstes soll die Notwendigkeit der Wissensanreicherung im Quelltext für die Anforderungsrückverfolgung begründet werden. Es sind nicht alle Informationen, die für die Zuordnung von Anforderungen zu Quelltextabschnitten notwendig sind, in der Anforderungsdokumentation und im Quelltext bereits enthalten. Beispielsweise kann in der Anforderung ein *„display“* erwähnt werden, während im Quelltext dafür ausschließlich *„user interface“* oder *„UI“* verwendet wird. Durch reinen Textvergleich ist in einem solchen Fall die Zuordnung nicht möglich. Mit dem Wissen, dass die Bedeutungen von *„display“* und *„user interface“* verwandt sind, kann die Zuordnung gelingen. Es ist also notwendig, für die TLR Wissen anzureichern, um Informationslücken zu schließen, die die Rückverfolgbarkeit erschweren.

Begriffe, die im Quelltext vorkommen, basieren überwiegend auf natürlicher Sprache und sind daher potentiell mehrdeutig. Da ein Software-Projekt üblicherweise die Software-Domäne und mindestens eine Anwendungs-Domäne überdeckt und zudem noch Domänenflexibilität vorausgesetzt ist, ist die Auflösung der Bedeutungen der vorkommenden Begriffe nicht trivial. Damit zielführend Wissen zu Begriffen im Quelltext angereichert werden kann, müssen daher deren Bedeutungen aufgelöst werden. Es ist anzumerken, dass Wortbedeutungsauflösung (im Folgenden auch *WSD*) bereits eine Form der Wissensanreicherung ist.

Als erstes Teilproblem kann die Frage nach dem Umfang der Wissensanreicherung verstanden werden. Schlüsselwörter wie `abstract`, `class`, `implements` oder `protected` werden in der worteinbettungsbasierten TLR nicht miteinbezogen, da sie als Implementierungsdetail in der Regel keine Entsprechung in der Anforderungsdokumentation besitzen.[1] Dies gilt auch für Bezeichner wie `HashtableHelper`, die aus programmierbezogenen Stoppwörtern wie `Hashtable` und `Helper` bestehen. Außerdem sind Vorgehensweisen im Quelltext häufig feingliedriger formuliert als in der Anforderungsdokumentation. In früheren Arbeiten wie *„Improving Traceability Link Recovery Using Fine-grained Requirements-to-Code Relations“* [Hey+21] von Hey et al. wurde daher die Menge der Begriffe, deren Bedeutung aufgelöst wird, eingeschränkt. Es ist also zu untersuchen, für welche Begriffe im Quelltext WSD und Wissensanreicherung eine Verbesserung für die TLR bewirken.

Zudem muss betrachtet werden, welche Informationen im Eingabequelltext für die Wissensanreicherung einzelner Begriffe herangezogen werden können. Dafür muss zunächst untersucht werden, welche Arten von anzureicherndem Wissen für diese Arbeit zielführend sind. Es können Vorverarbeitungsschritte hilfreich sein, um aus dem Eingabequelltext Wissen zu extrahieren (siehe Abschnitt 2.1). Analog zum vorherigen Absatz muss untersucht werden, in welchem Umfang Informationen miteinbezogen werden. Quelltextteile, welche nicht mit Wissen angereichert werden, können trotzdem noch als Quelle für die Wissensanreicherung infrage kommen. Beispielsweise kann die Berücksichtigung von Bezeichnern in Methodenrümpfen die Bedeutungsauflösung von Wörtern im Methodennamen zuverlässiger machen.

Das zweite Teilproblem stellt die Frage nach der geeignetsten Realisierung der WSD dar. Es muss einerseits sowohl ein geeignetes Verfahren gewählt als auch je nach Verfahren geeignete benötigte externe Datenquellen auf ihre Eignung untersucht werden. WSD benötigt zudem wie in Abschnitt 2.1.6 beschrieben Kontextinformationen, um einzelne Wörter korrekt auflösen zu können. Es muss also andererseits betrachtet werden, welche Begriffe im Quelltext überhaupt als Kontextwörter einzelner Begriffe geeignet sind. Wie oben erwähnt tragen viele Schlüsselwörter einer Programmiersprache keine für die Wortbedeutungen relevante Information. Zudem muss betrachtet werden, welche Quelltextbereiche in welchem Umfang als Kontext für einen einzelnen Begriff infrage kommen. In natürlichsprachlichen Texten kann bspw. für die Bedeutungsauflösung der Wörter eines Satzes der vorhergehende und der nachfolgende Satz als Kontext verwendet werden. Dies ist auf Quelltexte aufgrund ihrer hierarchischen Struktur nur begrenzt übertragbar. Es ist also zu untersuchen, welche Vorkommen von natürlichsprachlichen Begriffen im Quelltext als Kontext für die Bedeutungsauflösung eines einzelnen Begriffs zur Verfügung stehen und wie sie die Ergebnisse der Bedeutungsauflösung beeinflussen.

Das dritte Teilproblem ist die Frage, auf welche Weise und in welchem Umfang Wissensanreicherung über WSD hinaus die TLR verbessert. Die Einführung von Rauschen durch irrelevante Information kann die TLR erschweren: Wenn bspw. für große Mengen an Begriffen das gleiche unspezifische Wissen angereichert wird, kann dies Zuordnungen zwischen Anforderungen und Quelltextabschnitten suggerieren, welche gar nicht semantisch zusammengehörig sind. Es müssen also verschiedene Vorgehensweisen miteinander verglichen werden und auch untersucht werden, wie der Umfang des zusätzlich angereicherten Wissens die TLR beeinflusst. Des Weiteren muss untersucht werden, welche externen Wissensquellen für die Anreicherung im vorliegenden Fall geeignet sind.

Damit lassen sich die Teilprobleme der zentralen Problemstellung dieser Arbeit in Form von drei großen Zielfragen zusammenfassen:

[1] Es ist zu beachten, dass z. B. Klassen*namen* oder die *Namen* implementierter Schnittstellen durchaus Entsprechungen in Anforderungstexten haben können, die dies kennzeichnenden Schlüsselwörter an sich aber nicht.

ZF 1: Welche Teile des Quelltextes sollen für die Wissensanreicherung genutzt werden?
- **ZF 1.1:** Welche Begriffe kommen als Ziel von Anreicherung infrage?
- **ZF 1.2:** Welche Informationen im Quelltext können dafür genutzt werden?

ZF 2: Wie lässt sich Bedeutungsauflösung von Begriffen im Quelltext realisieren?
- **ZF 2.1:** Welche Verfahren sind für Bedeutungsauflösung von Begriffen im Quelltext geeignet?
- **ZF 2.2:** Welche Auswirkungen hat die Wahl unterschiedlicher Kontextquellen für Begriffe auf die Bedeutungsauflösung?

ZF 3: Welche Arten von Wissensanreicherung über die Bedeutungsauflösung hinaus führen zu einer Verbesserung von worteinbettungsbasierten Verfahren zur Anforderungsrückverfolgung?

Im Rest des Kapitels werden nun die zugehörigen Teilprobleme analysiert.

5.1. Umfang der Wissensanreicherung

In diesem Abschnitt soll analysiert werden, für welche Bestandteile des Eingabequelltextes Wissensanreicherung zur Verbesserung von worteinbettungsbasierter TLR geeignet ist. Außerdem soll untersucht werden, was als anzureicherndes Wissen infrage kommt und welche Teile des Eingabequelltextes dafür genutzt werden können.

5.1.1. Begriffe, die mit Wissen angereichert werden sollen

Quelltexte enthalten linguistische Information, die für worteinbettungsbasierte TLR nicht relevant ist. Dazu gehören beispielsweise Schlüsselwörter der verwendeten Programmiersprache. Diese tragen zwar strukturelle Information, die z. B. für eine Analyse von Abhängigkeiten im Quelltext genutzt werden kann. Sie tragen allerdings meist keine Bedeutung bezüglich Semantik des Quelltextverhaltens. Ihre natürlichsprachliche Bedeutung unterscheidet sich teilweise deutlich von der programmiersprachlichen Bedeutung. Somit sind sie für die Ziele dieser Arbeit nicht verwertbar.

Als Beispiele sind `double`, `final`, `finally`, `float`, `long`, `short`, `switch`, `protected` und viele weitere zu nennen.

Genauso im Sinne der WSD und damit auch der worteinbettungsbasierten TLR unverwertbar sind Sonderzeichen wie arithmetische/logische Operatoren, Quelltextblock-Klammerung und Zeichenketten-Trenner sowie numerische Konstanten. Es ist daher sinnvoll, all dies für eine weitere linguistische Verarbeitung auszuschließen.

Zeichenketten-Konstanten können in manchen Fällen für die Kommunikation zwischen Software und Mensch genutzt werden und damit für die TLR verwertbar sein, enthalten in anderen Fällen allerdings Dateisystempfade, Datenbank-Abfragen oder Schlüssel für assoziative Arrays. Da eine detaillierte Untersuchung von Zeichenketten-Konstanten auf ihre zielrelevante Verwertbarkeit den Rahmen dieser Arbeit sprengt, sollen diese im Folgenden genauso von der weiteren Verarbeitung ausgeschlossen werden.

Einzelne Quelltextelemente können mit Annotationen gekennzeichnet sein (z. B. `@NotNull` oder `@Deprecated`). Annotationen beziehen sich üblicherweise auf technische Aspekte und Implementierungsdetails, wie es bei den vordefinierten Annotationen in der Java-Standard-Bibliothek der Fall ist [Orad]. Es ist daher sinnvoll, sie ebenfalls von der weiteren Verarbeitung auszuschließen. Eigene Annotationstypen können mit dem Schlüsselwort `@interface` definiert werden, erben automatisch von der Schnittstelle `java.lang.annotation.Annotation` und werden im Folgenden wie herkömmliche Schnittstellendeklarationen behandelt.

Übrig bleiben demnach Kommentartexte und Bezeichner für lokale, Klassen- und Instanzvariablen sowie Methoden und Datentypen. Diese sollen im Folgenden näher untersucht werden. Da Quelltexte üblicherweise hierarchisch aufgebaut sind (Pakete enthalten Klassen, Klassen enthalten

Methoden usw.), soll die Analyse von oben herab für Elemente der einzelnen Hierarchie-Ebenen erfolgen.

Ein Bezeichner heiße im Folgenden *alphanumerisch*, wenn er aus alphabetischen Groß-/Kleinbuchstaben, Ziffern oder Unterstrichen besteht, nicht mit einer Ziffer beginnt und ungleich einem einzelnen Unterstrich (_) ist[2]. Ein Bezeichner heiße *großbeginnend*, wenn er mit einem Großbuchstaben beginnt, und *kleinbeginnend*, wenn er mit einem Kleinbuchstaben beginnt. Ein Bezeichner heiße *großbuchstabig*, wenn er keine Kleinbuchstaben enthält, und *kleinbuchstabig*, wenn er keine Großbuchstaben enthält. Diese Definitionen sollen der Mehrdeutigkeit der Begriffe „groß-/kleingeschrieben“ entgegentreten.

Pakete

Pakete sind die grobgliedrigsten Strukturelemente innerhalb eines Java-Projektes. Ein Paket kann Datentypdefinitionen und weitere Unterpakete zu einer Einheit zusammenfassen.

Pakete müssen einen eindeutigen *vollständigen Namen* haben. Dieser Name kann aus (möglicherweise mehreren) alphanumerischen Komponenten bestehen, die durch Punkt-Zeichen getrennt werden [Orac, Abschnitt 6.5]. Es ist Konvention, dass die ersten Komponenten aller vollständigen Paketnamen eines Projektes mit Kürzeln für die eindeutige Identifizierung des Projektes belegt sind (z. B. `edu.kit.ipd.indirect`) [Oraa]. Da diese üblicherweise keine zielrelevante Information enthalten, empfiehlt es sich vor der weiteren Verarbeitung, für alle Namen neu definierter Pakete deren größtes gemeinsames Präfix zu bestimmen und zu verwerfen. Die übrigen Komponenten können Begriffe enthalten, die die Kindelemente der Pakete zusammenfassend beschreiben. Es ist zu beachten, dass Namenskonventionen für Paketnamen von der Verwendung von Großbuchstaben abraten und Unterstriche unüblich sind [Oraa]. Daher kann es vorkommen, dass zusammengesetzte Paketnamenskomponenten nicht nach Binnenmajuskel- oder Binnenunterstrich-Schreibweise in ihre Bestandteile zerlegbar sind.

Einem Paket kann ein beschreibender Javadoc-Kommentar zugeordnet sein [Orab]. Dieser gibt häufig eine Zusammenfassung darüber, welche Datentypen/Unterpakete das Paket enthält.

Aus Perspektive der TLR sind Pakete zu grobgliedrig, um miteinbezogen zu werden. Es ist anzunehmen, dass die grobe Gruppierung von Anforderungen innerhalb der Anforderungsdokumentation üblicherweise anders strukturiert ist als die Gruppierung von Klassen in einzelne Pakete. Diese Annahme kann damit begründet werden, dass Anforderungsdokumente viel stärker durch Anwendersicht geprägt sind, während Quelltext überwiegend technischen Abstraktionskriterien genügen soll. Eine Zuordnung von Artefaktelementen sollte sich daher auf unterhalb der Paketebene beschränken. Tatsächlich werden im FTLR-Projekt Paketnamen nicht miteinbezogen [Hey+21].

Datentypen

Zu den Datentypen zählen Klassen, Schnittstellen und Aufzählungstypen. Klassen, Schnittstellen und Aufzählungstypen können innere Klassen/Schnittstellen/Aufzählungstypen, Klassen- und Instanzvariablen sowie Methoden zu einer Einheit zusammenfassen. Aufzählungstypen können zudem noch Aufzählungskonstanten enthalten, welche im Folgenden wie Klassenvariablen behandelt werden.

Datentypen haben alle einen sogenannten *einfachen Namen*, der aus einem alphanumerischen Bezeichner besteht und auf der Ebene des übergeordneten Pakets eindeutig sein muss. Außerdem haben sie einen projektweit eindeutigen *vollständigen Namen*, der sich aus dem vollständigen Namen des übergeordneten Pakets, einem Punkt-Zeichen und dem einfachen Namen zusammensetzt.

[2] Diese Benennungs-Kriterien stimmen mit der Java-Spezifikation überein [Orac, Abschnitt 3.8].

Es ist Konvention, dass sich der einfache Name aus einem oder mehreren Substantiven mittels großbeginnender Binnenmajuskel-Schreibweise zusammensetzt [Oraa]. Genauso sollen Akronyme großbeginnend und kleinbuchstabig fortgesetzt verwendet werden (z. B. `Html` statt `HTML`). So ergeben sich einfache Namen wie `HtmlParser` oder `CodeDependencyAnalyzer`. Allerdings kommt es vor, dass manche dieser Konventionen missachtet werden (z. B. `XMLexport` statt `XmlExporter` oder `DrawRect` statt `RectangleDrawer`).

Die Verwendung von Ziffern kann Versionsinformationen anzeigen wie z. B. in `Html5Parser`. Solche repräsentieren im Allgemeinen technische Details, welche in Anforderungstexten üblicherweise nicht vorkommen und daher auch bei der Rückverfolgung ausgelassen werden wie in der Arbeit von Hey et al.

Bei der anforderungsbasierten Modellierung werden Substantive in Anforderungstexten (welche z. B. Akteure oder behandelte Objekte darstellen können) als Klassen oder Schnittstellen modelliert [Som16]. Daher sind Datentypen in der TLR als relevant anzusehen. In der Arbeit von Hey et al. werden für die Rückverfolgung Klassennamen den Methodennamen vorangestellt, um gleich benannte Methoden in verschiedenen Klassen unterscheiden zu können. Außerdem werden Methoden mitsamt ihrer Parameternamen und -typen für die TLR verwendet. Es empfiehlt sich also, die Namen von Datentypen mit Wissen anzureichern.

Da Datentypnamen wie erwähnt aus mehreren Wörtern zusammengesetzt sein können, ist es sinnvoll, die Klassennamen anhand der Binnenmajuskel aufzutrennen. Prinzipiell können in Datentypnamen auch Komposita vorkommen (auch Multiwort-Ausdrücke, engl. *multi-word expressions*; z. B. `CarPark` in `CarParkRegistration`), welche für eine vollständige Verarbeitung erkannt und gesondert aufgelöst werden sollten. Da dies den Rahmen dieser Arbeit übersteigt, soll sich darauf beschränkt werden, in zusammengesetzten Namen die Wörter einzeln anzureichern. Als weiteren Vorbereitungsschritt empfiehlt sich die Lemmatisierung der Wörter, um Sonderfälle bezüglich Flexion oder Numerus der Wörter zu umgehen. Unter den einzelnen Wörtern können außerdem gebräuchliche Abkürzungen wie `Ident`, `Ui` oder `Dao` vorkommen. Es empfiehlt sich, diese Abkürzungen für die weitere Verarbeitung aufzulösen oder alternativ einzelne Wörter, die eine feste Länge unterschreiten, zu verwerfen.

Zusammengesetzte Namen können Wörter verschiedener Wortarten enthalten (z. B. `BigInteger`). Da homonyme Wörter unterschiedlicher Wortarten unterschiedliche Bedeutungen tragen können, ist es für die weitere Verarbeitung (insbesondere WSD) notwendig, Wortarten zu bestimmen. Während dies auf natürlichsprachlichen Texten ein mittlerweile gut erforschtes Problem ist, blieb diese Aufgabenstellung auf Quelltextseite bislang unbeachtet. Eine einfache Heuristik wäre der Abgleich der Wörter mit einem Lexikon, um herauszufinden, ob ein Wort in einer bestimmten gewünschten Wortart vorkommen kann oder was die am häufigsten vorkommende Wortart dieses Wortes ist. Dies kann viele Falschzuordnungen ergeben, da hierbei der Kontext, in dem das Wort verwendet wird, ignoriert wird. Alternativ kann auch versucht werden, die Wörter zu einem einfachen natürlichsprachlichen Ausdruck zusammenzubauen und mit herkömmlichen Werkzeugen dann die Wortarten zu bestimmen. Dies stellt eine vielversprechendere Heuristik dar, da so das Zusammenspiel mehrerer Wörter in die Wortarterkennung miteinfließt. Im Rahmen dieser Arbeit soll sich auf solche Heuristiklösungen beschränkt werden.

Ferner werden in der Programmierung Begriffe und Konzepte verwendet, die rein implementierungstechnischer Natur sind wie z. B. `Hashtable`, `Array`, `Helper`, `Util`, `Factory` etc. Ziel der Arbeit ist es jedoch, Wissen innerhalb der Anwendungsdomänen anzureichern, wie sie auch in der Anforderungsdokumentation abgedeckt werden. Somit empfiehlt es sich, diese programmierbezogenen Stoppwörter anhand vordefinierter Listen herauszufiltern.

Datentypendefinitionen können Javadoc-Kommentare zugeordnet sein. Diese enthalten meist eine Beschreibung der Aufgaben / der Zwecke des beschriebenen Datentyps. Solche Beschreibungen können Informationen aus Anforderungstexten widerspiegeln. Daher ist es sinnvoll, diese

ebenfalls anzureichern. Hier empfiehlt es sich, lediglich den Haupttext des Javadoc-Kommentars zu verwenden und Tags wie `@author` zu verwerfen, da diese keine zielrelevanten Informationen enthalten. Es ist sinnvoll, typische Vorverarbeitungsschritte für natürlichsprachliche Texte anzuwenden. Dazu gehören Tokenisierung, Lemmatisierung und Stoppwortfilterung. Auch hier können programmierbezogene Stoppwörter und Abkürzungen vorkommen, welche wie oben zu behandeln sind. Genauso sollten Ziffern- und Sonderzeichenfolgen verworfen werden.

Klassen- und Instanzvariablen

Klassenvariablen sind statische Attribute einer Klasse. Instanzvariablen sind nicht-statische Attribute einer Klasse. Beide Arten können unterschiedliche Sichtbarkeiten (`public`, `protected`, paket-privat (engl. *package-private*) oder `private`) haben und veränderbar oder nicht veränderbar (`final`) sein. Ihnen kann ein Javadoc-Kommentar zugeordnet sein.

Klassen- und Instanzvariablen haben einen datentypweit eindeutigen Namen, der ein alphanumerischer Bezeichner ist. Veränderliche Klassen- oder Instanzvariablen werden per Konvention mittels kleinbeginnender Binnenmajuskel-Schreibweise benannt, unveränderliche Klassen- oder Instanzvariablen mittels großbuchstabiger Binnenunterstrich-Schreibweise [Oraa]. Ein Javadoc-Kommentar kann eine Klassen- oder Instanzvariable beschreiben.

In der Arbeit von Hey et al. werden Klassen- und Instanzvariablen nicht für die TLR verwendet. Nicht-öffentliche Klassen- oder Instanzvariablen zählen als Implementierungsdetail und haben keine Entsprechung in der Anforderungsdokumentation. Außerdem werden diese üblicherweise als öffentliche Setter- und/oder Getter-Methoden repräsentiert und können in dieser Form zugeordnet werden. Öffentliche Klassen- oder Instanzvariablen sind gemäß Konvention unveränderlich und damit Implementierungsdetail. Somit sollen Klassen- und Instanzvariablen nicht mit Wissen angereichert werden.

Methoden

Methoden bestehen aus einem *Methodenkopf* sowie einem *Methodenrumpf*, wobei Letzterer die Anweisungen der Methode enthält. Im Methodenkopf werden Sichtbarkeiten und andere Modifizierer, der Rückgabetyp, der einfache Name der Methode sowie Parameternamen und -typen definiert. Der *einfache Name* einer Methode oder eines Parameters ist ein alphanumerischer Bezeichner, der per Konvention mit kleinbeginnender Binnenmajuskel-Schreibweise gebildet wird [Oraa]. Der *vollständige Name* einer Methode setzt sich aus dem übergeordneten vollständigen Datentypnamen, zwei Doppelpunkt-Zeichen und dem einfachen Namen der Methode zusammen. Die *vollständige Signatur* einer Methode besteht aus ihrem vollständigen Namen, an welchen in Klammern und kommagetrennt die vollständigen Parametertypnamen angehängt werden. Die *einfache Signatur* einer Methode verwendet statt dem vollständigen Datentypnamen den einfachen Datentypnamen. Zwei Methoden mit dem gleichen vollständigen Namen und unterschiedlicher Signatur heißen *Überladungen* voneinander. Es kann keine zwei Methoden mit gleicher vollständiger Signatur geben. Zusätzlich soll noch die *erweiterte Signatur* definiert werden, welche wie die einfache Signatur gebildet wird, jedoch nicht nur die Parametertypnamen verwendet, sondern an jene noch die Parameternamen mit verbindendem Leerzeichen anhängt, wie es die Syntax für Parameterdeklarationen vorschreibt.

Einfache Methodennamen sind oft aus mehreren Wörtern zusammengesetzt, welche zusammen eine Verbalphrase bilden, z. B. `getPatientRecord`, `updateUserInterface` oder `renderPlayerSprite`. Dies korreliert mit der Auffassung, dass Methoden das Verhalten einer Klasse beschreiben. Tatsächlich werden bei der anforderungsbasierten Modellierung Verben, die das Verhalten zwischen Subjekten und Objekten beschreiben, als Methoden der entsprechenden Klassen mit zugehörigen Parametern entworfen [Som16]. Für die Beispielaussage *„the reception creates a new record for the patient“* kann so eine Methode `Reception::createRecord(Patient)` angelegt werden. Nun gibt es aber Methoden, welche interne Implementierungsdetails einer Klasse behandeln und damit

in Anforderungstexten keine Entsprechung finden. Diese sind üblicherweise nicht öffentlich. Die öffentlichen Methoden einer Klasse hingegen sind gute Kandidaten für die Zuordnung zu Anforderungen. Es empfiehlt sich also, öffentliche Methoden mit Wissen anzureichern.

Nun muss untersucht werden, welche Bestandteile einer Methode geeignet sind, um durch Wissensanreicherung die TLR zu verbessern. Wie oben bereits erläutert, entsprechen der übergeordnete Datentypname und der einfache Methodenname häufig Subjekt und Verb oder Verbalphrase einer Anforderungsaussage. Wie auch bei Datentypnamen empfiehlt sich eine Tokenisierung, Lemmatisierung, Stoppwortfilterung und Abkürzungsbehandlung für die einfachen Methodennamen. Ebenso kann Wortarterkennung Verben von anderen Wortarten unterscheiden und damit für die spätere Bedeutungsauflösung nützlich sein.

Genauso repräsentieren Parameternamen und -typen häufig die Objekte einer solchen zugehörigen Verbalphrase und sollen daher genauso (mit denselben Vorverarbeitungsschritten) mit Wissen angereichert werden.

Einer öffentlichen Methode ist häufig ein Javadoc-Kommentar zugeordnet, der das Verhalten der Methode beschreibt. Diese Beschreibung enthält meist häufig mehr Information über das Verhalten der Methode, als allein aus der Signatur abgelesen werden kann. Daher können Javadoc-Kommentare die TLR ebenfalls unterstützen und sollen daher auch mit Wissen angereichert werden. Tags wie `@param`, `@return` oder `@throws` beschreiben häufig Implementierungsdetails bzw. wiederholen Informationen aus der Methodensignatur und sollen daher verworfen werden. Genauso sollen `@link`-Konstrukte vereinfacht und HTML-Tags verworfen werden.

Der Rumpf einer Methode enthält Methodenaufrufe, Deklaration und Nutzung lokaler Variablen, Zugriff auf Attribute und verschiedene Kontrollstrukturen. Es ist anzunehmen, dass dies hauptsächlich Implementierungsdetails abdeckt und daher nicht angereichert werden soll. Die Arbeit von Hey et al. verwendet für die TLR ebenfalls nur Methodennamen, Parameternamen und -typen und verwirft Rumpfanweisungen.

Kommentare im Methodenrumpf können einerseits Implementierungsdetails beschreiben sowie unfertige oder fehlerhafte Quelltextabschnitte kennzeichnen. Andererseits können sie auch auf klassenübergreifende Zusammenhänge hinweisen wie beispielsweise `// Record is already created` oder `// in case of collision, player may suffer damage twice`. Rumpfkommentare sollen daher mit Wissen angereichert werden und deren Effekt auf die Verbesserung der TLR untersucht werden. Dafür ist es notwendig, Rumpfkommentare wie oben beschrieben vorzuverarbeiten. Zusätzlich müssen Vorkommen von Sonderzeichen im Falle auskommentierter Quelltextabschnitte dafür verworfen werden und potentiell vorkommende Bezeichner vorher tokenisiert werden.

Zusammenfassung

Im Quelltext sollen Datentypnamen und zugehörige Javadoc-Kommentare, öffentliche Methodennamen und deren Parametertypen/-namen (also die erweiterte Methodensignatur) und zugehörige Javadoc-Kommentare mit Wissen angereichert werden. Der Nutzen der Anreicherung für Kommentare in öffentlichen Methodenrümpfen soll untersucht werden.

5.1.2. Quelltextinformationen, die für Anreicherung genutzt werden können

Bevor untersucht werden kann, welche Informationen zur Wissensanreicherung von Begriffen im Quelltext genutzt werden können, muss diskutiert werden, was im Sinne der worteinbettungsbasierten TLR als Wissen gilt, das diese unterstützen kann.

In der Arbeit von Hey et al. werden einem einzelnen Artefakt-Element wie einem Anforderungssatz oder einem Quelltextelement eine sogenannte Bag-of-Embeddings (BoE) zugeordnet. Diese ist eine Menge der statischen Worteinbettungen aller zu verarbeitenden Wörter des Artefakt-Elements. Solche Bags-of-Embeddings zweier verschiedener Artefakttypen werden dann über die Word-Mover's-Distance-Ähnlichkeitsmetrik einander zugeordnet und damit Zuordnungen zwischen den Artefakten bestimmt. Daraus folgt, dass als anzureichernde Information für Quelltextelemente ausschließlich weitere Wörter infrage kommen, die die Zuordnung der Bags-of-Embeddings zwischen Quelltextelementen und Anforderungen bezüglich Präzision und Überdeckung verbessern. Es sollen Falschzuordnungen aufgrund von Mehrdeutigkeiten verringert und das Fehlen von Zuordnungen aufgrund fehlender lexikalischer Übereinstimmung der Begriffe eingedämmt werden.

Bei worteinbettungsbasierter Anforderungsrückverfolgung, die wie FTLR statische Einbettungen nutzt, werden zwei Instanzen desselben Wortes mit unterschiedlichen Bedeutungen auf dieselben Vektoren abgebildet. Dadurch ist nicht immer gegeben, dass die Vektoren zweier Wörter, deren gemeinte Bedeutungen semantisch verwandt sind, eine geringe Vektordistanz zueinander haben. Falls nun wie in der Einleitung dieses Kapitels beschrieben Quelltextabschnitte Wörter beinhalten, die nicht in Anforderungstexten vorkommen, aber mit einzelnen Wörtern in Anforderungstexten dennoch semantisch verwandt sind, kann daher die Zuordnung der entsprechenden Bags-of-Embeddings misslingen. Um dieses Problem zu umgehen, bietet es sich an, Synonyme, Hyperonyme/Hyponyme, Meronyme/Holonyme und andere semantisch verwandte Wörter anzureichern. Dies hängt damit zusammen, dass für ein gegebenes mehrdeutiges Wort dessen semantisch verwandte Wörter wie in Abschnitt 2.1.6 beschrieben die gemeinte Bedeutung des fraglichen Wortes eingrenzen können. Übertragen auf die Bags-of-Embeddings können hinzugefügte semantisch verwandte Wörter die Word-Mover's-Distance in Problemfällen wie oben so beeinflussen, dass die Zuordnung der entsprechenden Bags-of-Embeddings wieder gelingt. Es ist also sinnvoll, als anzureicherndes Wissen Synonyme, Hyperonyme/Hyponyme, Meronyme/Holonyme und andere semantisch verwandte Wörter zu betrachten.

Abgesehen von Synonymbeziehungen sind übrige Verwandtschaftsbeziehungen nur sinnvoll auf Substantiven definiert. Das Anreicherungspotential von Wörtern, die keine Substantive sind, ist daher deutlich begrenzter als jenes von Substantiven. Daher sind nur Substantive für die Anreicherung geeignet. Die in Abschnitt 5.1.1 erwähnte Wortarterkennung soll also mit dem Ziel eingesetzt werden, Nicht-Substantive von der Anreicherung mit Wissen auszuschließen. Es ist anzumerken, dass für Wörter anderer Wortarten dennoch Worteinbettungen im FTLR-Projekt von Hey et al. berechnet werden; diese also nicht ignoriert werden.

Zu beantworten ist nun die Frage nach den Quelltextelementen, die als Quelle für die Wissensanreicherung infrage kommen. Wie schon in Abschnitt 5.1.1 ausgeführt, ist es sinnvoll, Schlüsselwörter der Programmiersprache, Sonderzeichen, Literale und Annotationen auszuschließen, da sie keine semantische Verwandtschaft zu anzureichernden Wörtern aufweisen bzw. als Quelle für Worteinbettungen nicht geeignet sind. All die Quelltextteile, die in Abschnitt 5.1.1 bereits als anzureichernde Elemente festgelegt worden sind, sind auch als Quellen für die Wissensanreicherung geeignet und sollen mit den erwähnten Schritten vorverarbeitet werden.

Klassen- und Instanzvariablen ohne Getter- oder Setter-Methoden können interne Informationen über den Zustand eines Objektes oder einer Klasse beinhalten. Die Möglichkeit einer semantischen Verwandtschaft zwischen Begriffen in deren Namen und Begriffen in den Namen ihrer übergeordneten Datentypen kann daher angenommen werden. Dies macht ihre Namen zu zusätzlichen Kandidaten für Wissensanreicherungsquellen für die Datentypen, denen sie untergeordnet

sind. Diese Namen müssen mit den zuvor beschriebenen Schritten vorverarbeitet werden, also tokenisiert, lemmatisiert, mit Wortarten annotiert, stoppwort- und abkürzungsbereinigt werden.

In manchen Fällen ist einer Klassen- oder Instanzvariablen auch ein Javadoc-Kommentar zugeordnet, der weitere Informationen bezüglich der gespeicherten Daten beinhalten kann. Auch dieser kann als Quelle betrachtet werden und wie übrige Javadoc-Kommentare vorverarbeitet genutzt werden.

Nicht-öffentliche Methoden werden nicht mit Wissen angereichert, können aber als Hilfsmethoden oder Methoden, die interne Zustandswechsel hervorrufen, Kontextinformation bieten, um für anzureichernde Begriffe in Methoden Bedeutungen aufzulösen. Ihre erweiterten Signaturen können also ebenfalls als Anreicherungsquellen betrachtet werden und entsprechend vorverarbeitet werden. Es ist anzunehmen, dass die Rümpfe nicht-öffentlicher Methoden anders als deren erweiterte Signaturen sich semantisch zu weit von dem öffentlichen Verhalten des umgebenden Datentyps entfernen. Genauso ist anzunehmen, dass nicht-öffentliche Rumpfanweisungen die Anwendungsdomäne nicht ausreichend abdecken und daher auch als Quelle für Wissensanreicherung nicht geeignet sind.

Die Rümpfe öffentlicher Methoden hingegen geben eine direkte Beschreibung des Verhaltens der nach außen hin sichtbaren Schnittstelle des umgebenden Datentyps. In ihnen können nicht-öffentliche Methoden aufgerufen und auf nicht-öffentliche Instanz- und Klassenvariablen zugegriffen werden. Es ist daher anzunehmen, dass die Namen aufgerufener Methoden und die Namen und Typnamen benutzter Variablen innerhalb der Rumpfanweisungen als Quelle für die Wissensanreicherung von erweiterten Methodensignaturen infrage kommen. An dieser Stelle soll daran erinnert werden, dass Schlüsselwörter (insbesondere diejenigen von Kontrollstrukturen) nicht als Quelle für die Anreicherung geeignet sind. Lediglich Namen von Bezeichnern, die die Anwendungsdomäne überdecken, sind für die Ziele dieser Arbeit als Quelle geeignet.

Mittels Quelltextanalyse lassen sich zudem weitere Informationen aus dem Quelltext gewinnen. Die Menge aller Methoden, die eine bestimmte Methode aufruft oder von ihr aufgerufen wird, kann ein semantisch verwandtes Aufgabengebiet überdecken. Genauso kann angenommen werden, dass sich Methoden, die auf die gleichen Daten zugreifen, semantisch zumindest überschneiden. Somit können Informationen in Abhängigkeitsgraphen ebenfalls als Anreicherungsquelle verwendet werden.

In den Abschnitten 5.2.2 und 5.3 wird u. a. näher analysiert werden, welche Quelltext-Informationen konkret für die verschiedenen Schritte der Wissensanreicherung verwendbar sind.

5.2. Wortbedeutungsauflösung

Hier soll analysiert werden, welche unterschiedlichen Verfahren für die TLR geeignet sind. Außerdem soll untersucht werden, wie ein geeigneter Wortkontext für aufzulösende Begriffe im Quelltext bestimmt werden kann.

5.2.1. WSD-Verfahren

WSD ist auf dem Gebiet der Verarbeitung natürlicher Sprache ein mittlerweile gut erforschtes Problem. Auf Quelltexteingaben wurde WSD allerdings bisher nicht angewendet. Dennoch sollen hier bereits erprobte Verfahren angewendet werden, da dafür lediglich die Bereitstellung entscheidender Kontextwörter essentiell ist. Dafür stehen wissensbasierte, lernbasierte und Hybrid-Verfahren zur Verfügung (siehe Abschnitt 4.1).

Domänenflexibilität

Um wissensbasierte Verfahren wie in Abschnitt 4.1.1 in unbekannten Domänen einzusetzen, muss die verwendete Wissensbasis ausgetauscht bzw. erweitert werden. Es müssen dabei domänenspezifische Bedeutungen und zugehörige Relationen identifiziert und eingetragen werden, wofür Domänenkenntnisse notwendig sind. Eine Veränderung der Implementierung ist bei wissensbasierten Verfahren nicht nötig.

Lernbasierte Verfahren wie in Abschnitt 4.1.2 können nur durch aufwendiges Neu-Trainieren an den Einsatz in neuen Domänen angepasst werden. Dafür sind große Mengen an (im überwachten Fall bedeutungsannotierten) Textquellen aus der neuen Domäne notwendig. Deren Verfügbarkeit kann nicht allgemein vorausgesetzt werden. Der Aufwand für die Erstellung und Annotation solcher Textkorpora ist vergleichbar mit dem Aufwand der manuellen TLR, was den Zweck dieser Arbeit verfehlen würde. Daher sind lernbasierte Verfahren nicht geeignet. Dies gilt ebenso für Hybridverfahren mit überwacht-lernbasierten Teilschritten wie EWISER in *„Breaking Through the 80% Glass Ceiling“* [BN20] (siehe Abschnitt 4.1.3).

Sowohl für wissensbasierte als auch für lernbasierte Verfahren sind also Anpassungen notwendig, falls sie auf neuen Domänen eingesetzt werden sollen. Allerdings sind wissensbasierte Verfahren diesbezüglich besser geeignet als lernbasierte Verfahren.

Wissensquellen

In bisherigen Arbeiten wurden als Quellen für die wissensbasierten Ansätze WordNet, BabelNet und DBpedia verwendet (siehe Abschnitt 4.1.1). Für WordNet wurden Bedeutungsknoten und semantische Relationen von Menschen definiert. Es handelt sich nicht um ein auf einem Korpus aufgebautes Wörterbuch; die grundlegende hierarchische Struktur entstammt der Intuition derjenigen, die WordNet erschaffen haben [Fel06]. Für DBpedia wurden Wikipedia-Artikel in Bedeutungsknoten umgewandelt und semantische Relationen über Daten aus Infoboxen und Kategorien sowie Verlinkungen zwischen den Artikeln generiert. Die Artikelmenge der Wikipedia entstand mit dem Ziel, konkrete Entitäten und reale Konzepte abzubilden. Via Crowdsourcing wurde zusätzlich für DBpedia händisch eine separate Ontologiestruktur aufgebaut, welche die Konzepte in einer Begriffshierarchie anordnet [Leh+15].

Das mehrsprachige BabelNet ist aus WordNet und Wikipedia sowie weiteren Quellen mittels automatischer Konsolidierung und maschineller Übersetzung entstanden. Für eine monolinguale Evaluation dieses Verfahrens wurden 1 000 zufällig ausgewählte Wikipedia-Artikel händisch ihren entsprechenden WordNet-Bedeutungsknoten zugeordnet, was 505 nichtleere Zuordnungen ergab. Im Abgleich mit diesen Musterlösungen ergab sich ein F_1-Bestwert von 77.7 %. Die Qualität der sprachübergreifenden Verknüpfungen wurde auf 600 Bedeutungsknoten in je fünf Sprachen evaluiert. Die Präzisionswerte für Zuordnungen mittels maschineller Übersetzung lagen für alle Sprachen zwischen 67.16 % und 75.58 %. Zuordnungen über Wikipedia-Verlinkungen erreichen Präzisionswerte zwischen 92.46 % und 99.09 % [NP12]. Für BabelNet muss also beachtet werden, dass die durch automatische Konsolidierung generierten semantischen Relationen stellenweise falsche Zuordnungen enthalten können.

Die unterschiedlichen Entstehungsgrundsätze von WordNet und DBpedia wirken sich auf deren Inhalt aus. Die hierarchische Struktur von WordNet wurde während der Entstehungszeit arbiträr festgelegt und später durch externe Zuschriften erweitert. Diese Annahme von Zuschriften ist momentan ausgesetzt, die aktuellste Version WordNet 3.1 wurde 2011 veröffentlicht [Prib]. DBpedia hingegen basiert auf der Wikipedia, die keine Hierarchie zur Grundlage hat, sondern ein Nebeneinander von Artikeln, welche untereinander Querverweise enthalten. Die Kategorisierung von Wikipedia-Artikeln ist mit der Hyperonym-Hierarchie von WordNet vergleichbar, jedoch keine strenge Hierarchie. Die Wikipedia wird ständig erweitert und Änderungen werden laufend in DBpedia übertragen [Leh+15].

Relation	Umkehrung
Antonym	
Hyperonym	Hyponym
Instanz-Hyperonym	Instanz-Hyponym
Mitglieds-Holonym	Mitglieds-Meronym
Substanz-Holonym	Substanz-Meronym
Bestandteils-Holonym	Bestandteils-Meronym
Attribut	
Abgeleitete Form	

Tabelle 5.1.: Auswahl semantischer Relationen zwischen Substantiv-Synsets in WordNet [Pria]

Bezeichner	Beschreibung
`dct:subject`	… hat als Thema …
`skos:broader`	Kategorie … hat als Oberkategorie …
`skos:subject`	… hat als Thema …
`skos:related`	… ist thematisch verwandt mit …
`rdf:type`	… ist vom Typ / gehört zur Klasse …
`dbo:class`	… gehört zur Klasse/Gruppe …
`gold:hypernym`	… hat als Oberbegriff …

Tabelle 5.2.: Auswahl semantischer Relationen in DBpedia nach [Leh+15], [Wei+19]

Durch diese Unterscheidung begründet sich die Tatsache, dass DBpedia in manchen Nischendomänen umfassendere Detailinformationen als WordNet enthält. Beispielsweise enthält DBpedia einen Eintrag für das Konzept der „Picardschen Terz“[3], wohingegen eine Suche in WordNet dazu keine Ergebnisse liefert[4] und eine Suche nach „third“ ebenfalls keine Ergebnisse für dieses spezifische Konzept liefert[5]. Genauso enthält DBpedia einen Eintrag für den „A*-Suchalgorithmus“[6], WordNet für die Suche nach „a star algorithm“[7] keine Ergebnisse und für „algorithm“[8] keine spezifischen Unterbegriffe für dieses Konzept.

Bezüglich Domänenüberdeckung sind also DBpedia und damit auch BabelNet gegenüber WordNet im Vorteil.

WordNet und BabelNet enthalten verschiedene Wortarten (Substantive, Verben etc.). DBpedia enthält konstruktionsbedingt nur Substantive und Eigennamen. Da laut Abschnitt 5.1.1 nur Substantive und Eigennamen angereichert werden sollen, ist dies für die Zwecke dieser Arbeit nicht von Nachteil.

Sowohl WordNet als auch DBpedia enthalten semantische Relationen zwischen ihren Bedeutungsknoten. WordNet ist als strenge Hierarchie aufgebaut, DBpedia nicht: In WordNet gibt es für Substantive eine Hyperonym-Relation, in DBpedia gibt es mehrere Kandidaten für Relationen, welche als Verallgemeinerungs-Relationen infrage kommen (siehe dazu Tabellen 5.1 und 5.2). Außerdem sind manche dieser Relationen im Gegensatz zu Hyperonym-Relationen in WordNet nicht rechtseindeutig.

[3] `https://dbpedia.org/page/Picardy_third` (abgerufen am 20.02.2022).
[4] `http://wordnetweb.princeton.edu/perl/webwn?s=picardy+third&sub=Search+WordNet&o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&h=000000000` (abgerufen am 20.02.2022).
[5] `http://wordnetweb.princeton.edu/perl/webwn?s=third&sub=Search+WordNet&o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&h=000000000` (abgerufen am 20.02.2022).
[6] `https://dbpedia.org/page/A*_search_algorithm` (abgerufen am 20.02.2022).
[7] `http://wordnetweb.princeton.edu/perl/webwn?s=a+star+algorithm&sub=Search+WordNet&o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&h=` (abgerufen am 20.02.2022).
[8] `http://wordnetweb.princeton.edu/perl/webwn?s=algorithm&sub=Search+WordNet&o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&h=` (abgerufen am 20.02.2022).

Beispiel 5.1: Semantische Relationen in WordNet und DBpedia

WordNet enthält folgende Relationen[a]:

$$\{\text{ rabbit, coney, cony }\} \xrightarrow{\text{hypernym}} \{\text{ leporid, leporid mammal }\}^{b}$$
$$\{\text{ leporid, leporid mammal }\} \xrightarrow{\text{hypernym}} \{\text{ mammal, mammalian }\}$$
$$\{\text{ mammal, mammalian }\} \xrightarrow{\text{hypernym}} \cdots \xrightarrow{\text{hypernym}} \{\text{ animal, …, creature, fauna }\}$$

DBpedia enthält dazu u. a. folgende Relationen[c] (als RDF-Tripel):

```
dbr:Rabbit   dbo:class       dbr:Mammal
dbr:Rabbit   dct:subject     dbc:Rabbits_and_hares
dbr:Rabbit   dct:subject     dbc:Herbivorous_mammals
dbr:Rabbit   gold:hypernym   dbr:Mammal
dbr:Rabbit   rdf:type        dbo:Mammal
dbr:Rabbit   rdf:type        dbo:Animal
```

[a] `http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&r=1&s=rabbit&i=4&h=1000100000` (abgerufen am 01.03.2022).
[b] „rabbits and hares“
[c] `https://dbpedia.org/resource/Rabbit` (abgerufen am 01.03.2022).

In Beispiel 5.1 werden diese Unterschiede anhand des Begriffs „rabbit“ gezeigt. In WordNet hat das Synset für „rabbit“ nur ein direktes Hyperonym, nämlich das Synset für „leporid mammal“. Dieses hat wiederum das Synset für „mammal“ als direktes Hyperonym. Geht man einige Hyperonym-Kanten im Graphen weiter, so erreicht man das Synset für „animal“. In DBpedia hingegen gibt es für den Begriff „mammal“ u. a. Wikipedia-Artikelknoten wie `dbr:Mammal` und Ontologie-Knoten wie `dbo:Mammal`, auf welchen unterschiedliche Relationen definiert sind. Für die Verallgemeinerung von „rabbit“ zu „mammal“ gibt es u. a. drei verschiedene Relationen (`dbo:class`, `gold:hypernym` und `rdf:type`). Außerdem hat `dbr:Rabbit` u. a. zwei `dct:subject`-Kanten zu den Wikipedia-Kategorien `dbc:Rabbbits_and_hares` und `dbc:Herbivorous_mammals`. Genauso gibt es u. a. zwei `rdf:type`-Kanten, eine zu `dbo:Mammal` und eine zu `dbo:Animal`.

Es gibt also in DBpedia kein direktes Äquivalent für Hyperonym-Beziehungen wie in WordNet, aber dafür mehrere Alternativen, die diesen Zweck erfüllen. Da die häufigste Relation in WordNet die Hyperonym-Relation ist [Pric] und damit die wichtigste Relation in Hinblick auf wissensbasierte WSD und es dafür in DBpedia entsprechenden Ersatz gibt, ist davon auszugehen, dass DBpedia ähnlich gut für WSD geeignet ist wie WordNet.

Anwendbarkeit

Für den Ansatz von Agirre et al. [ALS14] (siehe Abschnitt 4.1.1) ist eine Referenzimplementierung (UKB) quelloffen verfügbar. Außerdem kann das verwendete Wissensnetz ausgetauscht werden und somit z. B. auch DBpedia verwendet werden. Für Babelfy [MRN14] (siehe **??**) gibt es eine Online-Schnittstelle unter Verwendung von BabelNet. Um andere Wissensnetze zu verwenden, muss das Verfahren selbst implementiert werden.

Der Ansatz von Wang et al. [WWF20] (siehe Abschnitt 4.1.1) verwendet sowohl Textquellen, die aus Wikipedia entnommen wurden, als auch WordNet-Relationen für die Bedeutungsauflösung. Da wie zuvor beschrieben DBpedia oder BabelNet eine bessere Domänenüberdeckung als WordNet haben, müsste untersucht werden, inwieweit das Verfahren von WordNet auf andere Wissensnetze umgestellt werden kann. Das Zusammenspiel zwischen den unüberwacht erlernten Worteinbettungen basierend auf den Wikipedia-Textquellen und den WordNet-Relationen ist nicht ohne Weiteres auf den Einsatz anderer Wissensnetze übertragbar. Eine derartige Anpassung

Verfahren	Anmerkungen	F_1 (%) SemEval- 2013	F_1 (%) SemEval- 2015	Quelle
UKB: PPR_{w2w}	mit WordNet	68.8	70.3	[ALS18]
Babelfy	mit BabelNet $\leqslant$ 3.7 †	66.4	70.3	[RCN17]
	mit BabelNet 1.1.1	69.2	—	[MRN14]
	mit WordNet	65.9	—	[MRN14]
Wang et al.	versch. Textquellen, WordNet	68.4	72.3	[WWF20]
Hadiwinoto et al.	BERT, Gated-Linear-Unit	71.1	76.2	[HNG19]
EWISER	BERT, WordNet mit annot. Glossen	80.7	81.8	[BN20]

Tabelle 5.3.: Übersicht über die Ergebnisse der in Abschnitt 4.1 vorgestellten Arbeiten.
† In der Quelle war keine Version angegeben, lediglich ein Verweis auf die API unter `http://babelfy.org`. Die bei Veröffentlichung der Quelle aktuelle Version war 3.7 [Bab].

und Umsetzung des Ansatzes von Wang et al. würde aufgrund des Umfangs den zeitlichen Rahmen dieser Arbeit sprengen.

Für die Ansätze von Hadiwinoto et al. [HNG19] (siehe Abschnitt 4.1.2) und Bevilacqua et al. [BN20] (siehe Abschnitt 4.1.3) müssen für unbekannte Domänen umfangreiche Textquellen annotiert und damit die Sprachmodelle trainiert werden. Dies ist im Vergleich zu den wissensbasierten Verfahren von UKB und Babelfy ein deutlich größerer Aufwand. In der Praxis kann nicht vorausgesetzt werden, derartige Textquellen ausreichenden Umfanges für beliebige Projekte unbekannter Domänen zur Verfügung zu haben.

Die lernbasierten Ansätze sind also für die Ziele dieser Arbeit ungeeignet bzw. zu aufwendig in der Anpassung auf unbekannte Domänen. Die wissensbasierten Ansätze von UKB und Babelfy sind hingegen gut anwendbar.

Qualität der Ergebnisse

Für eine Untersuchung der erreichten F_1-Werte der unterschiedlichen Verfahren ist zu beachten, dass die hier aufgeführten Werte auf den SemEval-Benchmark-Korpora bestimmt wurden. Der SemEval-2013-Korpus besteht aus dreizehn Artikeln aus verschiedenen Themenbereichen wie Sportberichterstattung oder Börsennachrichten [NJV13]. Der SemEval-2015-Korpus besteht aus zwei medizinischen Fachdokumenten über Arzneimittel, einem Handbuch zur Bedienung einer Mathematik-Software und einer Erörterung über verschiedene soziale Fragestellungen [MN15]. Die so erreichten Ergebnisse sind nicht zwangsläufig repräsentativ für das Verhalten der vorgestellten Verfahren auf Quelltexteingaben, die beliebige Domänen abdecken können.

Wie in Tabelle 5.3 zu sehen ist, erreichen lernbasierte und Hybrid-Verfahren (von Hadiwinoto et al. [HNG19] und EWISER [BN20]) höhere F_1-Werte als wissensbasierte. EWISER liegt dabei vor dem Ansatz von Hadiwinoto et al. Unter den wissensbasierten Ansätzen erreicht der Ansatz von Wang et al. [WWF20] den höchsten F_1-Wert über alle angegebenen Korpora. UKB [ALS14] und Babelfy [MRN14] liegen ungefähr gleichauf, je nach für die Evaluation verwendeten Datenquellen ergeben sich Unterschiede für SemEval-2013.

Die wissensbasierten Verfahren liegen 10–15 F_1-Prozentpunkte unter den lernbasierten Verfahren. Somit sprechen die F_1-Werte allein zwar für die lernbasierten Verfahren von Hadiwinoto et al. und EWISER. Allerdings sind die lernbasierten Verfahren wie zuvor erörtert nicht anwendbar. Die wissensbasierten Verfahren sind anwendbar und erreichen mit F_1-Werten zwischen 65 und 72 Prozent akzeptable Ergebnisse. Es ist deshalb im Sinne der TLR nicht lohnenswert, auf die größere Domänenflexibilität der wissensbasierten Verfahren aufgrund des qualitativen Vorsprungs der lernbasierten Verfahren zu verzichten.

Zusammenfassung

Für die Verbesserung worteinbettungsbasierter TLR empfehlen sich für WSD die wissensbasierten Verfahren von UKB und Babelfy. Sie sind im Gegensatz zu lernbasierten Verfahren domänenflexibel. Zudem sind sie im Gegensatz zu lernbasierten Verfahren oder dem Ansatz von Wang et al. [WWF20] im Sinne der Ziele dieser Arbeit anwendbar. Als Wissensquellen eignen sich DBpedia und BabelNet. Diese weisen gegenüber WordNet Vorteile bezüglich Domänenüberdeckung und Aktualität auf.

5.2.2. Wahl eines geeigneten Wortkontextes

Wie bereits erklärt, können Wortbedeutungen nur mithilfe von Kontext aufgelöst werden. In verwandten Arbeiten zu WSD auf natürlichsprachlichen Eingaben werden beispielsweise um die Wörter eines einzelnen Satzes aufzulösen zusätzlich der vorangehende und der nachfolgende Satz als Kontext bereitgestellt [ALS18]. Da Quelltexte nicht nur linear formuliert werden (bspw. als eine Folge von Anweisungen in einer Methode), sondern auch hierarchisch gegliedert sind (Pakete enthalten Datentypen, Datentypen enthalten Methoden, Methoden enthalten Verzweigungen etc.), ergeben sich deutlich vielfältigere Möglichkeiten, einen Kontextrahmen für im Quelltext vorkommende Wörter festzulegen. Dies soll im Folgenden analysiert werden. Da die Lauffähigkeit des Quelltextes nicht vorausgesetzt werden kann, aber auch nicht-lauffähige Quelltexte mit Wissen angereichert werden sollen, werden in diesem Abschnitt ausschließlich statische Quelltextanalyseverfahren verwendet.

5.2.2.1. Auflösungseinheiten

In Abschnitt 5.1.1 wurde bereits festgestellt, dass Datentypen und Methoden anzureichern sind. Die Anreicherung erfolgt im Falle von Datentypen für ihre Namen und ihre Javadoc-Kommentare, im Falle von öffentlichen Methoden für ihre Namen, Parameter, Javadoc-Kommentare und ggf. Rumpfkommentare. Für andere Quelltextelemente ist anzunehmen, dass eine Anreicherung nicht lohnenswert ist. Somit sollen auch nur Wörter in den erwähnten Quelltextelementen aufgelöst werden.

Die aufzulösenden Quelltextelemente können in sogenannten *Auflösungseinheiten* zusammengefasst werden. Für Datentypen bilden Datentypnamen und ihre Javadc-Kommentare eine Auflösungseinheit. Für Methoden bilden Methodennamen, Parametertypen und -namen, Javadoc-Kommentare und ggf. Rumpfkommentare eine Auflösungseinheit. Alle Elemente einer Auflösungseinheit sind als Kontextquellen für die übrigen Elemente geeignet, da sie gemeinsam einem Hauptelement untergeordnet sind und semantisch verwandten Aufgabenbereichen zugeordnet werden können. So beschreiben Javadoc-Kommentare von Datentypen oder Methoden deren Einsatzzweck und Verhalten. Methodenparameter zeigen, welche Informationen beim Aufruf einer Methode von außen bereitgestellt werden, und geben damit ebenfalls Einblick in deren Verhalten. Rumpfkommentare können genauso einzelne Aspekte des Verhaltens einer Methode und ihres Zusammenspiels mit der Außenwelt beschreiben.

Es besteht die Möglichkeit, Auflösungseinheiten zusammen aufzulösen oder sie weiter aufzuteilen bzw. getrennt aufzulösen. Da die Elemente von Auflösungseinheiten als Kontextquellen füreinander geeignet sind, empfiehlt sich die gemeinsame Auflösung.

Als weitere mögliche Kontextquellen für die Auflösungseinheiten sollen im Folgenden Begriffe in den Bezeichnern der in Abschnitt 5.1.2 festgelegten Quelltextelemente untersucht werden.

5.2.2.2. Deklarativer Kontext

Aus der hierarchischen Gliederung von Quelltexten ergibt sich das naheliegende Vorgehen, zu einem Quelltextelement andere Quelltextelemente, die mit ihm gemeinsam im selben Gültigkeitsbereich deklariert worden sind, als Kontext zu betrachten. Dies soll der *deklarative Kontext* eines Quelltextelementes genannt werden.

Beispiel 5.2: Deklarativer Kontext von Methoden

Für die Methode `BankAccount:withdraw(double)` soll der deklarative Kontext angegeben werden. Nicht-öffentliche Elemente sollen ausgeschlossen werden.

```
1 class BankAccount {
2     private double balance;
3     public void withdraw(double amount) {
4         this.balance -= amount;
5     }
6     public void deposit(double amount) {
7         this.balance += amount;
8     }
9 }
```

Der deklarative Kontext der Methode umfasst die Klasse `BankAccount`, in der sie deklariert ist, und die Methode `deposit`. Falls der Rumpf der Methode hinzugenommen werden soll, so kommen noch `this.balance` und `amount` dazu.

Zum deklarativen Kontext einer Methode gehören also der umgebenden Datentyp, Klassen- und Instanzvariablen des umgebenden Datentyps sowie Methoden, mit denen zusammen die fragliche Methode deklariert ist. Zusätzlich kann der Rumpf einer Methode ebenfalls als deklarativer Kontext gewertet werden. Begriffe, die in Methodenrümpfen vorkommen, können in manchen Fällen weniger geeignet sein, da sie zu Implementierungsdetails gehören. Es ist zu untersuchen, inwieweit die Hinzunahme von Rumpfbegriffen die WSD von aufzulösenden Begriffen in Methodendeklarationen verbessert. Beispiel 5.2 zeigt den deklarativen Kontext einer Methode.

Der deklarative Kontext eines Datentyps sind analog die Elemente, die ihm untergeordnet deklariert werden, wie Methoden, Klassen- und Instanzvariablen sowie innere Typen. Dabei können nicht-öffentliche Quelltextelemente ausgeschlossen oder hinzugenommen werden. Es ist zu untersuchen, inwieweit die Sichtbarkeit dieser Elemente ihre Eignung als Kontextquelle beeinflusst. Vermutlich sind öffentliche Methoden als Kontext für die WSD von aufzulösenden Begriffen in Datentypdefinitionen besser geeignet als nicht-öffentliche Elemente. In der Arbeit *„Automatic generation of natural language summaries for Java classes“* [Mor+13] (siehe Abschnitt 4.2) werden natürlichsprachliche Beschreibungen für Klassen generiert. Dafür werden unter anderem innere Klassen und die Methoden der Klasse verwendet. Solche Klassenbeschreibungen sollen den Zweck und das Aufgabenfeld eines Datentyps vermitteln. Da dies der Bedeutung das Datentyps fürs Programmgeschehen ähnelt, kann angenommen werden, dass die verwendeten Quellen für die Beschreibungsgenerierung auch gute Kontextquellen für die WSD darstellen.

Es kann vorkommen, dass einem Datentyp mehrere Aufgabenbereiche zugeordnet sind. In diesem Fall kann angenommen werden, dass für Methoden, die deutlich spezifischer nur einem Aufgabenbereich zugeordnet sind, der deklarative Kontext weniger für die Bedeutungsauflösung von Methodenbegriffen geeignet ist als andere Kontextarten.

5.2.2.3. Situativer Kontext

Eine Alternative zum deklarativen Kontext ist die Idee, für die aufzulösenden Begriffe einer Methodendeklaration die Quelltextstellen zu finden, in denen die jeweilige Methode aufgerufen wird. Als Kontext sollen dann Quelltextelemente in Anweisungen um die Aufrufanweisung herum betrachtet werden. Dies soll der *situative Kontext* einer Methode genannt werden. Dazu gehören also die Namen und Typen von lokalen Variablen, die Namen von verwendeten Instanz- und Klassenvariablen und die Namen und Parameternamen von aufgerufenen Methoden. Es besteht die Möglichkeit, den jeweiligen Kontext auf eine feste Anzahl Zeilen um die Aufrufanweisung herum

Beispiel 5.3: Situativer Kontext von Methoden

```
1  {
2      patient = getCurrentPatient();
3      if (patient.getPrescriptionCount() > 0) {
4          printInsuranceReceipt();
5          showPrescriptionAlert();
6      } else {
7          promptPatientReferral();
8      }
9      finalizeAppointment();
10 }
```

Situativer Kontext von `printInsuranceReceipt` ist `patient`, `getCurrentPatient`, `patient::getPrescriptionCount`, `showPrescriptionAlert`, `promptPatientReferral` und `finalizeAppointment`. Beschränkt man diesen auf eine Zeile vor und nach dem Aufruf von `printInsuranceReceipt`, so bleiben lediglich `patient::getPrescriptionCount` und `showPrescriptionAlert` übrig. Lässt man hingegen beim situativen Kontext ausschließlich Methodenaufrufe zu, so gleicht der situative Kontext von `printInsuranceReceipt` der Menge ihrer Aufrufgeschwister.
Der verzweigungsabhängige situative Kontext von `printInsuranceReceipt` gleicht dem situativen Kontext ohne `promptPatientReferral`, da diese anders als `printInsuranceReceipt` im `else`-Zweig aufgerufen wird. Der verzweigungsabhängige situative Kontext von `promptPatientReferral` enthält dementsprechend `patient`, `getCurrentPatient`, `patient::getPrescriptionCount` und `finalizeAppointment`.

zu beschränken (sog. *zeilenbeschränkter situativer Kontext*). Falls eine solche Zeileneinschränkung zu wenig Kontextwörter liefert, können weitere Zeilen hinzugezählt werden, bis ein festzulegender Schwellenwert an Kontextwörtern erreicht ist. Zusätzlich besteht die Möglichkeit, den Kontext lediglich auf Methodenaufrufe oder auf Aufrufe öffentlicher Methoden einzuschränken, falls die Namen verwendeter Variablen oder nicht-öffentlicher Methoden zu viel Informations-Streuung bewirken. Der *verzweigungsabhängige situative Kontext* beachtet zudem, ob die fragliche Methode innerhalb einer Verzweigung aufgerufen wird und schließt Anweisungen aus anderen Ausführungszweigen als dem aktuellen aus. Beispiel 5.3 zeigt verschiedene Varianten für den situativen Kontext einer Methode.

Es ist möglich, dass obige Definitionen für eine einzelne Methode mehrere situative Kontexte liefern aufgrund verschiedener Aufrufstellen sowie verschiedener Ausführungszweige. Um mehrere Kontexte für die WSD von Begriffen in Methodendeklarationen zu nutzen, gibt es mehrere Möglichkeiten: Zum einen können die Mengen aller Wörter in den Kontexten zu einer großen Kontextmenge vereinigt werden, mit welcher dann die Bedeutungen aufgelöst werden. Zum anderen kann jeder einzelne Kontext nacheinander für die Bedeutungsauflösung derselben Begriffe verwendet werden und anschließend für jedes aufzulösende Wort die am häufigsten vorkommende Bedeutung gewählt werden.

Weiter ist anzunehmen, dass der situative Kontext einer Methode umso aussagekräftiger für die Bedeutungen der aufzulösenden Wörter einer Methodendeklaration wird, je größer der Aufrufstapel des Aufrufs der fraglichen Methode ist. Diese Annahme kann dadurch begründet werden, dass Methodenaufrufe innerhalb flacher Aufrufhierarchie deutlich grobgliedrigere Logik ausführen und somit mehrere Anwendungsdomänen überdecken können. Je tiefer innerhalb der Aufrufhierarchie ein Methodenaufruf stattfindet, desto feingliedriger und domänenspezifischer wird vermutlich der situative Kontext sein, was der Bedeutungsauflösung zugutekommt. Da die tatsäch-

Beispiel 5.4: Analytischer Kontext von Methoden

Wir nehmen an, dass folgende Abhängigkeiten vorliegen:

Linke Seite	Art	Rechte Seite
`GameEngine:update()`	ruft auf	`Physics:processCollisions()`
`Physics:processCollisions()`	ruft auf	`Entity:getAbsoluteBoundingBox()`
`Physics:processCollisions()`	ruft auf	`Player:reactToCollision(Entity)`
`Physics:processCollisions()`	greift zu auf	`Physics.collisionList`
`GameEngine.processSound()`	greift zu auf	`Physics.collisionList`

Der analytische Kontext der Methode `Physics:processCollisions()` besteht aus den Methoden `GameEngine:update()` (von welcher sie aufgerufen wird), `Entity:getAbsoluteBoundingBox()` und `Player:reactToCollision(Entity)` (welche sie selbst aufruft) sowie `GameEngine:processSound()` (zu welcher sie datenabhängig ist).

liche Aufrufstapeltiefe lediglich zur Laufzeit vorliegt, muss hierfür eine untere Schranke mittels Aufrufgraph-Traversierung ermittelt werden. Verschiedene situative Kontexte können dann desto stärker gewichtet werden, je größer die untere Schranke für die Aufrufstapeltiefe ist. Es ist zu beachten, dass der Aufrufgraph nicht zusammenhängend sein muss und Methoden vom Hauptausführungsstrang isoliert sein können, wie es oft bei Benutzeroberflächen-Ereignisbehandlung der Fall ist.

Analog zum situativen Kontext einer Methode lässt sich auch der situative Kontext eines Datentyps definieren als die Quelltextelemente der Anweisungen um die Verwendung dieses Datentyps herum. Als Verwendung eines Datentyps zählen die Deklaration lokaler Variablen mit diesem Typ oder das Aufrufen von Instanz- oder Klassenmethoden auf diesem Typ. Dies würde allerdings eine große Zahl an Verwendungsstellen generieren. Es ist anzunehmen, dass stellenweise nur einzelne Methoden einer Klasse genutzt werden und diese meist eigenen, spezifischeren Aufgabenbereichen zuzuordnen sind, als der zugehörige Datentyp allgemein abdeckt. Dementsprechend wären auch die Anweisungen um die Datentypverwendung herum meist deutlich spezifischeren Aufgabenbereichen zuzuordnen. Aufgrund der diesbezüglichen Gefahr von zu breiter Informationsstreuung ist die Nutzung der situativen Kontexte eines Datentyps vermutlich weniger geeignet zur WSD von aufzulösenden Begriffen einer Datentypdefinition.

5.2.2.4. Analytischer Kontext

Eine erweiterte Variante des situativen Kontextes einer Methode umfasst nicht nur die Quelltextzeilen um einen Aufruf der fraglichen Methode, sondern konzentriert sich auf jene Methoden selbst, in deren Rumpf die Methode aufgerufen wird. Die Menge aller Methoden, die eine bestimmte Methode aufrufen, kann so ebenfalls als Kontext der fraglichen Methode betrachtet werden und deren erweiterte Signaturen als Wortquellen genutzt werden. In der umgekehrten Richtung können ebenso alle Methoden betrachtet werden, die von einer bestimmten Methode selbst aufgerufen werden. Anders ausgedrückt werden statt Aufrufgeschwistern nun Aufrufabhängigkeiten betrachtet.

Genauso kann der situative Kontext erweitert werden auf die Betrachtung derjenigen Methoden, mit denen sich eine bestimmte Methode Zugriffe auf gemeinsam genutzte Daten teilt. Solche Datenabhängigkeiten können analog zu Aufrufabhängigkeiten zur Kontextbildung genutzt werden. Beide Varianten haben gemeinsam, dass ihre Kontextinformationen mittels Quelltextanalyse aus Abhängigkeitsgraphen abgelesen werden können. Daher soll eine solche Kontextvariante als *analytischer Kontext* bezeichnet werden. Beispiel 5.4 die verschiedenen Möglichkeiten von analytischem Kontext einer Methode.

In der Arbeit *„Automatic Source Code Summarization of Context for Java Methods“* [MM16] (siehe Abschnitt 4.2) werden natürlichsprachliche Beschreibungen von Methoden generiert, wobei Aufrufabhängigkeiten miteinbezogen werden. Mittels PageRank werden dafür die wichtigsten aufrufenden und aufgerufenen Methoden ermittelt. Natürlichsprachliche Methodenbeschreibungen haben zum Zweck, entwickelnden Menschen die Absichten bzw. das Verhalten einer Methode zu vermitteln. Da dies ähnlich zur Bedeutung eines Methodenaufrufs fürs Programmgeschehen ist, ist anzunehmen, dass die Quellen, die für die Generierung von Methodenbeschreibungen verwendet werden, auch als Kontextquellen für WSD von Methodenbegriffen geeignet sind. Somit sind Aufrufabhängigkeiten als potentiell geeignete Kontextwahl zu betrachten. Ferner kann der in der Arbeit vorgestellte Ansatz insofern übertragen werden, als dass eine Einschränkung auf PageRank-maximale Kontextmethoden die WSD-Ergebnisse verbessern könnte.

In der Arbeit *„Do data dependencies in source code complement call dependencies for understanding requirements traceability?“* [Kua+12] (siehe Abschnitt 4.2) werden Datenabhängigkeiten verwendet, um die TLR zu verbessern. Gegenüber bloßem Einsatz von Aufrufabhängigkeiten konnte der Einfluss von Datenabhängigkeiten die Fehlerrate um 1.6 % senken. Analog zu Aufrufabhängigkeiten ist davon auszugehen, dass die Berücksichtigung Datenabhängigkeiten für die WSD von Methodenbegriffen verbessern kann. Allerdings verwenden Kuang et al. in ihrer Arbeit Laufzeitanalysetechniken zur Bestimmung der Datenabhängigkeiten. Es ist daher zu untersuchen, inwiefern statisch bestimmte Datenabhängigkeiten die WSD von Begriffen in Methodendeklarationen verbessern können.

In der Arbeit *„Automatic generation of natural language summaries for Java classes“* [Mor+13] (siehe Abschnitt 4.2) werden natürlichsprachliche Beschreibungen für Klassen generiert. Dafür werden neben den zuvor erwähnten Quellen zusätzlich die implementierten Schnittstellen und Oberklassen einer Klasse verwendet. Daraus lässt sich der analytische Kontext eines Datentyps herleiten. Zu diesem gehören vererbungsabhängige Typen, also Obertypen und Untertypen des Datentyps. Es ist anzunehmen, dass in Obertypen vermehrt Oberbegriffe vorkommen und in Untertypen eher Unterbegriffe. Beispielsweise kann eine Klasse `ShapeTransformer` die Klassen `RectangleTransformer` und `CircleTransformer` als Untertypen haben, wobei *„rectangle“* und *„circle“* Unterbegriffe von *„shape“* sind. Für die wissensbasierte WSD ist anzunehmen, dass sowohl Ober- als auch Unterbegriffe als Kontext geeignet sind, da wissensbasierte WSD wie das Verfahren von Agirre et al. [ALS14] (siehe Abschnitt 4.1.1) auf ungerichteten Graphen arbeitet und damit die Relationsrichtung irrelevant ist.

In der Arbeit *„Facilitating program comprehension with call graph multilevel hierarchical abstractions“* [AGL21] (siehe Abschnitt 4.2) werden Aufrufabhängigkeiten zwischen Klassen definiert. Eine Klasse ist dabei aufrufabhängig von einer anderen, sofern die eine Klasse Methoden der anderen Klasse aufruft. Übertragen auf die Aufgabenstellung können Aufrufabhängigkeiten zwischen Datentypen dem analytischen Kontext eines Datentyps zugeordnet werden. Es ist zu untersuchen, inwieweit im analytischen Kontext eines Datentyps Aufrufabhängigkeiten im Vergleich zu Vererbungsabhängigkeiten die WSD von Begriffen in Datentypdefinitionen beeinflussen.

5.2.2.5. Vergleich der Kontextarten

Die vorgestellten Kontextarten können die Bedeutungsauflösung unterschiedlich beeinflussen. Die Anzahl und Relevanz der Wörter, die sie umfassen, kann variieren. Darauf soll im Folgenden kurz eingegangen werden.

So umfasst der deklarative Kontext eines Elements unter Einbeziehung nicht-öffentlicher Elemente vermutlich deutlich mehr Wörter als der deklarative Kontext unter Ausschluss nicht-öffentlicher Elemente. Gleichzeitig könnten dabei viele irrelevante Wörter, die sich auf Implementierungsdetails beziehen, das Ergebnis der Bedeutungsauflösung verschlechtern.

Der situative Kontext einer Methode kann ohne Filterung nach Elementart ähnlich viele Wörter liefern. Dabei können die Wörter wenig Sinnverwandtschaft mit den Wörtern der fraglichen

Methode haben, sofern der Einfluss der fraglichen Methode in ihren Aufrufkontexten eher gering ist und das Aufgabenfeld der sie aufrufenden Methoden semantisch deutlich entfernt ist von ihrem eigenen. Es ist anzunehmen, dass die Relevanz der Wörter eines situativen Kontextes steigt, je kleiner der Rumpf der aufrufenden Methode ist, da dann die Rolle der aufgerufenen Methode umso wichtiger für den Aufrufer ist. Sich beim situativen Kontext lediglich auf Aufrufer mit geringer Anweisungsanzahl zu beschränken, birgt allerdings die Gefahr, zu wenig Kontextwörter für die WSD zur Verfügung zu haben.

Der analytische Kontext einer Methode liefert vermutlich nur wenige Wörter, da nur die unmittelbaren aufrufenden und aufgerufenen Methoden miteinbezogen werden. Noch weniger dürfte die Wortanzahl beim analytischen Kontext von Datentypen sein, da die Anzahl der Untertypen vermutlich recht gering und die Anzahl der Obertypen noch geringer ist, da in Java keine Mehrfachvererbung von Klassen möglich ist. Auch hält sich die Anzahl implementierter Schnittstellen eines Datentyps vermutlich eher in Grenzen. Sich allein auf den analytischen Kontext zu beschränken, könnte ebenfalls zu wenig Kontextwörter für eine erfolgreiche Bedeutungsauflösung bereitstellen. Ähnlich wie beim situativen Kontext einer Methode kann die Relevanz der Kontextmethoden zunehmen, je kleiner ihr Rumpf ist. Dies ist begrenzt auf die implementierten Schnittstellen eines Datentyps übertragbar: Je mehr Methoden eine Schnittstelle vorschreibt, desto größer ist vermutlich ihr Einfluss auf die Gestalt des implementierenden Datentyps.

Um nicht zu riskieren, zu wenig Kontextwörter zur Verfügung zu haben, kann es notwendig sein, verschiedene Kontexte zu kombinieren. Dies kann schwellenwertgesteuert passieren, indem zunächst alle Wörter der einen Kontextart eingesammelt werden und im Falle der Unterschreitung des Schwellenwerts noch Wörter aus anderen Kontextarten hinzugenommen werden. Bei der Kombination könnten verschiedene Kontextarten unterschiedlich gewichtet werden, um deren Einfluss zu steuern. Als Gewichte kommen beispielsweise die oben diskutierte Relevanz der enthaltenen Wörter infrage. Dazu könnten die Wörter, die bereits in einer Auflösungseinheit vorkommen, stärker gewichtet werden als hinzugefügte Kontextwörter.

Eine weitergehende Untersuchung dieser Thematik soll im beschränkten Rahmen dieser Arbeit ausbleiben.

5.2.2.6. Zusammenfassung

Zusammengefasst ist die Eignung situativer, deklarativer und analytischer Kontexte für die WSD von aufzulösenden Begriffen im Quelltext zu untersuchen. Es wurden verschiedene Varianten dieser Kontextdefinitionen vorgestellt, die unterschiedlichen Einfluss auf die Ergebnisse der WSD haben können.

5.3. Weitere Wissensanreicherung

Für die Wissensanreicherung über WSD hinaus müssen zunächst mögliche Informationsquellen für die Anreicherung betrachtet werden. Neben dem Quelltext als offensichtlicher Quelle können im Falle der Anwendung von wissensbasierter WSD das dafür verwendete Wissensnetz in Betracht gezogen werden. Andere Wissensnetze als bereits verwendete bringen den Nachteil mit sich, dass eine Zuordnung von Bedeutungsknoten zwischen den Wissensnetzen benötigt wird, was nicht immer gegeben ist. Dies trifft prinzipiell auch auf nicht-wissensbasierte Informationsquellen wie lernbasierte Sprachmodelle zu.

Die nicht-wissensbasierte Anreicherung von Wissen (bspw. über lernbasierte Sprachmodelle) ist für eine Anbindung ans FTLR-Projekt auch deshalb nicht zu empfehlen, da FTLR bereits Worteinbettungen und Vektorabstände nutzt und daher wissensbasierte Anreicherung einen Mehrwert bringen kann. Zudem ist eine vektordistanzbasierte Anreicherung anfällig für Ausreißer, wenn beispielsweise zu einer Menge von Worteinbettungen der Clustermittelpunkt als angereicherte

Einbettung verwendet werden soll. Ausreißer können den Clustermittelpunkt deutlich verschieben und somit die angereicherten Bedeutungen verzerren. Darüber hinaus ist es stark vom verwendeten Sprachmodell abhängig, ob zwei beliebige Vektoren mit geringer Distanz zueinander immer ausreichend bedeutungsverwandt sind, um eine vektordistanzbasierte Anreicherung zu ermöglichen. Falls nämlich wie im Falle von FTLR statische Worteinbettungen genutzt werden, so könnte der Vektor eines mehrdeutigen Wortes in der Nachbarschaft von Vektoren liegen, die semantisch nur wenig mit der gemeinten Bedeutung des Wortes verwandt sind.

5.3.1. Verfahren

Im Folgenden sollen verschiedene Ansätze vorgestellt werden, weiteres Wissen ohne bzw. mit externen Wissensquellen anzureichern.

Ohne externe Wissensquelle

Ein mögliches Verfahren, um Quelltextelemente mit Begriffen ausschließlich aus dem Quelltext anzureichern, wird in der Arbeit *„Improving topic model source code summarization“* [McB+14] (siehe Abschnitt 4.2) vorgestellt. Solche Ansätze haben den Vorteil, dass ihre Domänenflexibilität nicht durch externe Datenquellen, die nicht alle Domänen ausreichend abdecken, eingeschränkt wird. Gerade bezüglich TLR allerdings können in Anforderungsdokumenten und im Quelltext jeweils unterschiedliche Wörter und/oder Formulierungen gewählt werden, die jedoch semantisch verwandt sind und damit externes Wissen für deren Zuordnung notwendig machen. Falls bspw. in Anforderungen ausschließlich *„display“* als Begriff für die Benutzeroberfläche verwendet wird, im Quelltext allerdings dafür ausschließlich *„user interface“* genutzt wird, fehlt die Zuordnung zwischen diesen beiden semantisch äquivalenten Begriffen, da eine solche nicht aus dem Quelltext allein abgeleitet werden kann. Wenn nun in Anforderungen ebenfalls der Begriff *„interface“* in einer völlig sinnfremden Bedeutung genutzt wird, könnten die Quelltext-Vorkommen von *„user interface“* den Anforderungs-Vorkommen von *„interface“* zugeordnet werden anstatt den Vorkommen von *„display“*. Daher sind solche Ansätze als alleinige Wissensanreicherung nicht empfehlenswert. Als Vorverarbeitungsschritt für wissensbasierte Anreicherung kommt dieses Verfahren allerdings durchaus infrage. So könnte mit dem in der Arbeit beschriebenen HDTM-Verfahren die resultierende Zuordnung von Begriffen zu Quelltextelementen als Ausgangspunkt für die weitere Anreicherung genutzt werden.

Mit externer Wissensquelle

Die Arbeit *„Context Model Acquisition from Spoken Utterances“* [WHT17] (siehe Abschnitt 4.3) beschreibt einen Ansatz, der zu zwei gegebenen Begriffen den Lowest Common Subsumer (LCS) dieser Begriffe als Elternknoten hinzufügt, sofern diese Begriffe bezüglich der Lin-Ähnlichkeitsmetrik (siehe Abschnitt 2.4.2) eine Schranke von 0.7 unterschreiten. Dies wird solange wiederholt, bis keine neuen Knoten mehr hinzugefügt werden können. Dieser naheliegende Ansatz benötigt eine kompatible Wissensquelle, welche für einzelne Begriffe eindeutige Oberbegriffe definiert. Zudem setzt die Verwendung der Lin-Ähnlichkeitsmetrik voraus, dass die gemeinsame Auftrittswahrscheinlichkeit zweier Begriffe kleiner oder gleich der Auftrittswahrscheinlichkeit deren LCS ist [Lin98]. Für WordNet existieren entsprechende Knotengewichtungen, die dieses Kriterium erfüllen. Für DBpedia und abgeleitete Wissensnetze müssen die Auftrittswahrscheinlichkeiten approximiert werden, bspw. über die Häufigkeit der eingehenden Artikelverweise. Dabei kann die Einhaltung der obigen Ungleichung für Knoten höher in der Hierarchie nicht garantiert werden. Die Auftrittswahrscheinlichkeiten müssen also für DBpedia von den Blättern eines Oberbegriffbaumes aufwärts neu berechnet werden. Eine Untersuchung diesbezüglicher Möglichkeiten ist im zeitlichen Rahmen dieser Arbeit nicht möglich. Für DBpedia besteht zusätzlich das Problem, dass es keine eindeutige Oberbegriffsrelation gibt, sondern stattdessen mehrere infrage kommende Relationen aus mehreren Datensätzen mit unterschiedlicher Abdeckung gegeben sind (siehe

Abschnitt 5.2.1). Ein Vergleich der Eignung verschiedener Oberbegriffsrelationen kann im Rahmen dieser Arbeit ebenfalls nicht erfolgen. Der Ansatz nach Weigelt et al. eignet sich also generell nicht.

Eine ebenfalls intuitive Lösung wäre die Anreicherung einer festen kleinen Zahl von semantisch verwandten Begriffen für jeden Begriff. Geeignet wären neben Synonymen auch Hyperonyme oder Holonyme, da diese die Bedeutungen der Ausgangsbegriffe nicht einschränken, sondern höchstens erweitern. Hyponyme und Meronyme können ebenfalls nützlich sein, falls beispielsweise in Anforderungen ein Quelltextbegriff nicht vorkommt, dafür aber stattdessen ausschließlich ein Meronym oder Hyponym des Begriffs verwendet wird. Allerdings kann die Anreicherung von Hyponymen oder Meronymen die Bedeutungen der Ausgangsbegriffe zu spezifisch verzerren. Während bei Hyperonymen und Holonymen beispielsweise ein bis zwei davon zur Anreicherung ausreichen dürften, ohne die Ausgangsbedeutungen zu sehr zu verzerren, könnte die Anreicherung von Meronymen und Hyponymen eine möglichst vollständige Abdeckung aller hierarchisch tiefer liegenden Begriffe notwendig machen, um ein möglichst breites Feld an Unterfacetten eines Begriffs abzudecken. Dieser Aspekt ist vor allem relevant, falls die angereicherten Wörter den jeweiligen Bags-of-Embeddings hinzugefügt werden (siehe Abschnitt 5.3.2 für weitere Möglichkeiten). In diesem Fall müsste zusätzlich beachtet werden, dass am Ende immer noch die Ausgangsbegriffe gegenüber den hinzugefügten Begriffen in der Mehrzahl sind, um die ursprünglichen Bedeutungen nicht zu sehr zu verzerren. Es müsste also eine obere Schranke für die Zahl hinzugefügter Begriffe in Abhängigkeit der bereits vorhandenen Begriffe eingeführt werden. Zusätzlich könnte sichergestellt werden, dass der Informationsgehalt der hinzugefügten Begriffe für eine Methode nicht geringer ist als der Informationsgehalt der Begriffe für eine Klasse. Damit kann einer zu großen informationellen Streuung entgegengewirkt werden. Letzteres Vorgehen benötigt aber wie erwähnt aufgrund der Berücksichtigung von Informationsgehalt bzw. Auftrittswahrscheinlichkeit ein kompatibles Wissensnetz und ist für diese Arbeit damit ungeeignet.

Zu diesem Ansatz wird in der Arbeit *„Verbesserung von Worteinbettungs-basierter Rückverfolgbarkeitsanalyse durch Konzeptwissen“* [Koc22] von Koch ein worteinbettungsbasiertes Verfahren vorgestellt, welches zu gegebenen Ausgangsbedeutungs-Vektoren diejenigen Synonyme, Hyperonyme und Holonyme anreichert, deren Einbettungen eine bestimmte Distanzschwelle zu den Ausgangsvektoren unterschreiten. Somit soll einerseits die ausreichende semantische Verwandtschaft der Begriffe sowie andererseits die Mehrheit der Ausgangsbegriffe sichergestellt werden. Dieser in Teilen lernbasierte Ansatz erfordert allerdings ein Sprachmodell für Worteinbettungen.

In der Arbeit *„Unsupervised Multi-Topic Labeling for Spoken Utterances“* [Wei+19] (siehe Abschnitt 4.3) wird ein Verfahren zur Themenbeschriftung kurzer mündlicher Aussagen vorgestellt. Dafür wird ausgehend von den initialen Bedeutungsknoten ein Subgraph aufgebaut und innerhalb diesem dann Themenbeschriftungen ermittelt. Vorgestellt werden dafür die *top connectivity strategy*, welche als Kandidaten diejenigen Knoten wählt, die am stärksten mit den initialen Bedeutungsknoten verbunden sind; sowie die *max coverage strategy*, welche als Kandidaten diejenigen Knoten wählt, die am meisten noch unerreichte initiale Bedeutungsknoten überdecken. Für die Ziele dieser Arbeit scheint die *top connectivity strategy* gut geeignet zu sein, da das Anreichern von besonders umfassend aussagekräftigen Begriffen gewünscht ist, um die Semantik der Bags-of-Embeddings im FTLR-Projekt feiner auszudifferenzieren. Die *max coverage strategy* hingegen scheint weniger geeignet, weil eine Präferenz möglichst großer Abdeckung gegenüber informationeller Streuung einzelner Begriffe anfälliger ist. Außerdem ist eine Bag-of-Embeddings-übergreifende Abdeckung angereicherter Begriffe kein erklärtes Ziel der FTLR-Optimierung. Insgesamt scheint der Ansatz von Weigelt et al. vielversprechend zu sein, da er auf die Themenbeschriftung kurzer Aussagen ausgelegt ist und Begriffsmengen von Methodensignaturen oder Javadoc-Kommentaren von ähnlich kleinem Umfang sind. In der Arbeit wurde DBpedia verwendet, allerdings stellt das Verfahren keine weiteren Anforderungen an das verwendete Wissensnetz, sodass prinzipiell auch WordNet oder BabelNet nutzbar sind. Ein weiterer Vorteil ist, dass bereits eine Java-Implementierung des Verfahrens zur Verfügung steht und daher gut adaptierbar ist.

Die Arbeit *„Knowledge-based graph document modeling“* [SP14] (siehe Abschnitt 4.3) stellt einen weiteren Ansatz zur Themenbeschriftung vor. Darin wird ein Subgraph basierend auf kurzen Pfaden zwischen den initialen Begriffen aufgebaut und über Informationsgehalt-Metriken die Kanten gewichtet. Anschließend werden Pfade gesucht, deren Informationsgehalt besonders groß ist, und die so erreichbaren Knoten als Beschriftungskandidaten ausgewählt. Manche dieser Metriken basieren unter anderem auf dem Informationsgehalt des jeweiligen Kantentyps $\mathrm{I}(\mathrm{pred})$. Bei einer Verwendung eines DBpedia-Ausschnitts (mit Kantentypen wie in Abschnitt 5.2.1) führt dies zum Problem, dass nur wenige Kantentypen zur Verfügung stehen. Es ist anzunehmen, dass deren Auftrittswahrscheinlichkeiten in den DBpedia-Datensätzen weniger aussagekräftig für die Zwecke solcher Metriken sind. Die Hinzunahme weiterer DBpedia-Datensätze würde hingegen eine zu große informationelle Streuung und außerdem Qualitätseinbußen bedeuten. Dazu kommt der größere Implementierungsaufwand für das vorgestellte Verfahren. Der Ansatz ist also für den Rahmen dieser Arbeit nicht geeignet.

Fazit

Da in Abschnitt 5.2.1 bereits die Verwendung von DBpedia gegenüber WordNet empfohlen wurde, eignen sich vor allem das Anreichern einer beschränkten Zahl an verwandten Begriffen sowie der Ansatz aus der Arbeit *„Unsupervised Multi-Topic Labeling for Spoken Utterances“* [Wei+19] (siehe Abschnitt 4.3). Ansätze, die den Informationsgehalt einzelner Bedeutungen miteinbeziehen, benötigen ein kompatibles Wissensnetz wie WordNet.

5.3.2. Einbindung der Anreicherungen in FTLR

In diesem Abschnitt sollen Möglichkeiten untersucht werden, wie die angereicherten Begriffe in für die Verbesserung von Anforderungsrückverfolgung in FTLR [Hey+21] genutzt werden können. Dafür soll zunächst die Funktionsweise von FTLR kurz umrissen werden.

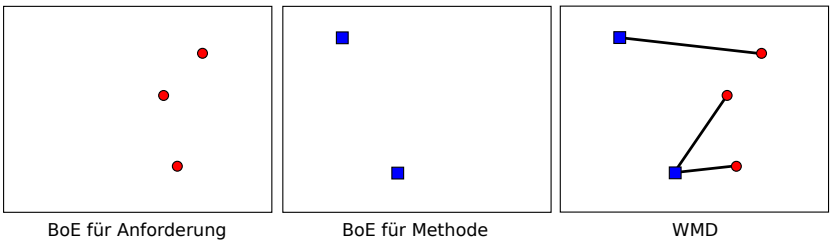
Zu einzelnen Elementen der Anforderungsdokumentation wie Anforderungssätzen oder Anwendungsfall-Beschreibungsschritten werden Wörter extrahiert und daraus Mengen von statischen Worteinbettungen, sogenannte *Bags-of-Embeddings* (kurz *BoE*), erstellt. Dasselbe geschieht auf der Quelltextseite für Datentypennamen und erweiterte Methodensignaturen, wobei optional weitere Begriffe z. B. aus Kommentaren zu den einzelnen BoE hinzugefügt werden können.

Anschließend wird paarweise für je eine Anforderungs-BoE und eine Quelltext-BoE die *Word Mover's Distance* (kurz *WMD*) bestimmt. Vereinfacht ausgedrückt wird dabei jeder Wortvektor des einen Artefakts dem nächstgelegenen Wortvektor des anderen Artefakts zugeordnet und anschließend die Distanzen zwischen diesen Vektorpaaren aufsummiert, was dann als Distanz der beiden BoE verstanden wird. Über diese BoE-Distanzen werden dann die Zuordnungen zwischen Anforderungen und Datentypen hergestellt (Näheres dazu ist in *„Improving Traceability Link Recovery Using Fine-grained Requirements-to-Code Relations“* [Hey+21] von Hey et al. zu finden). Ziel dieser Arbeit ist es, die Zuordnungen zwischen BoE der Anforderungsdokumentation und BoE der Datentypen über hinzugefügte Wörter zu verbessern.

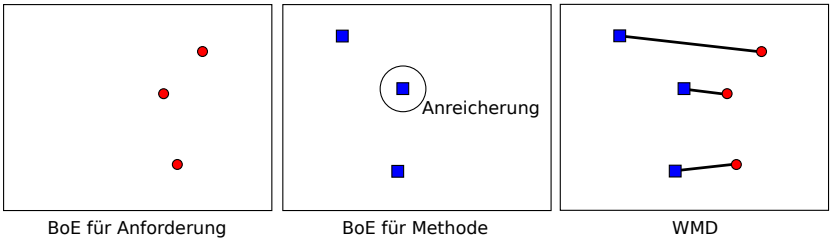
In der Arbeit *„Verbesserung von Worteinbettungs-basierter Rückverfolgbarkeitsanalyse durch Konzeptwissen“* [Koc22] von Koch werden Begriffe in Anforderungstexten mit Wissen angereichert, um einbettungsbasierte TLR zu verbessern. Für die Einbindung angereicherter Wörter in das FTLR-Projekt werden dort mehrere Ansätze vorgestellt.

Eine naheliegende Lösung ist die Vereinigung der Ausgangswortmenge einer BoE mit der Menge dazu angereicherter Wörter. Die Hinzunahme angereicherter Wörter kann dabei helfen, die WMD zwischen BoE zugehöriger Artefakt-Elemente zu verringern. Beispiel 5.5 veranschaulicht dies grafisch. Bei diesem Ansatz muss allerdings beachtet werden, dass die Ausgangswörter in der Mehrzahl bleiben, um eine zu starke Verzerrung durch die hinzugefügten Wörter zu vermeiden. Dafür gibt es mehrere Ansätze: Zum einen kann bei der Wissensanreicherung die Anzahl hinzugefügter

Beispiel 5.5: BoE-Erweiterung



Oben links und oben Mitte zeigen jeweils zweidimensionale Darstellungen von Bags-of-Embeddings. Oben rechts können die WMD-Zuordnungen der Einbettungen abgelesen werden. Die mittlere Worteinbettung der Anforderungs-BoE muss auf die untere Einbettung der Methoden-BoE abgebildet werden, welche allerdings weit entfernt ist, wodurch sich für beide BoE eine hohe WMD ergibt.
Durch Hinzufügen der Einbettung eines angereicherten Wortes in der Methoden-BoE (unten Mitte) verringern sich die Abstände der Vektorzuordnungen (unten rechts) und damit auch die WMD beider BoE.



Wörter eingeschränkt werden. Zum anderen können die neben den angereicherten Begriffen die ursprünglich aufgelösten Bedeutungen ebenfalls hinzugefügt werden, um durch mehrfache Vorkommen der Ausgangswörter die Berechnung der WMD zugunsten der Ausgangswörter zu gewichten.

Ein weiterer Ansatz aus der Arbeit von Koch erstellt für jedes Anforderungselement eine eigene BoE mit den zugehörigen angereicherten Begriffen, welche dann separat zugeordnet werden kann. Auf die Quelltextseite übertragen ermöglicht dieses Vorgehen, dass beispielsweise ein Datentyp neben den BoE seiner Methoden weitere BoE dazugewinnt, welche dann später die Zuordnung dieses Datentyps zu zugehörigen Anforderungen erleichtern kann.

Da der zeitliche Rahmen dieser Arbeit eine tiefergehende Analyse der Einbindungsmöglichkeiten nicht erlaubt, soll hierfür stattdessen auf die bereits erwähnte Arbeit *„Verbesserung von Worteinbettungs-basierter Rückverfolgbarkeitsanalyse durch Konzeptwissen“* [Koc22] von Koch verwiesen werden.

6. Entwurf

In diesem Kapitel soll aufbauend auf der Analyse in Kapitel 5 der Lösungsentwurf für die beschriebenen Teilprobleme aufgestellt und begründet werden. Die Aufteilung des Entwurfes erfolgt anhand der dort definierten Zielfragen.

So ergibt sich als erstes aus der Frage nach dem Umfang der Wissensanreicherung und der Nutzbarkeit der Quelltextinformationen ein Vorverarbeitungsschritt, in welchem für die Wissensanreicherung nützliche Informationen aus dem Quelltext extrahiert und linguistisch verarbeitet werden (siehe Abschnitt 6.1). Basierend darauf ist der zweite Schritt die Bedeutungsauflösung derjenigen Wörter, die für eine weitere Anreicherung infrage kommen (siehe Abschnitt 6.2). Der dritte Schritt beinhaltet sowohl die weitere Anreicherung der aufgelösten Bedeutungen mit Wissen als auch die Einbindung der angereicherten Wörter in das FTLR-Projekt [Hey+21] zur Anforderungsrückverfolgung (siehe Abschnitt 6.3). Abschnitt 6.4 fasst die einzelnen Teilschritte nochmals zusammen.

6.1. Vorverarbeitung

Wie in Abschnitt 5.1.2 ausgeführt wird, sind für die natürlichsprachliche Verarbeitung von Begriffen in Quelltexten lediglich Bezeichner von Quelltextelementen sowie Kommentartexte für die Ziele dieser Arbeit geeignet. Diese sollen wie in Abschnitt 5.1 empfohlen vorverarbeitet werden.

Für Bezeichner im Quelltext sollen dabei folgende Schritte durchgeführt werden: Die Bezeichner werden anhand Binnenmajuskel- bzw. Binnenunterstrich-Schreibweise in einzelne Bestandteile zerlegt. Ziffern und Unterstriche werden dabei entfernt. Die übrigen Bestandteile werden mithilfe einer Heuristik lemmatisiert und deren Wortart erkannt. Es wurde dafür eine Heuristik gewählt, weil wie in Abschnitt 5.1.1 beschrieben auf Quelltextbezeichnern bislang Wortarterkennung nicht erforscht wurde.

Diese Heuristik bildet einfache englischsprachige Sätze bzw. Satzteile aus den Bestandteilen und verarbeitet diese anschließend mit herkömmlichen Werkzeugen zur Lemmatisierung und Wortarterkennung von natürlichsprachlichen Texten. Wie zuvor analysiert wurde, stellt dieser Ansatz eine vielversprechendere Heuristik dar als beispielsweise der Abgleich mit einem Wörterbuch. Für diese Heuristik werden Bestandteile von Methodennamen in Kleinbuchstaben umgewandelt, mit Leerzeichen konkateniert und mit dem Präfix `"Objects␣"` versehen, Bestandteile aller anderen

Beispiel 6.1: Bildung einfacher Sätze aus Bezeichner-Bestandteilen

Klasse `HtmlParser`:
{ `Html`, `Parser` } ⟶ `"The␣html␣parser."`
Methode `getPatientRecord`:
{ `get`, `Patient`, `Record` } ⟶ `"Objects␣get␣patient␣record."`
Variable `patientList`:
{ `patient`, `List` } ⟶ `"The␣patient␣list."`

Bezeichner (Datentyp- und Variablennamen) werden in Kleinbuchstaben umgewandelt, mit Leerzeichen konkateniert und mit dem Präfix `"The␣"` zu einer Nominalphrase zusammengesetzt. Die so gebildeten Sätze bzw. Satzteile werden mit einem Punkt beendet. Beispiel 6.1 zeigt das Verfahren an einigen Bezeichnern. Unter der Annahme, dass die meisten Methodennamen aus einem Verb, potentiell gefolgt von einem/mehreren Objekten bestehen, soll mit *„Objects“* als vorangestelltem Substantiv ein vollständiger Subjekt-Verb-Objekt-Satz gebildet werden. Die Pluralform *„Objects“* soll die Erkennung der üblicherweise verwendeten Infinitiv-Verbformen zuverlässiger machen. Unter der Annahme, dass die meisten Nicht-Methoden-Bezeichner aus Substantiven bestehen, soll eine Voranstellung des bestimmten Artikels *„The“* eine Erkennung der nachfolgenden Bestandteile als Substantive zuverlässiger machen.

In Abschnitt 5.1.2 wird erläutert, dass ausschließlich Substantive für die Anreicherung geeignet sind. Somit werden nur als Substantiv erkannte Bestandteile beibehalten und alle anderen Bestandteile verworfen. Bestandteile, die aus drei oder weniger Buchstaben bestehen, werden als Abkürzungen eingestuft und verworfen, da eine Abkürzungsauflösung den Rahmen dieser Arbeit übersteigt. Anschließend werden mithilfe einer Stoppwortliste alle Stoppwörter (inkl. programmierbezogener Stoppwörter) verworfen.

Für Javadoc-Kommentare und übrige Kommentare sollen folgende Schritte durchgeführt werden: Der Text wird mittels herkömmlicher Werkzeuge zur Tokenisierung, Lemmatisierung und Wortarterkennung von natürlichsprachlichen Texten verarbeitet. Tags wie `@author` und HTML-Tags sowie URLs werden verworfen, genauso wie Ziffern- oder Sonderzeichenfolgen. Kommentare, die ein `TODO` oder `FIXME` enthalten, werden verworfen. Die aufgespaltenen Bestandteile werden wie oben beschrieben weiter abkürzungs- und stoppwortbereinigt. Weiter werden mit einer einfachen Heuristik unter den Textbestandteilen mögliche Kandidaten für Quelltextbezeichner erkannt. Diese Heuristik sucht nach Binnenunterstrich- oder Binnenmajuskel-Schreibweise in den aufgespaltenen Textbestandteilen. Solche möglichen Kandidaten werden wie Quelltextbezeichner behandelt und separat wie oben beschrieben verarbeitet. Anschließend werden ebenfalls nur Substantive beibehalten und alle übrigen Bestandteile verworfen.

Es liegen nun also zu jedem Quelltextelement mit Bezeichner bzw. zu jedem Kommentar die enthaltenen erkannten Substantive vor, wobei Sonderzeichen und Ziffern sowie Abkürzungen und Stoppwörter verworfen worden sind. Beispiel 6.2 zeigt einen Beispielquelltext und Beispiel 6.3 das Ergebnis der Vorverarbeitung an diesem Quelltext.

6.2. Wortbedeutungsauflösung (WSD)

Nach den erfolgten Vorverarbeitungsschritten werden nun die Bedeutungen der extrahierten Substantive aufgelöst.

6.2.1. WSD-Verfahren

Da in Abschnitt 5.2.1 der Einsatz eines wissensbasierten Verfahrens empfohlen wird, soll zunächst ein Wissensnetz gewählt werden. Da WordNet den beiden Alternativen DBpedia und BabelNet

Beispiel 6.2: Ausgangsquelltext

```
1  /**
2    * Class for managing a bank's checking account.
3    */
4  public class CheckingAccount implements BankAccount {
5      /* ... */
6
7      /**
8        * Withdraws an amount of money from this account if
           possible.
9        * Throws an exception otherwise.
10       */
11     public void withdraw(double amount) {
12         if (!canWithdraw(amount)) {
13             throw new IllegalArgumentException("Balance too
                   low for withdrawal.");
14         } else {
15             this.balance -= amount;
16         }
17         // TODO make threadsafe
18     }
19
20     /* ... */
21 }
```

Beispiel 6.3: Vorverarbeitung von Beispiel 6.2

Zuerst werden die Bezeichner aufgespalten. Die folgende Tabelle zeigt dies auszugsweise.

Zeile	aufgespaltene Textbestandteile
2	Class, for, managing, a, bank, 's, checking, account, .
4	Checking, Account
...	...
9	Throws, an, exception, otherwise, .
11	withdraw, amount
12	can, Withdraw, amount
...	...
17	TODO, make, threadsafe

Die so erhaltenen Bestandteile werden anschließend lemmatisiert und deren Wortarten erkannt. Nicht-Substantive wie *„managing“*, *„withdraw“* oder *„otherwise“* werden verworfen. Anschließend werden Stoppwörter und Wörter mit Länge $\leqslant 3$ verworfen. Daher bleibt in Zeile 9 auch das Substantiv *„exception“* nicht erhalten, da es ein programmierbezogenes Stoppwort ist. Auch der Kommentar in Zeile 17 verschwindet, da er ein *„TODO“* enthält. Das Ergebnis kann in folgender Tabelle abgelesen werden.

Zeile	extrahierte Substantive	Zeile	extrahierte Substantive
2	bank, account	11	amount
4	account	12	amount
8	amount, money, account	15	balance, amount
9	—	17	—

Beispiel 6.4: WSD innerhalb Beispiel 6.2 mit analytischem Kontext

Wir nehmen an, dass die Methode `CheckingAccount:withdraw(double)` ausschließlich von der Methode (A) `WithdrawalTransaction:execute()` aufgerufen wird und wie in Beispiel 6.2 die Methode (B) `CheckingAccount.canWithdraw(double amount)` aufruft. Es folgt eine Auflistung aller aufzulösenden Wörter sowie der Kontextwörter, welche alle an UKB übergeben werden.

Quelle	Substantiv	auflösen?	Art
Klassenname (Zeile 4)	account	nein	aus Auflösungseinheit
Methodenparameter (Z. 11)	amount	ja	aus Auflösungseinheit
Javadoc (Z. 8)	amount	ja	aus Auflösungseinheit
Javadoc (Z. 8)	money	ja	aus Auflösungseinheit
Javadoc (Z. 8)	account	ja	aus Auflösungseinheit
Klassenname	withdrawal	nein	Kontext (A)
Klassenname	transaction	nein	Kontext (A)
Klassenname	account	nein	Kontext (B)
Methodenparameter	amount	nein	Kontext (B)

Nach der WSD für diese Auflösungseinheit erhalten wir beispielsweise folgende DBpedia-Bedeutungen für die aufzulösenden Wörter:

Substantiv	Bedeutung
amount	Quantity
amount	Quantity
money	Money
account	Bank account

bezüglich Domänenüberdeckung unterlegen ist, soll WordNet hier nicht verwendet werden. Aufgrund der automatischen Konsolidierung mehrerer Wissensnetze in mehreren Sprachen kann BabelNet eine höhere Fehlerrate aufweisen als DBpedia (siehe Abschnitt 5.2.1). Daher fällt die Wahl auf DBpedia als verwendetes Wissensnetz.

Für die WSD wird das in Abschnitt 5.2.1 empfohlene wissensbasierte Verfahren von Agirre et al. [ALS14] (siehe Abschnitt 4.1.1) eingesetzt. Dafür spricht, dass das verwendete Wissensnetz frei gewählt werden kann, ohne das Verfahren neu implementieren zu müssen. Beim Verfahren von Moro et al. [MRN14] (siehe Abschnitt 4.1.1) hingegen kann in der Referenzimplementierung nur BabelNet genutzt werden; für andere Wissensnetze muss das Verfahren selbst neu implementiert werden. Da Letzteres den Rahmen der Arbeit sprengt, fällt die Entscheidung auf das Verfahren von Agirre et al.

Erste Testläufe auf einem kleinen selbstverfassten Drei-Klassen-Projekt aus der Domäne der Finanzwirtschaft suggerieren, dass eine Anwendung des PPR-Verfahrens nicht lohnenswert ist, weshalb nur das PPR_{w2w}-Verfahren verwendet werden soll (siehe Abschnitt 4.1.1).

6.2.2. Wahl eines geeigneten Wortkontextes

Die in Abschnitt 5.2.2.1 definierten Auflösungseinheiten sollen als kleinste Zellen gemeinsam aufzulösender Begriffe übernommen werden. Für Datentypen bilden die extrahierten Substantive aus den einfachen Namen und zugeordneten Javadoc-Kommentaren eine Auflösungseinheit. Für öffentliche Methoden bilden die extrahierten Substantive aus den einfachen Methodennamen, zugeordneten Javadoc-Kommentaren und den einfachen Namen von Parametern und ihren Typen

eine Auflösungseinheit. Zusätzlich soll über eine Option die Auflösung von Substantiven aus Rumpfkommentaren wählbar sein; diese gehören dann ebenfalls zur Auflösungseinheit einer Methode.

Die Auflösungseinheiten sollen mit weiteren Kontextwörtern aus verschiedenen Quellen vergrößert werden und dann mit dem Verfahren von Agirre et al. [ALS14] gemeinsam aufgelöst werden. Dabei werden die Kontextwörter nicht aufgelöst, beeinflussen allerdings die Bedeutungsauflösung der übrigen Begriffe. Zu den Kontextwörtern einer Methode zählen immer die Substantive aus dem Namen des übergeordneten Datentyps. Im Folgenden soll die Verwendung weiterer möglichen Kontextquellen erläutert werden. Es ist anzumerken, dass die einzelnen Kontextquellen miteinander kombinierbar sind.

Deklarativer Kontext

Als deklarativer Kontext eines Datentyps sollen die ihm untergeordnet deklarierten Elemente verstanden werden (siehe Abschnitt 5.2.2.2). Als deklarativer Kontext einer Methode sollen die mit ihr gemeinsam deklarierten Quelltextelemente, also der deklarative Kontext des übergeordneten Datentyps, sowie der Name des übergeordneten Datentyps verstanden werden. Über eine Option soll der Ausschluss nicht-öffentlicher Elemente wählbar sein. Zusätzlich soll über eine weitere Option die Hinzunahme von Rumpfanweisungs-Begriffen zu Methodenkontexten wählbar sein. Als solche zählen Methodenaufrufe und Datenzugriffe.

Situativer Kontext

Der situative Kontext eines Datentyps soll nicht verwendet werden, da ein solcher wie in Abschnitt 5.2.2.3 begründet vermutlich nicht geeignet ist.

Aufgrund der Struktur des abstrakten Syntaxbaumes des Eingabequelltextes, mit welcher dieser innerhalb von INDIRECT dargestellt wird, ist eine Implementierung von zeilenbeschränktem oder verzweigungsabhängigem situativen Kontext einer Methode im zeitlichen Rahmen dieser Arbeit nicht möglich. Daher soll lediglich die grundlegende Definition des situativen Kontextes einer Methode zur Verfügung stehen. Es sollen also Rumpfanweisungen um einen Aufruf der fraglichen Methode herum als situative Kontextquellen betrachtet werden. Dabei sollen die situativen Kontexte mehrerer Aufrufstellen zu einer Kontextmenge vereinigt werden, da eine Umsetzung der in Abschnitt 5.2.2.3 beschriebenen Alternativen den zeitlichen Rahmen der Arbeit sprengt. Genauso schließt auch der Zeitrahmen eine Gewichtung der Kontexte nach abgeschätzter Aufrufstapeltiefe aus.

Analytischer Kontext

Die Definition von analytischem Kontext in Abschnitt 5.2.2.4 wird größtenteils übernommen: Als analytischer Kontext einer Methode sollen die von ihr aufrufabhängigen und datenabhängigen Methoden verstanden werden. Als analytischer Kontext eines Datentyps sollen die von ihm direkt vererbungsabhängigen Typen, also direkte Ober- und Untertypen, verstanden werden. Eine Analyse der Aufrufabhängigkeiten zwischen Datentypen wie in der Arbeit *„Facilitating program comprehension with call graph multilevel hierarchical abstractions“* [AGL21] (siehe Abschnitt 4.2) soll aufgrund des zeitlichen Rahmens dieser Arbeit ausbleiben.

Beispiel

Beispiel 6.4 veranschaulicht die Wahl aufzulösender Begriffe und die Wahl von Kontextwörtern. In diesem Beispiel soll der analytische Kontext für die WSD verwendet werden, um die Begriffe innerhalb der Auflösungseinheit der Methode `CheckingAccount:withdraw(double)` aufzulösen (siehe Beispiel 6.3). Dazu gehören die Substantive aus dem zugehörigen Javadoc-Kommentar (amount, money, account) sowie die Substantive aus der erweiterten Methodensignatur (amount). Die erste Tabelle des Beispiels enthält die Wörter, die an UKB übergeben werden. Die zweite Tabelle zeigt das Ergebnis der Bedeutungsauflösung.

6.3. Weitere Wissensanreicherung

Die zuvor ermittelten Wortbedeutungen können nun verwendet werden, um Quelltextelemente mit weiterem Wissen anzureichern. Zuerst sollen das verwendete Verfahren vorgestellt werden und anschließend die Einbindung in FTLR beschrieben werden.

6.3.1. Verfahren

Um den zeitlichen Rahmen dieser Arbeit nicht zu überschreiten, soll von den in Abschnitt 5.3.1 vorgestellten Ansätzen lediglich der Ansatz von *„Unsupervised Multi-Topic Labeling for Spoken Utterances“* [Wei+19] (siehe Abschnitt 4.3) verwendet werden. Für diesen Ansatz spricht die bereits verfügbare Implementierung sowie die Kompatibilität mit dem für die WSD gewählten Wissensnetz DBpedia. Die verwendete Strategie (*„top connectivity strategy“* oder *„max connectivity strategy“*) kann über eine Option gewählt werden, um den Einfluss beider Vorgehensweisen evaluieren zu können.

Die Gruppierung und gemeinsame Anreicherung der aufgelösten Bedeutungen erfolgt analog zu den zuvor definierten Auflösungseinheiten: Für Datentypen werden die aufgelösten Bedeutungen aus deren Namen und Javadoc-Kommentaren gemeinsam angereichert, für Methoden werden die aufgelösten Bedeutungen aus deren erweiterten Methodensignaturen, deren Javadoc-Kommentaren und optional auch deren Rumpfkommentaren gemeinsam angereichert.

Um sicherzustellen, dass die Ausgangsbedeutungen in der Mehrzahl bleiben, wird die Anzahl maximaler Themenbeschriftungen auf die Anzahl der Ausgangsbedeutungen minus eins gesetzt, wobei gemäß der Referenzimplementierung eins und acht zusätzlich als feste Schranken gelten. Die Themenbeschriftungen werden zusätzlich mit Standardwerkzeugen zur Verarbeitung natürlicher Sprache lemmatisiert und stoppwortbereinigt, da FTLR für die Berechnung der Worteinbettungen ebenfalls lemmatisierte und stoppwortbereinigte Wortmengen verwendet. Dabei werden nichtalphabetische Zeichen verworfen, da diese für die Worteinbettungen keinen Nutzen tragen. Beispiel 6.5 zeigt mögliche Ergebnisse des Verfahrens anhand der in Beispiel 6.4 aufgelösten Bedeutungen.

6.3.2. Einbindung der Anreicherung in FTLR

Um den zeitlichen Rahmen der Arbeit nicht zu überschreiten, wird lediglich das in Abschnitt 5.3.2 vorgestellte Verfahren umgesetzt, bei welchem zu einer bestehenden Wortmenge eines Quelltextelements die zugehörigen angereicherten Begriffe hinzugefügt werden. Dieses naheliegende Verfahren ist vergleichsweise einfach implementierbar. Wie in der Analyse beschrieben soll optional das zusätzliche Hinzufügen der ursprünglichen Ausgangsbedeutungen wählbar sein. Zusätzlich

Beispiel 6.5: Wissensanreicherung für Begriffe aus Beispiel 6.4

Die Menge der aufgelösten Bedeutungen der Auflösungseinheit `CheckingAccount:withdraw(double)` ist { Quantity, Money, Bank account }.
Die Anzahl der ursprünglichen Bedeutungen ist drei, also wird zwei als Anzahl maximaler Themenbeschriftungen gesetzt, was zwischen den Schranken eins und acht liegt.
Mit dem Verfahren von Weigelt et al. können somit beispielsweise folgende Themenbeschriftungen berechnet werden:

Themenbeschriftung	ursprüngl. Bedeutungen
Finance	Money, Bank account
Measurement	Quantity

Beispiel 6.6: Einbinden der angereicherten Begriffe aus Beispiel 6.5 in FTLR

Die Wortmenge für `CheckingAccount:withdraw(double)` soll zur Berechnung der zugehörigen Bag-of-Embeddings (BoE) bestimmt werden. Die folgende Tabelle zeigt die vorliegenden Informationen. FTLR ermittelt bereits eine Menge der Ausgangswörter aus dem Quelltext (erste Tabellenzeile; siehe auch Beispiel 6.2 Z. 4 und 11).

FTLR-Ausgangswörter	checking, account, withdraw, amount
Urspr. Bedeutungen	quantity, money, bank account
Anreicherungen	finance, measurement

Nach dem Hinzufügen der angereicherten Begriffe zu den Ausgangswörtern ergeben sich folgende Ergebniswortmengen. Duplikate können wahlweise verhindert oder zugelassen werden. Die ursprünglichen Bedeutungen können wahlweise ebenfalls hinzugefügt oder ausgelassen werden.

mit Duplikaten?	mit urspr. Bed.?	Ergebniswortmenge
nein	nein	checking, account, withdraw, amount, finance, measurement
ja	nein	(wie erste Zeile, da keine Duplikate)
nein	ja	checking, account, withdraw, amount, quantity, money, bank, finance, measurement
ja	ja	checking, account, withdraw, amount, quantity, money, bank, account, finance, measurement

sollen optional mehrfache Wortvorkommen entweder erlaubt oder verhindert werden. Damit soll der Einfluss einer durch Mehrfachvorkommen erfolgten Gewichtung der Ausgangswörter evaluiert werden können.

Bevor die Themenbeschriftungen (und ggf. auch die ursprünglichen Bedeutungen) zu einer Wortmenge hinzugefügt werden, werden diese an Leerzeichen aufgetrennt. Beispiel 6.6 zeigt die verschiedenen Wahlmöglichkeiten der FTLR-Einbindung anhand der zuvor angereicherten Begriffe aus Beispiel 6.5.

6.4. Zusammenfassung

Der hier vorgestellte Lösungsentwurf besteht aus den folgenden Schritten, welche in Abbildung 6.1 grafisch dargestellt werden:

1. Vorverarbeitung aller Bezeichner und Kommentartexte zur Extraktion enthaltener Substantive
2. Bedeutungsauflösung der einzelnen Auflösungseinheiten mittels gewählter Kontextoption
3. Anreicherung für die ermittelten Wortbedeutungen mit dem gewählten Verfahren
4. Import der Begriffe und ihrer Anreicherungen in FTLR [Hey+21] mittels gewähltem Verfahren, anschließend Anforderungsrückverfolgung mittels FTLR

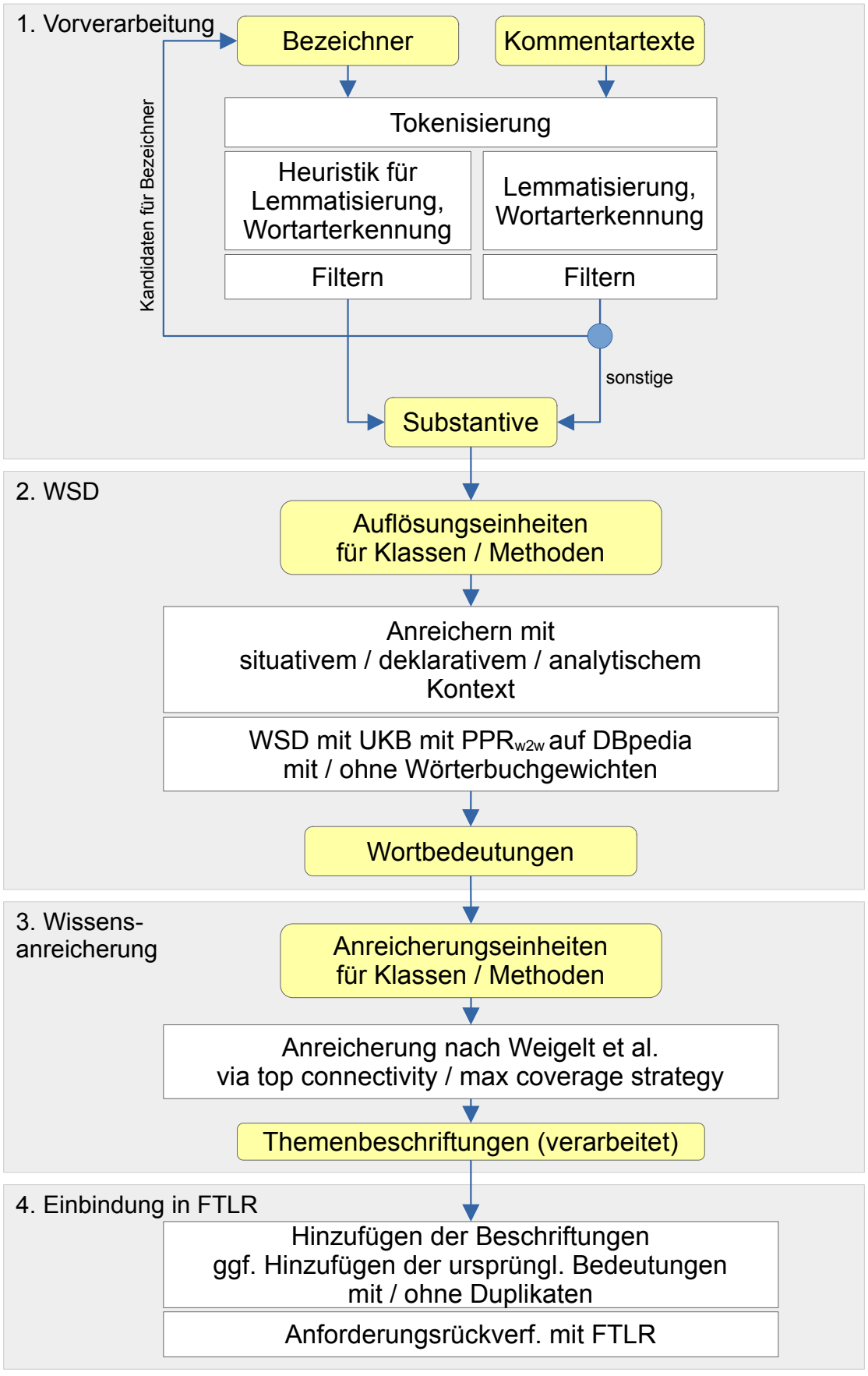


Abbildung 6.1.: Ablaufgrafik für den Lösungsentwurf

7. Implementierung

In diesem Kapitel soll die Umsetzung des in Kapitel 6 vorgestellten Lösungsentwurf beschrieben werden. Dabei werden die Schritte von der Vorverarbeitung bis zur Wissensanreicherung in Form von INDIRECT-Agenten [Hey19] implementiert und die Resultate mittels Dekorationsklassen in das FTLR-Projekt [Hey+21] eingebunden (siehe Kapitel 3).

Die Umsetzung innerhalb von INDIRECT ist auf vier Maven-Projekte aufgeteilt:

- `indirect_codeKnowledgeParent`, welches Hilfsklassen enthält, die von den anderen Projekten benutzt werden,
- `indirect_codeWsdPreprocessing`, welches die für die Wortbedeutungsauflösung (WSD) nötige Vorverarbeitung durchführt (siehe Abschnitt 6.1),
- `indirect_codeDisambiguation`, welches die Bedeutungen der vorverarbeiteten Begriffe auflöst (siehe Abschnitt 6.2) und
- `indirect_codeEnrichment`, welches die aufgelösten Bedeutungen mit weiterem Wissen anreichert (siehe Abschnitt 6.3.1).

Die letzten drei aufgeführten Maven-Projekte werden in den Abschnitten 7.1, 7.2 und 7.3.1 näher beschrieben. Die Umsetzung der Einbindung der angereicherten Begriffe in FTLR (siehe Abschnitt 6.3.2) wird in Abschnitt 7.3.2 kurz beschrieben.

7.1. Vorverarbeitung

Zunächst soll beschrieben werden, welche Daten von der Vorverarbeitungs-Pipeline von INDIRECT zur Verfügung gestellt werden. Der Eingabequelltext wird von INDIRECT analysiert und in eine Graphenstruktur überführt, die einem abstrakten Syntaxbaum ähnelt. Einzelne Quelltextelemente wie Datentypen, Methoden, Attribute, aber auch Anweisungen oder Schleifenabschnitte sind durch Knoten dargestellt. Die Quelltextelemente sind durch Kanten miteinander hierarchisch verknüpft, wobei Kreuzkanten Beziehungen wie Methodenaufrufe oder Datenzugriffe darstellen.

Die Klasse `CodePreprocessingAgent` führt auf diesem bereits vorliegenden Graphen nun die WSD-Vorverarbeitung durch. Jedes Quelltextelement, das einen Bezeichner hat, sowie alle Kommentar-Knoten bekommen eigene `CodeToken`-Knoten zugeordnet. Die `CodeToken`-Knoten eines Bezeichners enthalten die aufgespaltenen Bezeichner-Bestandteile sowie zugehörige Lemmata und Wortarten, die mithilfe der Stanford-CoreNLP-Werkzeugsammlung [Man+14] gemäß der in Abschnitt 6.1 beschriebenen Heuristik ermittelt werden. Die `CodeToken`-Knoten eines Kommentars enthalten analog die Textbestandteile, Lemmata und Wortarten, die via Standardverfahren mit

denselben Werkzeugen ermittelt werden. Dabei werden Textbestandteile, die als Kandidaten für Bezeichner erkannt werden, wie Bezeichner aufgespalten und weiterverarbeitet. Anschließend werden alle Nicht-Substantiv-Knoten entfernt.

Die Klasse `CodeStopwordRemoverAgent` entfernt daraufhin alle `CodeToken`-Knoten, die Stoppwörter oder Wörter mit drei oder weniger Buchstaben enthalten. Dafür werden Stoppwortlisten aus dem FTLR-Projekt verwendet, welche auch programmierbezogene Stoppwörter enthalten.

7.2. Wortbedeutungsauflösung (WSD)

Nach der Vorverarbeitung sollen die Bedeutungen bestimmter `CodeToken`-Knoten aufgelöst werden. Die einzelnen Bestandteile des Verfahrens sind in die folgenden Unterpakete gegliedert:

Im Paket `disambiguationsourceprovider` liegen Klassen, die die in Abschnitt 5.2.2 definierten Auflösungseinheiten für die jeweiligen Quelltextelemente erfassen und die entsprechenden `CodeToken`-Mengen zusammenstellen. Im Paket `contextprovider` liegen Kontextanbieter-Klassen, die für eine Auflösungseinheit die zugehörigen Kontextwörter-`CodeToken` ermitteln. Die unterschiedlichen Kontextvarianten aus Abschnitt 6.2.2 sind als Kompositum-Struktur umgesetzt und damit beliebig kombinierbar. Im Paket `analysis` liegen Klassen zur Quelltextanalyse wie dem Ermitteln von Aufrufabhängigkeiten, Datenabhängigkeiten oder Vererbungsabhängigkeiten. Die ermittelten Abhängigkeiten werden als eigene Kanten in den Graphen eingefügt, welche anschließend von den Kontextanbietern benutzt werden.

Die Klasse `CodeDisambiguationAgent` koordiniert die Zusammenstellung der Auflösungseinheiten, Ermittlung der Kontextwörter, Durchführen der WSD und Eintragen der Ergebnisse in den Graphen. Relevante Aspekte dieses Verfahrens sollen im Folgenden beleuchtet werden.

7.2.1. Verfahren

Für das in Abschnitt 6.2.1 gewählte WSD-Verfahren von Agirre et al. [ALS14] (siehe Abschnitt 4.1.1) existiert mit der Werkzeugsammlung UKB eine Referenzimplementierung. Diese erhält als Eingabe eine Menge von Auflösungseinheiten, in denen aufzulösende Wörter und Kontextwörter unterschiedlich markiert sind. UKB ermittelt dann mit dem gewählten Verfahren die Bedeutungen der aufzulösenden Wörter anhand eines gegebenen Wissensnetzes.

7.2.1.1. DBpedia als Wissensnetz für UKB

Um DBpedia als Wissensquelle für UKB einzusetzen, müssen für UKB relevante Daten aus DBpedia extrahiert und speziell formatiert werden. Die Verwendung von DBpedia als Wissensnetz für UKB wurde bereits in der Arbeit *„Multiwort-Bedeutungsauflösung für Anforderungen“* [Bar20] von Bartel erörtert, woran sich das Vorgehen dieser Arbeit orientiert.

Wörterbuch

UKB benötigt ein Wörterbuch, in welchem für jedes vorkommende Eingabewort die verschiedenen möglichen Bedeutungen aufgelistet sind. Dafür eignet sich der DBpedia-Datensatz „Disambiguations“, der aus den Begriffsklärungs-Artikeln der Wikipedia generiert worden ist. Beispielsweise enthält der Artikel *„Key“* Verlinkungen zu möglichen Bedeutungen des Wortes *„Key“* wie z. B. *„Key (cryptography)“* oder *„Lock and key“*. Diese Beziehungen werden im Datensatz als RDF-Tripel wiedergegeben:

```
dbr:Key dbo:wikiPageDisambiguates dbr:Key_(cryptography).
dbr:Key dbo:wikiPageDisambiguates dbr:Lock_and_key.
```

UKB bietet die Option, die verschiedenen möglichen Bedeutungen eines Wortes gemäß ihrer Auftrittshäufigkeit zu gewichten. Diese Gewichte beeinflussen das Verhalten des zugrundeliegenden PPR-Algorithmus. Um solche Gewichte für DBpedia zu ermitteln, wurde für jeden Wikipedia-Artikel die Anzahl an Verlinkungen auf diesen Artikel aus dem „Wikilinks"-Datensatz anhand der Relation `dbo:wikiPageWikiLink` ermittelt. Es ist anzunehmen, dass die Verlinkungshäufigkeit eine ausreichend adäquate Ersatzlösung für das Problem der fehlenden Gewichte darstellt.

Aus den „Disambiguations"- und „Wikilinks"-Einträgen lassen sich also Wörterbucheinträge für UKB generieren. Für jedes Wort werden dafür die möglichen Bedeutungen zusammen mit deren Auftrittshäufigkeiten aufgelistet. Beispiel 7.1 zeigt in Zeile 1 das Ergebnis der Umwandlung der beiden RDF-Tripel oben.

Eine weitere Quelle für Wörterbuch-Einträge stellt der „Redirects"-Datensatz dar. Dieser enthält automatische Weiterleitungen von Wikipedia. Beispielsweise leitet „Amount" auf „Quantity" weiter, da beides Synonyme sind. Solche RDF-Tripel sind ebenfalls nützlich für die WSD, um Synonyme zu vereinheitlichen. Beispiel 7.1 zeigt in den Zeilen 2 und 3 die entsprechenden Wörterbucheinträge für „Amount" und „Quantity".

Beispiel 7.1: UKB-Wörterbucheinträge

```
1  Key Key_(cryptography):34 Lock_and_key:153
2  Amount Quantity:623
3  Quantity Quantity:623
```

DBpedia-Datensatz	Relation	Relation (vollständig)
Disambiguations	`dbo:wikiPageDisambiguates`	`<http://dbpedia.org/ontology/wikiPageDisambiguates>`
Redirects (transitive)	`dbo:wikiPageRedirects`	`<http://dbpedia.org/ontology/wikiPageRedirects>`
Wikilinks	`dbo:wikiPageWikiLink`	`<http://dbpedia.org/ontology/wikiPageWikiLink>`

Tabelle 7.1.: Verwendete DBpedia-Relationen für die Erstellung des UKB-Wörterbuchs

Wissensgraph

Um aus DBpedia einen Wissensgraphen zu erstellen, der von UKB für das PPR_{w2w}-Verfahren benutzt werden kann, wurden die „Categories“- und die „Linked Hypernyms“-Datensätze verwendet. Der „Categories (Articles)“-Datensatz ordnet einem Wikipedia-Artikel die Kategorien zu, denen er zugeordnet ist. In „Categories (SKOS)“ werden Kategorien einander zugeordnet, die thematisch verwandt sind oder zueinander in einer Über/-Unterkategorie-Beziehung stehen. Die „Linked Hypernym“-Datensätze ordnen einzelnen Artikeln solche Artikel zu, die als Oberbegriffe bzw. Klassifizierungen dienen. Wie bereits in Abschnitt 5.2.1 argumentiert wird, sind vor allem Hyperonym-Relationen für die WSD relevant. Daher ist es naheliegend, diese Datensätze für die Graphen-Erstellung zu verwenden.

Für UKB müssen die Relationen für den zu definierenden Wissensgraphen kantenweise angegeben werden. Beispielsweise lässt sich aus dem „Categories“-Tripel

```
dbr:Rabbit  dct:subject  dbc:Herbivorous_mammals.
```

somit folgende Kantendefinition für den UKB-Graphen gewinnen:

```
u:Rabbit v:Herbivorous_mammals
```

In den Tabellen 7.1 und 7.2 sind die verwendeten DBpedia-Relationen im Detail referenziert.

Datenbereinigung

Es sind nicht alle derartig erhaltenen Wikipedia-Artikel als Bedeutungsknoten oder Wörterbucheinträge für UKB geeignet. Beispielsweise ist anzunehmen, dass Artikel über öffentliche Personen, Listen bestimmter Ereignisse nach Jahreszahl, Film- oder Buchtitel sowie andere populärkulturelle Entitäten nicht als Bedeutungsknoten infrage kommen, da diese wahrscheinlich kaum in Anforderungsdokumentation noch Quelltext vorkommen, mindestens aber keine Rolle für die Software-Semantik spielen.

Da eine umfassende Datenbereinigung den Rahmen dieser Arbeit sprengt, wurde stattdessen eine einfache Heuristik angewendet: Es wurden ausschließlich solche Wörterbucheinträge bzw. Bedeutungsknoten zugelassen, die aus alphabetischen Zeichen, Unterstrichen, Bindestrichen oder Klammern bestehen. So bleibt beispielsweise `Key_(cryptography)` bestehen, während Einträge für `Bambi_(1942)` verworfen werden. Dieses Vorgehen ist mit der in Abschnitt 5.1.1 beschriebenen Vorverarbeitung abgestimmt, bei welcher ausschließlich Wörter aus alphabetischen Zeichen übrig bleiben.

7.2.1.2. UKB-Aufruf

Der `CodeDisambiguationAgent` ermittelt die Auflösungseinheiten, stellt die entsprechenden Knotenmengen mithilfe der Anbieter-Klassen zusammen und schreibt eine Eingabedatei für UKB, in welche für jedes aufzulösende Wort zusätzlich die Knoten-ID sowie der vollständige Name des zugehörigen Quelltextelements eingetragen werden, um aus der Ausgabe von UKB die Ergebnisse wieder in den Graphen zurückzuschreiben.

DBpedia-Datensatz	Relation	Relation (vollst.)
Categories (Articles)	`dct:subject`	`<http://purl.org/dc/terms/subject>`
Categories (SKOS)	`skos:broader`	`<http://www.w3.org/2004/02/skos/core#broader>`
Categories (SKOS)	`skos:related`	`<http://www.w3.org/2004/02/skos/core#related>`
Linked Hypernyms (DBO)	`rdf:type`	`<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>`
Linked Hypernyms (Ext)	`gold:hypernym`	`<http://purl.org/linguistics/gold/hypernym>`

Tabelle 7.2.: Verwendete DBpedia-Relationen für die Erstellung des UKB-Graphen

Anschließend wird UKB mit Kommandozeilenparametern aufgerufen. Im Paket `ukboptions` liegen Aufzählungstypen, die die verschiedenen UKB-Optionen kapseln. Es kann hierbei gewählt werden, welches Verfahren angewendet werden soll (PPR oder PPR_{w2w}) sowie das Beachten der angegebenen Wörterbuchgewichtung aktiviert oder deaktiviert werden. Die ermittelten Bedeutungen werden in die `CodeToken`-Knoten als Bedeutungsattribut zurückgeschrieben. Als Format wurden die jeweiligen DBpedia-Ressourcen-URLs gewählt.

7.3. Weitere Wissensanreicherung

UKB wurde zu Evaluierungszwecken getrennt von der Agenten-Pipeline aufgerufen. Im späteren Implementierungsverlauf wurde festgestellt, dass technisch bedingt bei der Generierung des Quelltextgraphen die Knoten-IDs nicht deterministisch gewählt werden. Da die Identifizierung der zugehörigen Knoten über die IDs abläuft, konnten die ermittelten UKB-Ergebnisse nicht in den Graphen zurückgeschrieben werden. Weil eine lineare Suche nach vollständigen Namen bei der vorliegenden Evaluationsprojekt-Größe nicht skaliert, wurde auf eine agentenbasierte Implementierung verzichtet.

Stattdessen wurde im Paket `codeEnrichmentGraphless` eine simple Hashmap-basierte Datenstruktur gebaut, welche die relevanten Teile des Quelltextgraphen weitestgehend imitiert. Näheres dazu soll im Folgenden kurz beschrieben werden.

7.3.1. Verfahren

Analog zu der Bedeutungsauflösung werden im Paket `enrichmentunit` Klassen für die Anreicherungseinheiten (siehe Abschnitt 6.3.1) definiert. Das Paket `enrichmentstrategy` bietet Kapselung für verschiedene Anreicherungsstrategien wie beispielsweise die `WeigeltHypernymEnrichment`-Strategie für das Verfahren nach Weigelt et al. [Wei+19] (siehe Abschnitt 4.3). Für dieses Verfahren wurde die Referenzimplementierung eingebunden. Die Größe der Datenmenge von DBpedia macht die Verwendung der Online-Zugriffs-Schnittstelle der Referenzimplementierung notwendig.

Die Begriffe der ermittelten Themenüberschriften werden lemmatisiert und stoppwortbereinigt mit den vollständigen Namen der zugehörigen Quelltextelemente in eine Eingabedatei für die FTLR-Erweiterung geschrieben.

7.3.2. Einbindung in FTLR

In FTLR gibt es verschiedene `TraceabilityRunner`-Klassen, welche die verschiedenen FTLR-Verfahrensoptionen abbilden. Diese verwenden sogenannte `WordChooser`-Klassen, die zu einem Quelltextelement die Wortliste bereitstellen, die für die Bags-of-Embeddings eingebettet werden soll (siehe Abschnitt 5.3.2).

Zwei Dekorierer-Klassen im Paket `enrichment` fügen die aus der Eingabedatei gelesenen Wörter in die Wortlisten ein bzw. dekorieren in den `TraceabilityRunner`-Klassen die jeweiligen `WordChooser` entsprechend. Dabei stehen Optionen für das Einbinden der ursprünglichen Bedeutungswörter sowie für das Zulassen oder Verhindern von Duplikaten in den Wortlisten zur Verfügung.

8. Evaluation

In diesem Kapitel soll versucht werden, die Zielfragen aus Kapitel 5 zu beantworten. Dafür wird die in Kapitel 7 vorgestellte Implementierung evaluiert. Zunächst werden die verwendeten Datensätze beschrieben, die verwendeten Metriken definiert und die evaluierten Verfahrensoptionen angegeben. Anschließend werden die Evaluationsergebnisse diskutiert.

8.1. Evaluationsgrundlagen

Ziel dieser Arbeit ist es, worteinbettungsbasierte Anforderungsrückverfolgung (engl. *traceability link recovery, TLR*) durch Wissensanreicherung von Begriffen im Quelltext zu verbessern. Dafür sollten die Ergebnisse dieser Arbeit in das FTLR-Projekt [Hey+21] integriert werden. Zur besseren Vergleichbarkeit ist es naheliegend, Datensätze sowie einzelne Metriken und Verfahrensoptionen aus der ursprünglichen Evaluation von FTLR zu übernehmen.

8.1.1. Verwendete Datensätze

Aus Zeitgründen werden lediglich zwei der ursprünglich vier verwendeten Datensätze aus FTLR übernommen: eTour aus der Tourismusdomäne und eAnci aus der Domäne der Regierungsverwaltung (siehe Tabelle 8.1). Beide Projekte enthalten kompilierbaren, Java-basierten Quelltext und werden häufig in Arbeiten zur automatischen Anforderungsrückverfolgung verwendet. eTour enthält hauptsächlich natürlichsprachlichen Text auf Englisch mit Ausnahme von Quelltextbezeichnern und Anwendungsfalltiteln, die auf Italienisch verfasst sind. Da der worteinbettungsbasierte Ansatz von FTLR auf Einsprachigkeit ausgelegt ist, wurden alle italienischen Wörter und Quelltextbezeichner mithilfe eines Wörterbuches von Hand übersetzt. eAnci ist vollständig auf Italienisch verfasst. Im FTLR-Projekt wurde für eAnci daher ein italienisches Sprachmodell

			Artefakt-Anzahl			∅ Methoden pro K.		Überdeckung	
Projekt	Domäne	Sprache	AF	K	TL	öffentl.	n.-öffentl.	AF	K
eTour	Tourismus	EN/IT	58	116	308	7.0	3.6	0.983	0.767
eAnci	Verwaltung	IT	139	55	567	5.1	0.2	0.281	1.000

Tabelle 8.1.: Die in der Evaluation verwendeten Datensätze [Hey+21]. Die Artefakt-Überdeckung gilt in Bezug auf Rückverfolgbarkeitsverbindungen zwischen AF und K.
AF: Anwendungsfälle; K: Klassen;
TL: Tracelinks (dt. *Rückverfolgbarkeitsverbindungen*) laut Musterlösung.

verwendet. Für die Evaluation von Wissensanreicherung auf englischsprachigen Wissensquellen wurde der eAnci-Datensatz mithilfe von Google Translate übersetzt.

Wie in Tabelle 8.1 zu sehen ist, ist für eAnci die Überdeckung der Anwendungsfälle mit TL niedrig. Dies kann darauf hindeuten, dass nur wenige Anforderungen im Quelltext implementiert worden sind oder die TL-Musterlösung unvollständig ist. Eine niedrige Überdeckung kann hierbei zu einer hohen Zahl falsch-positiver Zuordnungen führen (siehe nächster Abschnitt) [Hey+21].

8.1.2. Verwendete Metriken

TLR kann als Klassifikationsproblem aufgefasst werden. Um TLR zu evaluieren, werden übliche Klassifikations-Metriken verwendet. Dafür werden folgende Kennzahlen definiert: TP als Anzahl der *richtig-positiven* Zuordnungen (engl. *true positives*), FP als Anzahl der *falsch-positiven* Zuordnungen (engl. *false positives*), TN als Anzahl der *richtig-negativen* Zuordnungen (engl. *true negatives*) und FN als Anzahl der *falsch-negativen* Zuordnungen (engl. *false negatives*). Eine Zuordnung ist richtig-positiv, wenn sie vom Klassifikationsverfahren ermittelt wurde und in der Musterlösung vorkommt; falsch-positiv, wenn sie vom Verfahren ermittelt wurde, aber in der Musterlösung nicht vorkommt; richtig-negativ, wenn sie vom Verfahren nicht ermittelt wurde und auch in der Musterlösung nicht vorkommt; oder falsch-negativ, wenn sie vom Verfahren nicht ermittelt wurde, obwohl sie in der Musterlösung vorkommt. Die Menge aller richtig- und falsch-positiven Zuordnungen sind die insgesamt ermittelten Zuordnungen (engl. *retrieved links*). Die Menge aller richtig-positiven und falsch-negativen Zuordnungen, also alle laut Musterlösung korrekten Zuordnungen, heißen auch die *relevanten* Zuordnungen (engl. *relevant links*). [JM21]

Präzision und Ausbeute

Die *Präzision* (engl. *precision*) gibt an, wieviele der insgesamt ermittelten Zuordnungen auch laut Musterlösung korrekt sind. Die *Ausbeute* (engl. *recall*) gibt an, wieviele der laut Musterlösung korrekten Zuordnungen auch als positiv ermittelt worden sind. [JM21]

$$\text{Präzision} := \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\left|\text{Ermittelte Zuordnungen}\right|} \tag{8.1}$$

$$\text{Ausbeute} := \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\left|\text{Relevante Zuordnungen}\right|} \tag{8.2}$$

F_1-Maß

Das F_1-Maß ist definiert als das harmonische Mittel aus Präzision und Ausbeute. Das harmonische Mittel aus zwei Werten strebt anders als das arithmetische Mittel eher in Richtung des kleineren Wertes. Dies ist hier von Vorteil, da Verfahren mit hoher Ausbeute dazu tendieren, eine geringe Präzision zu liefern bzw. Verfahren mit hoher Präzision eher eine geringe Ausbeute liefern. Das F_1-Maß spiegelt derartige Qualitätsmängel eines Verfahrens adäquater wider als beispielsweise das arithmetische Mittel aus Präzision und Ausbeute. Da für TLR sowohl eine hohe Ausbeute als auch eine hohe Präzision gefragt sind, ist eine Evaluation nach der F_1-Metrik geeignet. [JM21]

$$\text{F}_1 := 2 \cdot \frac{\text{Präzision} \cdot \text{Ausbeute}}{\text{Präzision} + \text{Ausbeute}} \tag{8.3}$$

MAP-Maß

Da TLR-Verfahren bislang keine F_1-Werte erreichen, die eine vollständige Automatisierung des Prozesses zulassen, geben die meisten Verfahren für jedes Anforderungsartefakt eine Rangliste von Zuordnungs-Kandidaten aus, die Menschen bei der manuellen TLR unterstützen soll. Um diese Ranglisten zu bewerten, wird die *gemittelte durchschnittliche Präzision* (engl. *mean average precision,*

MAP) verwendet. Dabei ist die *durchschnittliche Präzision* (*AP*) für ein Anforderungsartefakt a wie folgt definiert:

$$\mathrm{AP}(a) := \frac{\sum_{r=1}^{|\text{ermittelt}|} \big(\text{Präzision}(r) \cdot \text{relevant}(r)\big)}{\big|\text{Relevante Zuordnungen für } a\big|}, \tag{8.4}$$

wobei $|\text{ermittelt}|$ die Anzahl der ermittelten Zuordnungen für a, die Zahl r der Listenrang, der Wert $\text{Präzision}(r)$ die Präzision der hinter Rang r abgeschnittenen Liste, die Funktion $\text{relevant}(r) = 1$ falls die r-te Zuordnung der Liste korrekt (ansonsten null) und $\big|\text{Relevante Zuordnungen für } a\big|$ die Anzahl der relevanten (= laut Musterlösung korrekten) Zuordnungen für a ist. Der MAP-Wert ist demnach $\mathrm{AP}(a)$ gemittelt über alle Anforderungsartefakte a. [Hey+21]

8.1.3. Evaluierte Verfahren

Aufgrund des beschränkten zeitlichen Rahmens dieser Arbeit können nicht alle für diese Arbeit implementierten Techniken und Verfahrensweisen evaluiert werden. Eine Evaluation der Vorverarbeitungsschritte und der dort verwendeten Heuristiken zur Lemmatisierung und Wortarterkennung von Quelltextbezeichnern (siehe Abschnitt 6.1) entfällt daher. Zielfrage 1 bleibt damit unbeantwortet.

Ebenso konnte keine Musterlösung für die Wortbedeutungsauflösung (WSD) erstellt werden, weshalb die WSD nicht als Einzelschritt, sondern nur in Verbindung mit der anschließenden Wissensanreicherung evaluiert werden kann. Dennoch soll untersucht werden, inwieweit sich für feste Anreicherungsparameter die FTLR-Ergebnisse in Abhängigkeit der WSD-Parameter verbessern. Zielfrage 2 kann daher nur eingeschränkt beantwortet werden.

Die Wissensanreicherung kann ebenfalls nicht sinnvoll als Einzelschritt evaluiert werden, da eine Musterlösung sehr viele verschiedene mögliche Anreicherungskombinationen enthalten müsste, um nicht zu einschränkend zu sein. Eine Nutzerstudie zur Bewertung der angereicherten Begriffe ließe nur begrenzt Rückschlüsse auf die Verbesserung der FTLR-Ergebnisse zu. Deshalb ist es naheliegend, die Wissensanreicherung nur in Verbindung mit der anschließenden TLR zu evaluieren. Somit liegen nach Ablauf der Vorverarbeitung, WSD, Wissensanreicherung und Ausführen des FTLR-Verfahrens ausschließlich die FTLR-Evaluationsergebnisse vor. Mit den Ergebnissen dieser Evaluation kann dann Zielfrage 3 beantwortet werden. Die konkret evaluierten Verfahrensoptionen der einzelnen Schritte werden im Folgenden beschrieben.

Wie bereits zuvor erläutert, wird die Evaluation auf den eTour- und eAnci-Datensätzen durchgeführt. Nach der Vorverarbeitung erfolgt die WSD, wobei als Kontext für alle Auflösungseinheiten lediglich der deklarative Kontext ohne Rumpfanweisungen oder -kommentare unter Ausschluss nicht-öffentlicher Elemente verwendet wird. Es sei daran erinnert, dass zu einer Auflösungseinheit gehörende Javadoc-Kommentar-Substantive standardmäßig mitaufgelöst werden. Als WSD-Verfahren wird PPR_{w2w} via UKB nach Agirre et al. [ALS14] (siehe Abschnitt 4.1.1) verwendet. Das Verfahren wird mit und ohne Berücksichtigung von UKB-Wörterbuch-Gewichten (engl. *dictionary weights, dw*) durchgeführt. Das Wörterbuch und dessen Gewichte sowie der UKB-Graph wurden aus DBpedia abgeleitet.

Für die Wissensanreicherung wird das Verfahren nach Weigelt et al. [Wei+19] (siehe Abschnitt 4.3) verwendet. Als mögliche Strategien sollen „max coverage“ und „top connectivity“ verwendet werden.

Die Einbindung der angereicherten Begriffe in FTLR erfolgt mit und ohne Hinzunahme der Wörter aus den ermittelten ursprünglichen Bedeutungen. Parallel wird beim Hinzufügen von Begriffen das Auftreten von Duplikaten sowohl verhindert als auch zugelassen.

Anschließend wird die Implementierung von FTLR aufgerufen. Dafür stehen folgende voneinander unabhängige Optionen zur Verfügung:

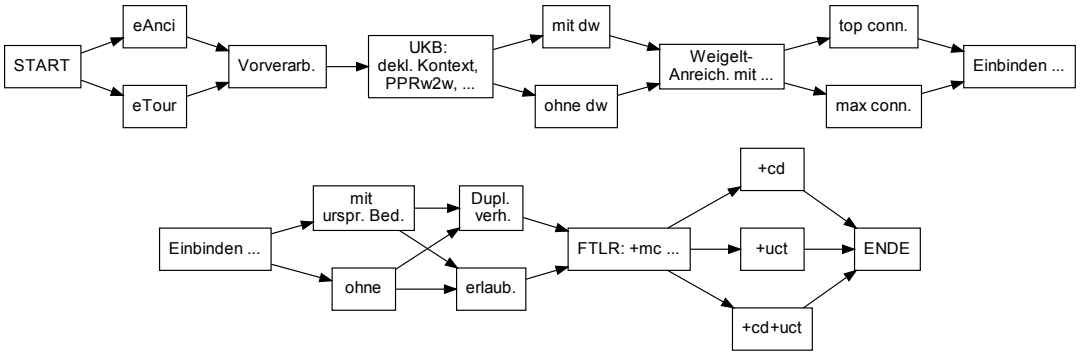


Abbildung 8.1.: Ablaufgraph zu den versch. Verfahrensoptionen

- +mc (engl. *with method comments*) – Hinzunahme von Wörtern in Methodenkommentaren,
- +cd (engl. *with call dependencies*) – Hinzunahme von Wörtern in Signaturen von aufrufabhängigen Methoden und
- +uct (engl. *with use case templates*) – Benutzung von Anwendungsfall-Schablonen zur verbesserten Informationsextraktion (für Details siehe die Arbeit von Hey et al. [Hey+21]).

Da bereits im WSD-Schritt Javadoc-Kommentare miteinbezogen wurden, werden lediglich diejenigen Aufrufkonfigurationen von FTLR verwendet, die ebenfalls Methodenkommentare bei der Erstellung von Bags-of-Embeddings miteinbeziehen. Die entsprechenden Kombinationen sind +mc, +mc+cd, +mc+uct und +mc+cd+uct. Es ist zu beachten, dass in der momentanen Standardkonfiguration die Hinzunahme von Methodenparameter-Typnamen deaktiviert ist. Diese Einstellung wurde für die Evaluation nicht verändert. Es sei jedoch angemerkt, dass die Hinzunahme von Methodenparameter-Typnamen in der Implementierung des WSD-Verfahrens standardmäßig aktiv ist. Ferner liegen in FTLR für eAnci keine Aufrufabhängigkeitsinformationen vor, weshalb für eAnci die Hinzunahme der Option +cd keine Auswirkungen hat und daher nicht gesondert evaluiert wird. Abbildung 8.1 zeigt den Ablauf und die möglichen Kombinationen der Verfahrensoptionen noch einmal grafisch.

FTLR verwendet zwei Schwellenwerte $\mathfrak{m}$ und $\mathfrak{f}$ („major“ und „final“), um die Herstellung von Rückverfolgbarkeitsverbindungen zu kalibrieren [Hey+21]. Standardeinstellung ist $\mathfrak{m} = 0.59$ und $\mathfrak{f} = 0.44$. Durch die Einbindung der Wissensanreicherung kann es notwendig werden, die Schwellenwerte neu zu kalibrieren. In Abschnitt 8.3 wird darauf näher eingegangen. Sofern nicht anders angegeben, geben Wertetabellen den Zustand der Standardeinstellung wieder.

In den Tabellen dieses Kapitels werden die folgenden Abkürzungen verwendet: „Dupl. verh./erlaub.“ für das Verhindern oder Erlauben von Duplikaten sowie „+uB“ und „−uB“ für das Hinzufügen bzw. Weglassen der ursprünglichen Bedeutungen. Tabellenzeilen mit „∅“ kennzeichnen den jeweiligen Durchschnitt über die beiden Datensätze. Referenzwerte für FTLR-Anwendung ohne Anreicherung (engl. *Baseline*) sind mit „BL“ beschriftet und werden auch in Differenz-Tabellen nicht als Differenz, sondern absolut angegeben. Es sei daran erinnert, dass die Option „+cd“ bei eAnci wirkungslos ist und diesbezüglich keine separaten Werte angegeben werden. Die Option „+cd“ wird für eAnci daher eingeklammert dargestellt.

8.2. Wortbedeutungsauflösung (WSD)

Da bei der WSD lediglich die Berücksichtigung der DBpedia-Wörterbuchgewichte variiert wurde, lässt sich für Zielfrage 2 auch nur dieser Aspekt untersuchen. Es sei nochmals verdeutlicht, dass der reine Einfluss dieser WSD-Optionenwahl anhand der vorliegenden Ergebnisse nur schwierig einschätzbar ist, da die anderen Verfahrensoptionen die Ergebnisse deutlich stärker und teilweise auch gegenteilig beeinflussen können. Aus demselben Grund lässt sich der reine Einfluss der WSD ebenfalls nicht untersuchen. Tabelle 8.2 zeigt, wie sich die F_1-Werte der FTLR-Ausgabe verändern,

F_1 (ohne dw) minus F_1 (mit dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	+.005	−.004	+.020	−.010	+.017	−.008	**+.032**	−.010
+mc+uct(+cd)	+.003	+.002	+.027	−.002	+.015	−.001	**+.042**	−.002
eTour +mc	−.004	+.012	−.012	**+.013**	−.004	+.009	−.009	+.012
+mc+cd	+.002	+.013	−.012	**+.015**	+.003	+.011	−.007	+.011
+mc+uct	**+.023**	−.016	**+.023**	−.014	+.015	−.014	−.002	−.015
+mc+cd+uct	+.009	−.016	**+.020**	−.015	+.011	−.015	+.002	−.016
∅ +mc	+.001	+.004	+.004	+.002	+.007	+.001	**+.012**	+.001
∅ +mc+cd	+.004	+.005	+.004	+.003	+.010	+.002	**+.013**	.000
∅ +mc+uct	+.013	−.007	**+.025**	−.008	+.015	−.008	+.020	−.009
∅ +mc+cd+uct	+.006	−.007	**+.024**	−.009	+.013	−.008	+.022	−.009

Tabelle 8.2.: Veränderung der F_1-Werte durch Nichtbeachten der Wörterbuchgewichte (dw) statt Miteinbeziehen der Gewichte. Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.

wenn Wörterbuchgewichte ignoriert statt berücksichtigt werden. In den Spalten finden sich die Parameter der Wissensanreicherung und FTLR-Einbindung. In den Zeilen sind die Datensätze und FTLR-Optionen angegeben, wobei die letzten vier Zeilen den Durchschnitt über die evaluierten Datensätze zeigen.

In den Daten finden sich Verbesserungen von bis zu +3.2 % und Verschlechterungen von bis zu −1.6 %. Es fällt auf, dass beim Einbinden der ursprünglichen Bedeutungen in FTLR („+uB“) tendenziell häufiger eine F_1-Verbesserung zustande kommt bzw. F_1-Verbesserungen stärker ausfallen als beim Nicht-Einbinden der ursprünglichen Bedeutungen („−uB“). Dies kann auch in den Durchschnittszeilen über alle Datensätze gesehen werden, wo nur in den erwähnten Spalten ausschließlich bzw. deutlich größere Verbesserungen zu finden sind. Darüber hinaus befinden sich in den Durchschnittszeilen ausschließlich in „+uB“-Spalten Zeilenmaxima und Zeilen-Zweitbeste, wohingegen ausschließlich in „−uB“-Spalten Verschlechterungen zu finden sind. Die erwähnten F_1-Verbesserungen bedeuten, dass das Verfahren in den jeweiligen Fällen bei Nichtbeachtung der DBpedia-Wörterbuchgewichte („ohne dw“) bessere Ergebnisse liefert als bei Beachtung der Wörterbuchgewichte („mit dw“). Dass diese Verbesserungen überwiegend in den Konfigurationen auftreten, in denen ursprüngliche Bedeutungen miteingebunden werden („+uB“), könnte darauf hinweisen, dass das Nichtbeachten der Wörterbuchgewichte präziser aufgelöste Bedeutungen liefert, welche dann das F_1-Ergebnis gegenüber dem Beachten der Wörterbuchgewichte verbessern. Würden hingegen die F_1-Verbesserungen überwiegend in den „−uB“-Spalten auftreten, wäre das ein Hinweis darauf, dass das Nichtbeachten der Gewichte die WSD so sehr verschlechtert, dass ein Einbinden der aufgelösten Bedeutungen schlechtere F_1-Werte liefert als das Nicht-Einbinden. Anhand der Durchschnittszeilen der Tabelle 8.2 kann dies ausgeschlossen werden.

Eine solche Schlussfolgerung würde allerdings voraussetzen, dass das Einbinden von ursprünglichen Bedeutungen unabhängig von deren Korrektheit nicht inhärent nachteilig für das gesamte Verfahren ist. Dies kann ohne Musterlösung für die WSD-Ergebnisse nicht überprüft werden.

Es sei darauf hingewiesen, dass bei der Betrachtung der F_1-Differenzen unterschlagen werden kann, dass die umgekehrte Differenz (F_1 mit dw minus F_1 ohne dw) größere Verbesserungen aufweisen kann. Damit bestünde die Gefahr einer fehlerhaften Schlussfolgerung aufgrund der Subtraktionsrichtung. Da die negativen Werte in den Durchschnittszeilen und auch im Rest der Tabelle betragsmäßig kleiner sind als die positiven Werte, ist dies ausgeschlossen.

ΔF_1 (mit dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	+.005	+.042	−.006	**+.046**	−.002	+.026	−.017	+.028	.159
+mc+uct(+cd)	−.006	+.013	−.017	**+.017**	−.010	+.001	−.028	+.002	.178
eTour +mc	−.012	−.017	**−.005**	−.019	−.017	−.016	−.020	−.018	.351
+mc+cd	−.012	−.014	**−.001**	−.016	−.019	−.012	−.016	−.012	.343
+mc+uct	−.032	−.020	−.028	−.020	−.023	−.019	**−.016**	−.019	.470
+mc+cd+uct	−.026	−.019	−.028	−.019	−.016	−.018	**−.011**	−.018	.473
∅ +mc	−.004	+.013	−.006	**+.014**	−.010	+.005	−.019	+.005	
∅ +mc+cd	−.004	+.014	−.004	**+.015**	−.011	+.007	−.017	+.008	
∅ +mc+uct	−.019	−.004	−.023	**−.002**	−.017	−.009	−.022	−.009	
∅ +mc+cd+uct	−.016	−.003	−.023	**−.001**	−.013	−.009	−.020	−.008	

Tabelle 8.3.: Differenz ΔF_1 aus F_1-Werten für Anreicherung bei WSD mit Wörterbuchgewichtung (dw) sowie den Referenzwerten ohne Anreicherung (siehe Anhang). Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.

Es fällt auf, dass sich für die Zeilen „eTour +mc“ und „eTour +mc+cd“ die Größenunterschiede der F_1-Differenzen von den meisten übrigen Zeilen unterscheiden: Die „−uB“-Spalten haben dort tendenziell größere F_1-Verbesserungen als die „+uB“-Spalten. Betrachtet man allerdings die Veränderung der Präzisionswerte in Bezug auf die Baseline ohne Anreicherung in den Tabellen 8.5 und 8.6, so ist erkennbar, dass sowohl mit als auch ohne Wörterbuchgewichte die Bevorzeichnung mit den vorher diskutierten Werten übereinstimmt: In den „+uB“-Spalten sind ausschließlich Verbesserungen zu finden, Verschlechterungen finden sich nur in den „−uB“-Spalten. Die schwankenden F_1-Werte können bei der vorliegenden Präzisionsverbesserung nur durch Ausbeute-Verschlechterungen zustande kommen. Diese können damit erklärt werden, dass die Werte sich bislang auf die Standardeinstellung der FTLR-Schwellenwerte m und f beziehen. Diese Schwellenwerte tragen vereinfacht gesagt dazu bei, eine ausgewogene Balance zwischen Präzision und Ausbeute zu erhalten. In den Tabellen 8.7 und 8.8 ist angegeben, wie sehr sich die F_1-Werte in Bezug auf die Baseline verändern, wenn für jede Tabellenzelle die optimale Schwellenwertkombination verwendet wird. Für die eTour-Zeilen „+mc“ und „+mc+cd“ sind in den Spalten eins, zwei und vier sind die Werte für „ohne dw“ größer als die Werte für „mit dw“. In den übrigen Spalten sind die Größenverhältnisse der Zeilen allerdings wieder umgekehrt. Da in jeder Tabellenzelle eine andere optimale Schwellenwertkombination verwendet wurde, lassen sich keine Aussagen machen, wie die Größenverhältnisse für eine global optimale Schwellenwertkombination aussehen. Eine solche optimale Neu-Kalibrierung der Schwellenwerte entfällt im Rahmen dieser Arbeit. Die unsichere Datenlage erlaubt für die genannten eTour-Zeilen lediglich spekulative Erklärungsversuche.

Insgesamt kann festgehalten werden, dass die vorliegenden Daten die Annahme suggerieren, dass die Verwendung der aus DBpedia extrahierten Wörterbuchgewichte die WSD mit DBpedia und damit auch die weitere Wissensanreicherung verschlechtern. Dies kann allerdings nur mit einem Abgleich der ermittelten Wortbedeutungen mithilfe einer Musterlösung sicher bestätigt werden, was im Rahmen dieser Arbeit entfällt.

8.3. Wissensanreicherung

Im Folgenden sollen nun alle Schritte des evaluierten Verfahrens untersucht werden. WSD soll hier als Teil der Wissensanreicherung aufgefasst werden. Da für die Evaluation der weiteren Wissensanreicherung keine Musterlösung für die Wortbedeutungen genutzt wurde, ist es möglich,

ΔF_1 (ohne dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	+.010	**+.037**	+.014	**+.037**	+.016	+.018	+.015	+.018	.159
+mc+uct(+cd)	−.003	**+.015**	+.010	**+.015**	+.004	−.000	+.014	−.001	.178
eTour +mc	−.016	**−.005**	−.016	−.006	−.020	−.006	−.029	−.006	.351
+mc+cd	−.009	**−.001**	−.013	**−.001**	−.016	**−.001**	−.022	**−.001**	.343
+mc+uct	−.009	−.036	**−.005**	−.035	−.008	−.033	−.018	−.034	.470
+mc+cd+uct	−.017	−.035	−.008	−.034	**−.005**	−.034	−.009	−.034	.473
∅ +mc	−.003	**+.016**	−.001	**+.016**	−.002	+.006	−.007	+.006	
∅ +mc+cd	.000	**+.018**	.000	**+.018**	.000	+.009	−.004	+.009	
∅ +mc+uct	−.006	−.011	**+.003**	−.010	−.002	−.017	−.002	−.018	
∅ +mc+cd+uct	−.010	−.010	+.001	−.010	−.001	−.017	**+.003**	−.018	

Tabelle 8.4.: Differenz ΔF_1 aus F_1-Werten für Anreicherung bei WSD ohne Wörterbuchgewichtung (dw) sowie den Referenzwerten ohne Anreicherung (siehe Anhang). Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.

ΔPräzision (mit dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	**+.110**	+.090	+.096	+.093	+.082	+.041	+.055	+.042	.158
+mc+uct(+cd)	**+.092**	+.023	+.078	+.026	+.075	−.000	+.045	.000	.164
eTour +mc	+.020	−.018	**+.037**	−.019	+.014	−.018	+.014	−.019	.276
+mc+cd	+.022	−.015	**+.042**	−.016	+.014	−.015	+.020	−.015	.270
+mc+uct	+.020	−.008	+.027	−.008	+.020	−.008	**+.028**	−.008	.374
+mc+cd+uct	+.029	−.005	+.031	−.005	+.029	−.005	**+.036**	−.005	.378

Tabelle 8.5.: Differenz ΔPräzision aus Präzisions-Werten für Anreicherung bei WSD mit Wörterbuchgewichtung (dw) sowie den Referenzwerten ohne Anreicherung (siehe Anhang). Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.

dass die weitere Wissensanreicherung durch falsch aufgelöste Bedeutungen verfälscht wird und somit zum Teil semantisch unverwandte Begriffe angereichert werden.

In den Tabellen 8.3 und 8.4 kann abgelesen werden, inwiefern sich die F_1-Werte im Vergleich zur Referenzimplementierung ohne Anreicherung verändert haben. Es werden dabei beim Einbeziehen von Wörterbuchgewichten (Tabelle 8.3) Verbesserungen von bis zu +4.6 % und Verschlechterungen von bis zu −3.2 % erreicht. Für das Nichtbeachten der Wörterbuchgewichte (Tabelle 8.4) werden Veränderungen zwischen −3.6 % und +3.7 % erreicht. Es fällt auf, dass das Miteinbeziehen von Wörterbuchgewichten („dw“) tendenziell größere F_1-Verbesserungen bewirkt als das Nichtbeachten. Betrachtet man jedoch die Präzisions-Verbesserungen (siehe Tabellen 8.5 und 8.6), so zeichnet sich ein anderes Bild: Ohne Wörterbuchgewichte werden Verbesserungen bis zu +11.5 % erreicht, mit Gewichten +11.0 %. Auch hier kann angenommen werden, dass dies an der fehlenden Neu-Kalibrierung der Schwellenwerte m und f liegt. In den Tabellen 8.7 und 8.8 sind die F_1-Verbesserungen aufgeführt, die sich ergeben, wenn für die Konfiguration einer jeden Tabellenzelle die optimalen Schwellenwerte $m \in [0.53, 0.63]$ und $f \in [0.4, 0.5]$ gewählt werden, um die F_1-Werte zu maximieren. Diese Werte können als obere Schranke für die optimal-mögliche Verbesserung betrachtet werden. Dort ist zu sehen, dass in Tabelle 8.8 („ohne dw“) in drei Datensatz-Zeilen die Zeilenmaxima größer sind als in Tabelle 8.7 („mit dw“). In den Datensatz-Zeilen, in denen umgekehrt die Zeilenmaxima für „mit dw“ größer sind als die Zeilenmaxima für „ohne dw“, ist

ΔPräzision (ohne dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	+.100	+.073	**+.105**	+.072	+.090	+.027	+.088	+.025	.158
+mc+uct(+cd)	+.114	+.022	**+.115**	+.021	+.105	−.004	+.100	−.005	.164
eTour +mc	**+.013**	−.009	**+.013**	−.010	+.008	−.012	+.002	−.011	.276
+mc+cd	**+.018**	−.006	+.015	−.006	+.012	−.007	+.008	−.007	.270
+mc+uct	+.040	−.020	**+.048**	−.020	+.038	−.020	+.032	−.021	.374
+mc+cd+uct	+.036	−.017	**+.047**	−.018	+.043	−.018	+.042	−.019	.378

Tabelle 8.6.: Differenz ΔPräzision aus Präzisions-Werten für Anreicherung bei WSD ohne Wörterbuchgewichtung (dw) sowie den Referenzwerten ohne Anreicherung (siehe Anhang). Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.

ΔF_1^{OPT} (mit dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	**+.081**	+.067	+.075	+.067	+.066	+.039	+.062	+.040	.159
+mc+uct(+cd)	**+.042**	+.020	+.036	+.023	+.024	+.003	+.024	+.005	.178
eTour +mc	+.025	+.018	+.028	+.017	+.030	+.022	**+.035**	+.021	.351
+mc+cd	+.033	+.025	+.032	+.025	**+.039**	+.028	+.037	+.028	.343
+mc+uct	−.014	**+.004**	−.019	+.003	−.014	−.002	−.007	−.001	.470
+mc+cd+uct	−.007	−.001	−.008	−.001	−.006	**+.005**	−.002	+.004	.473
∅ +mc	**+.053**	+.043	+.052	+.042	+.048	+.031	+.049	+.031	
∅ +mc+cd	**+.057**	+.046	+.054	+.046	+.053	+.034	+.050	+.034	
∅ +mc+uct	**+.014**	+.012	+.009	+.013	+.005	+.001	+.009	+.002	
∅ +mc+cd+uct	**+.018**	+.010	+.014	+.011	+.009	+.004	+.011	+.005	

Tabelle 8.7.: Differenz ΔF_1^{OPT} aus mit pro Tabellenzelle optimalen Schwellenwerten berechneten F_1-Werten für Anreicherung bei WSD mit Wörterbuchgewichtung (dw) sowie den Referenzwerten ohne Anreicherung (siehe Anhang). Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.

der betragliche Unterschied kleiner gleich 0.2 %. Dafür sind in den übrigen Datensatz-Zeilen die Unterschiede groß genug, sodass in den Durchschnittszeilen die Zeilenmaxima für „ohne dw“ über den Zeilenmaxima für „mit dw“ liegen. Es lässt sich also ablesen, dass das Nichtbeachten der Wörterbuchgewichte größeres Verbesserungspotenzial hat als das Miteinbeziehen.

Besonders auffallend ist bei eTour, dass sich für „+mc+uct“ und „+mc+cd+uct“ die schwellenwertoptimalen F_1-Werte (wie auch die standardmäßigen F_1-Werte) ausschließlich verschlechtern. Dies kann damit erklärt werden, dass das gewählte Verfahren für die Einbindung der Anreicherungen in FTLR für die Option „+uct“ schlechter geeignet ist. Mithilfe dieser Option wählt FTLR Begriffe aus Anwendungsfallschablonen gezielter aus, wodurch davon auszugehen ist, dass die entsprechenden Bags-of-Embeddings weniger Einbettungen enthalten. Die FTLR-Einbindung über das Hinzufügen der angereicherten Wörter zu bestehenden Wortlisten könnte die Bags-of-Embeddings auf der Quelltextseite deutlich größer machen als diejenigen auf der Anforderungsseite. Dadurch kann die Zuordnung der entsprechenden Bags-of-Embeddings ungenauer werden. Eine andere Art der FTLR-Einbindung wie zum Beispiel das Erstellen von separaten Bags-of-Embeddings für die angereicherten Begriffe (siehe Abschnitt 5.3.2) könnte hier Verbesserung bewirken.

ΔF_1^{OPT} (ohne dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	**+.094**	+.063	+.083	+.063	+.076	+.044	+.062	+.044	.159
+mc+uct(+cd)	+.053	+.015	**+.057**	+.016	+.048	+.004	+.039	+.005	.178
eTour +mc	**+.033**	+.025	+.022	+.028	+.027	**+.033**	+.016	+.032	.351
+mc+cd	+.037	+.029	+.024	+.027	+.039	**+.045**	+.026	+.041	.343
+mc+uct	−.006	−.002	−.005	+.001	−.008	**+.003**	−.011	−.002	.470
+mc+cd+uct	−.012	−.001	−.008	−.006	−.005	+.002	−.005	**+.004**	.473
∅ +mc	**+.064**	+.044	+.053	+.046	+.052	+.039	+.039	+.038	
∅ +mc+cd	**+.066**	+.046	+.054	+.045	+.058	+.045	+.044	+.043	
∅ +mc+uct	+.024	+.007	**+.026**	+.009	+.020	+.004	+.014	+.002	
∅ +mc+cd+uct	+.021	+.007	**+.025**	+.005	+.022	+.003	+.017	+.005	

Tabelle 8.8.: Differenz ΔF_1^{OPT} aus mit pro Tabellenzelle optimalen Schwellenwerten berechneten F_1-Werten für Anreicherung bei WSD ohne Wörterbuchgewichtung (dw) sowie den Referenzwerten ohne Anreicherung (siehe Anhang). Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.

Insgesamt kann angenommen werden, dass der Verzicht auf die DBpedia-Wörterbuchgewichte die bessere Option ist. Als nächstes soll untersucht werden, welche Anreicherungs-Strategie die bessere ist. Vergleicht man die schwellenwertoptimale F_1-Differenz (ohne dw) in Tabelle 8.8 in den „top connectivity“-Spalten der Durchschnittszeilen mit denen für „max coverage strategy“, so sind die ersteren Werte für eAnci größer als die entsprechenden zweiten. In Tabelle 8.6 sind durchweg alle Zeilenmaxima der Präzisionsdifferenzen in den „top-connectivity“-Spalten zu finden. Dies könnte darauf hindeuten, dass die „top connectivity strategy“ die besser geeignetere Anreicherungs-Strategie ist. Allerdings verhalten sich die eTour-Zeilen anders als der Rest der Tabelle. Aufgrund der unsicheren Datenlage kann keine definitive Aussage über das besser geeignetere Verfahren gemacht werden. Eine Evaluation ausgehend von Bedeutungen aus einer händisch erstellten Musterlösung sowie eine vollständige Neu-Kalibrierung der FTLR-Schwellenwerte könnte hier Abhilfe schaffen.

Unter der vorsichtigen Annahme, dass die „top connectivity strategy“ die bessere Option ist, soll nun der Einfluss des Hinzufügens der ursprünglichen Bedeutungen untersucht werden. Während in Tabelle 8.8 mit den schwellenwertoptimalen F_1-Werten (ohne dw) das Hinzufügen der ursprünglichen Bedeutungen für eAnci deutlich vor dem Weglassen liegt, sind bei eTour die Werte nicht eindeutig. Für die ersten zwei eTour-Zeilen liegt zumindest die „+uB“-Spalte für „Duplikate verhindern“ deutlich vor den anderen drei Spalten. Für die letzten zwei eTour-Zeilen „+mc+uct“ und „+mc+cd+uct“ enthalten jeweils die „−uB“-Spalten die größeren Werte. Die Präzisionsverbesserungen (ohne dw) in Tabelle 8.6 zeigen, dass in den vier „top-connectivity“-Spalten die „+uB“-Werte für eTour deutlich vor denen für „−uB“ liegen. Aufgrund der unsicheren Datenlage kann nur vermutet werden, dass das Hinzufügen von ursprünglichen Bedeutungen besser geeignet ist.

Unter der vorsichtigen Annahme, dass das Hinzufügen von ursprünglichen Bedeutungen besser geeignet ist, soll nun untersucht werden, ob das Verhindern oder das Erlauben von Duplikaten besser geeignet ist. Tabelle 8.8 mit den schwellenwertoptimalen F_1-Differenzen (ohne dw) zeigt, dass bei beiden Datensätzen für die Optionen „+mc“ und „+mc+cd“ das Verhindern von Duplikaten größeres Verbesserungspotenzial hat, für die übrigen Optionen hingegen das Erlauben von Duplikaten leicht vorne liegt. Ähnliches ist auch in Tabelle 8.6 ablesbar: Größtenteils liegen die Präzisionsverbesserungen sehr nahe beieinander. Lediglich für eTour „+mc+cd+uct“ verbessert das Erlauben die Präzision um +1.1 % mehr als das Verhindern. Mit den vorliegenden Daten kann

		mit Wissensanreicherung					Baseline				
		Präz.	Ausb.	F_1	(F_1^{OPT})	MAP	Präz.	Ausb.	F_1	(F_1^{OPT})	MAP
eAnci	+mc(+cd)	**.264**	.129	**.173**	(**.242**)	**.149**	.158	**.159**	.159	(.201)	.135
	+mc+uct(+cd)	**.279**	.141	**.187**	(**.235**)	**.145**	.164	**.194**	.178	(.203)	.135
eTour	+mc	**.289**	.399	.335	(.374)	**.286**	.276	**.484**	**.351**	(**.385**)	.285
	+mc+cd	**.285**	.393	.331	(.368)	.287	.270	**.471**	**.343**	(**.385**)	**.296**
	+mc+uct	**.422**	.519	.466	(.466)	.436	.374	**.633**	**.470**	(**.508**)	**.492**
	+mc+cd+uct	**.425**	.513	.465	(.465)	.428	.378	**.630**	**.473**	(**.512**)	**.489**

Tabelle 8.9.: Vergleich der Absolutwerte für mutmaßlich beste Kombination mit der Baseline: WSD ohne Wörterbuchgewichte, Wissensanreicherung mit „top connectivity“, Erlauben von Duplikaten, Hinzunahme der ursprünglichen Bedeutungen. Zeilenmaxima für die jeweiligen Größen sind fett markiert.

keine diesbezügliche Empfehlung getroffen werden. Es ist lediglich eine leichte Tendenz zum Erlauben von Duplikaten zu verzeichnen.

Absolutwerte

Die Resultate der zuvor diskutierten mutmaßlich besten Kombination der Verfahrensoptionen sind in Tabelle 8.9 angegeben. Die Werte gelten für die Standardbelegungen der FTLR-Schwellenwerte. Die schwellenwertoptimalen F_1-Werte für die Wissensanreicherung und für die Baseline sind in Klammern angegeben. Der Vergleich mit der Baseline zeigt, dass das vorgestellte Verfahren bezüglich Präzision deutlich überlegen ist (bis zu $+11.5\,\%$). Es werden Präzisionswerte zwischen $26.4\,\%$ und $42.5\,\%$ erreicht. Die Ausbeute ist hingegen durchweg schlechter geworden (bis zu $-11.7\,\%$) und bewegt sich zwischen $12.9\,\%$ und $51.9\,\%$. Es sei noch einmal daran erinnert, dass eine Neu-Kalibrierung der Schwellenwerte die Ausbeute-Werte mit den Präzisions-Werten besser ausbalancieren kann, sodass sich womöglich die Ausbeute nicht zu sehr verschlechtert. Die F_1-Werte haben sich auf eAnci verbessert (bis zu $+1.4\,\%$), hingegen auf eTour verschlechtert (bis zu $-1.6\,\%$), dabei werden Werte zwischen $17.3\,\%$ und $46.5\,\%$ erreicht. Für die MAP-Werte sind auf eAnci Verbesserungen von bis zu $+1.5\,\%$ zustande gekommen. Für alle Optionen außer „+mc“ sind die MAP-Werte auf eTour gesunken (bis zu $-6.1\,\%$). Dabei werden Werte zwischen $14.5\,\%$ und $43.6\,\%$ erreicht.

Im Anhang sind zu den hier vorgestellten Metriken vollständige Absolutwert- und Delta-Tabellen zu finden.

8.4. Fazit

Von den in Kapitel 5 vorgestellten Zielfragen kann nur Zielfrage 3 beantwortet werden: Beim Verfahren von Weigelt et al. [Wei+19] (siehe Abschnitt 4.3) können sowohl die „top connectivity strategy“ als auch die „max coverage strategy“ die Anforderungsrückverfolgung bezüglich Präzision verbessern. Eine Neu-Kalibrierung der FTLR-Schwellenwerte könnte bewirken, dass auch F_1-Werte sich signifikant verbessern. Die „top connectivity strategy“ ist vermutlich etwas besser geeignet. Die Hinzunahme von aufgelösten Bedeutungen zur Erstellung von Bags-of-Embeddings wirkt sich vermutlich positiv auf die F_1-Verbesserungen aus. Dabei können sowohl das Verhindern als auch das Erlauben von Duplikaten gute Ergebnisse erzielen, wobei tendenziell das Erlauben von Duplikaten einen größeren Verbesserungsspielraum hat.

Zielfrage 2 kann aufgrund der unsicheren Datenlage nur dahingehend beantwortet werden, dass die Anwendung von DBpedia als Wissensnetz vermutlich nicht von einer PPR-Gewichtung unter Zuhilfenahme von Verlinkungs-Statistiken profitiert. Es kann zumindest angesichts der Präzisionsverbesserungen davon ausgegangen werden, dass eine DBpedia-basierte Bedeutungsauflösung

von Begriffen im Quelltext mittels deklarativem Kontext einen ausreichenden Nutzen für die Anforderungsrückverfolgung hat.

9. Zusammenfassung und Ausblick

Diese Arbeit hatte zum Ziel, die Möglichkeiten von Wissensanreicherung für Begriffe im Quelltext zur Verbesserung worteinbettungsbasierter Anforderungsrückverfolgung zu untersuchen. Dabei wurde ein zweiteiliges Verfahren erarbeitet, bestehend aus Wortbedeutungsauflösung und Wissensanreicherung.

Zunächst wurde analysiert, welche Begriffe im Quelltext sich für eine Anreicherung mit Wissen eignen und welche Informationen im Quelltext dafür nutzbar sind. Dabei wurden notwendige Vorverarbeitungsschritte wie Tokenisierung, Lemmatisierung und Wortarterkennung erarbeitet, um natürlichsprachliche Begriffe aus dem Quelltext zu extrahieren und für die weitere Verarbeitung zu vereinheitlichen.

Für die Wortbedeutungsauflösung wurden verschiedene Verfahren für natürlichsprachliche Texte analysiert und untersucht, inwieweit sich diese auf Quelltext anwenden lassen. Dabei stellte sich heraus, dass wissensbasierte Ansätze gegenüber anderen Ansätzen verschiedene Vorteile wie z. B. Domänenflexibilität bieten, was ein relevantes Kriterium für die Anwendung in der Anforderungsrückverfolgung darstellt. Als Werkzeug für die Bedeutungsauflösung wurde UKB von Agirre et al. [ALS14] gewählt. Dazu wurden verschiedene Wissensquellen wie WordNet, DBpedia oder BabelNet auf ihre Eignung zur wissensbasierten Wortbedeutungsauflösung untersucht, wobei DBpedia als Wissensnetz gewählt wurde. Zusätzlich wurde analysiert, inwieweit sich der linguistische Kontext-Begriff auf Quelltext übertragen lässt und dabei verschiedene Kontextarten für die Bedeutungsauflösung vorgestellt.

Für die weitere Wissensanreicherung wurden verschiedene Themenbeschriftungs- und Themenmodellierungs-Verfahren untersucht mit dem Ziel, mithilfe aufgelöster Wortbedeutungen semantisch verwandte Begriffe zu den Ausgangswörtern zu finden. Dabei wurde das wissensbasierte Themenbeschriftungs-Verfahren nach Weigelt et al. [Wei+19] ausgewählt, das für kurze Aussagen mit wenig Inhalt ausgelegt wurde und zu einer gegebenen Menge an Wortbedeutungen anhand des verwendeten Wissensnetzes Themenbeschriftungen ermittelt. Diese aus DBpedia gewonnenen Themenbeschriftungen wurden dann in FTLR [Hey+21], ein Projekt zur worteinbettungsbasierten Anforderungsrückverfolgung, eingebunden. Die so angereicherten Wörter werden bereits vorhandenen Quelltextwortmengen hinzugefügt und dann als Worteinbettungen für die Anforderungsrückverfolgung genutzt.

Das so entworfene Verfahren wurde mit den FTLR-Evaluationsdatensätzen evaluiert, um die Verbesserung der Anforderungsrückverfolgung mittels Wissensanreicherung zu untersuchen. Dabei wurde das gesamte Verfahren am Stück getestet, ohne die Zwischenschritte mithilfe von Musterlösungen gesondert zu evaluieren. Eine Analyse der Ergebnisse ergab eine mutmaßlich

beste Verfahrenskonfiguration. Bei dieser wurden vor allem Verbesserungen bis zu +11.5 % bei der Präzision festgestellt, was jedoch mit einer deutlicheren Verschlechterung der Ausbeute einherging. Trotzdem konnten auf dem eAnci-Datensatz F_1-Verbesserungen zwischen +1.0 % und +1.4 % ermittelt werden, was für auf dieser Konfiguration optimal gewählte FTLR-Schwellenwerte auf bis zu +8.3 % anstieg. Auf dem eTour-Datensatz hingegen wurden nur Präzisionsverbesserungen, aber keine F_1-Verbesserungen festgestellt, da die Ausbeute sich deutlich verschlechtert hatte. Für auf dieser Konfiguration optimal gewählte FTLR-Schwellenwerte konnten jedoch auch auf eTour F_1-Verbesserungen bis zu +2.4 % festgestellt werden.

Es ist anzunehmen, dass bei schlechterer Qualität der Bedeutungsauflösung auch das anschließende Wissensanreicherungs-Verfahren vermehrt unspezifische oder unpassende Themenbeschriftungen ermittelt. Um die Qualität der Bedeutungsauflösung zu erhöhen, kann beispielsweise die DBpedia-Wissensbasis noch weiter vorgefiltert werden. Heuristiken zum Erkennen nicht zielführender DBpedia-Einträge wie beispielsweise populärkultureller Einträge oder Einträge über Personen können die Wahrscheinlichkeit senken, falsch-positiv zugeordnete Wortbedeutungen zu erhalten. Auch kann versucht werden, eine bessere Alternative für die aus DBpedia extrahierten UKB-Wörterbuchgewichte zu finden. Techniken wie Multiwort-Erkennung oder Abkürzungsauflösung können als weitere Vorverarbeitungsschritte eingesetzt werden. Ebenso kann eine Evaluation verschiedener Kontextarten und ihrer Kombinationen für die Bedeutungsauflösung von Begriffen im Quelltext von Nutzen sein.

Auch auf dem Gebiet der Wissensanreicherung blieben viele Ansätze unerforscht. Beispielsweise wurden verschiedene Informationsgehalt-basierte Verfahren im Rahmen dieser Arbeit nicht eingesetzt, da DBpedia aufgrund fehlender kompatibler Auftrittswahrscheinlichkeits-Daten dafür nicht geeignet ist. Das hier implementierte Anreicherungs-Verfahren kann zudem genau wie die Bedeutungsauflösung von weiterer Vorfilterung von DBpedia-Einträgen profitieren.

Literatur

[AA07] Alfred V. Aho und Alfred V. Aho, Hrsg. *Compilers: principles, techniques, & tools*. 2nd ed. Boston: Pearson/Addison Wesley, 2007. 1009 S. ISBN: 978-0-321-48681-3 (zitiert auf S. 11).

[AGL21] Rakan Alanazi, Gharib Gharibi und Yugyung Lee. „Facilitating program comprehension with call graph multilevel hierarchical abstractions". In: *Journal of Systems and Software* 176 (Juni 2021), S. 110945. ISSN: 01641212. DOI: 10.1016/j.jss.2021.110945 (zitiert auf S. 23, 44, 55).

[ALS14] Eneko Agirre, Oier López de Lacalle und Aitor Soroa. „Random Walks for Knowledge-Based Word Sense Disambiguation". In: *Computational Linguistics* 40.1 (1. März 2014), S. 57–84. ISSN: 0891-2017. DOI: 10.1162/COLI_a_00164 (zitiert auf S. 19, 38, 39, 44, 54, 55, 60, 67, 77).

[ALS18] Eneko Agirre, Oier López de Lacalle und Aitor Soroa. „The risk of sub-optimal use of Open Source NLP Software: UKB is inadvertently state-of-the-art in knowledge-based WSD". In: *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Melbourne, Australia: Association for Computational Linguistics, Juli 2018, S. 29–33. DOI: 10.18653/v1/W18-2505 (zitiert auf S. 20, 39, 40).

[Bab] BabelNet. *News | BabelNet*. URL: https://babelnet.org/news (besucht am 10. 02. 2022) (zitiert auf S. 39).

[Bar20] Thomas Bartel. „Multiwort-Bedeutungsauflösung für Anforderungen". Bachelorarb. Karlsruher Institut für Technologie (KIT) – IPD Tichy, März 2020. URL: https://code.ipd.kit.edu/hey/indirect/wikis/Theses/bartel_ba (zitiert auf S. 60).

[BLB16] Shraey Bhatia, Jey Han Lau und Timothy Baldwin. „Automatic Labelling of Topics with Neural Embeddings". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Dez. 2016, S. 953–963. URL: https://aclanthology.org/C16-1091 (besucht am 04. 02. 2022) (zitiert auf S. 25).

[BN20] Michele Bevilacqua und Roberto Navigli. „Breaking Through the 80% Glass Ceiling: Raising the State of the Art in Word Sense Disambiguation by Incorporating Knowledge Graph Information". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics, 2020, S. 2854–2864. DOI: 10.18653/v1/2020.acl-main.255 (zitiert auf S. 21, 36, 39).

[BP98] Sergey Brin und Lawrence Page. „The anatomy of a large-scale hypertextual Web search engine". In: *Computer Networks and ISDN Systems*. Proceedings of the Seventh International World Wide Web Conference 30.1 (1. Apr. 1998), S. 107–117. ISSN: 0169-7552. DOI: 10.1016/S0169-7552(98)00110-X (zitiert auf S. 19).

[Buß02] Hadumod Bußmann. *Lexikon der Sprachwissenschaft*. 3. Aufl. 2002. ISBN: 3-520-45203-0 (zitiert auf S. 5, 6).

[Dev+19] Jacob Devlin et al. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. Association for Computational Linguistics, Juni 2019, S. 4171–4186. DOI: 10.18653/v1/N19-1423 (zitiert auf S. 21).

[Eur20] Felix Eurich. „Entwurf und Aufbau einer semantischen Repräsentation von Quelltext“. Magisterarb. Karlsruher Institut für Technologie (KIT) – IPD Tichy, Jan. 2020. URL: https://code.ipd.kit.edu/hey/indirect/wikis/Theses/eurich_ma (zitiert auf S. 18).

[Fel06] Christiane Fellbaum. „WordNet(s)“. In: *Keith Brown (ed.), Encyclopedia of Language and Linguistics, 2nd Edition* 13 (2006), S. 665–700. URL: https://iaoa.org/isc2012/docs/encycloped.article.pdf (besucht am 21. 02. 2022) (zitiert auf S. 36).

[Got+12] Orlena Gotel et al. „The Grand Challenge of Traceability (v1.0)“. In: *Software and Systems Traceability*. Hrsg. von Jane Cleland-Huang, Orlena Gotel und Andrea Zisman. London: Springer London, 2012, S. 343–409. ISBN: 978-1-4471-2238-8 978-1-4471-2239-5. DOI: 10.1007/978-1-4471-2239-5_16 (zitiert auf S. 1).

[Gro21] Gilbert Groten. „Automatisches Auflösen von Abkürzungen in Quelltext“. Magisterarb. Karlsruher Institut für Technologie (KIT) – IPD Tichy, Apr. 2021. URL: https://code.ipd.kit.edu/hey/indirect/wikis/Theses/groten_ma (zitiert auf S. 18).

[Hey+21] Tobias Hey et al. „Improving Traceability Link Recovery Using Fine-grained Requirements-to-Code Relations“. In: *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (2021), S. 12–22. DOI: 10.1109/ICSME52107.2021.00008 (zitiert auf S. 15, 18, 27, 28, 30–34, 48, 51, 57, 59, 65–68, 77).

[Hey19] Tobias Hey. „INDIRECT: Intent-Driven Requirements-to-Code Traceability“. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). Mai 2019, S. 190–191. DOI: 10.1109/ICSE-Companion.2019.00078 (zitiert auf S. 1, 17, 59).

[HKT21] Tobias Hey, Jan Keim und Walter F Tichy. „Knowledge-based Sense Disambiguation of Multiword Expressions in Requirements Documents“. In: *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. Sep. 2021, S. 70–76. DOI: 10.1109/REW53955.2021.00017 (zitiert auf S. 18).

[HNG19] Christian Hadiwinoto, Hwee Tou Ng und Wee Chung Gan. „Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations“. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019, S. 5296–5305. DOI: 10.18653/v1/D19-1533 (zitiert auf S. 21, 39).

[Hof+12] Johannes Hoffart et al. „KORE: keyphrase overlap relatedness for entity disambiguation“. In: *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*. the 21st ACM international conference. Maui, Hawaii, USA: ACM Press, 2012, S. 545. ISBN: 978-1-4503-1156-4. DOI: 10.1145/2396761.2396832 (zitiert auf S. 25).

[ISO10] „ISO/IEC/IEEE International Standard – Systems and software engineering – Vocabulary“. In: *ISO/IEC/IEEE 24765:2010(E)* (2010), S. 1–418. DOI: 10.1109/IEEESTD.2010.5733835 (zitiert auf S. 15).

[JM21] Daniel Jurafsky und James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing,Computational Linguistics, and Speech Recognition.* Third Edition draft. 21. Sep. 2021. URL: https://web.stanford.edu/~jurafsky/slp3/ed3book_sep212021.pdf (besucht am 02. 11. 2021) (zitiert auf S. 3–10, 66).

[Koc22] Jonas Koch. „Verbesserung von Worteinbettungs-basierter Rückverfolgbarkeitsanalyse durch Konzeptwissen“. Bachelorarb. Karlsruher Institut für Technologie (KIT) – IPD Tichy, 25. März 2022. URL: https://code.ipd.kit.edu/hey/indirect/wikis/Theses/koch_ba (besucht am 11. 04. 2022) (zitiert auf S. 47–49).

[Kua+12] Hongyu Kuang et al. „Do data dependencies in source code complement call dependencies for understanding requirements traceability?“ In: *2012 28th IEEE International Conference on Software Maintenance (ICSM).* 2012 28th IEEE International Conference on Software Maintenance (ICSM). Sep. 2012, S. 181–190. DOI: 10.1109/ICSM.2012.6405270 (zitiert auf S. 11, 13, 22, 44).

[Leh+15] Jens Lehmann et al. „DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia“. In: *Semantic Web* 6.2 (2015), S. 167–195. ISSN: 15700844. DOI: 10.3233/SW-140134 (zitiert auf S. 8, 36, 37).

[Les86] Michael Lesk. „Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone“. In: *Proceedings of the 5th Annual International Conference on Systems Documentation.* SIGDOC '86. New York, NY, USA: Association for Computing Machinery, 1986, S. 24–26. ISBN: 0-89791-224-1. DOI: 10.1145/318723.318728 (zitiert auf S. 9).

[Lin98] Dekang Lin. „An Information-Theoretic Definition of Similarity“. In: In Proceedings of the 15th International Conference on Machine Learning. 1998, S. 9. DOI: 10.1.1.55.1832 (zitiert auf S. 14, 15, 24, 46).

[Man+14] Christopher D. Manning et al. „The Stanford CoreNLP Natural Language Processing Toolkit“. In: *Association for Computational Linguistics (ACL) System Demonstrations.* 2014, S. 55–60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010 (zitiert auf S. 59).

[McB+14] Paul W. McBurney et al. „Improving topic model source code summarization“. In: *Proceedings of the 22nd International Conference on Program Comprehension.* ICPC 2014. New York, NY, USA: Association for Computing Machinery, 2. Juni 2014, S. 291–294. ISBN: 978-1-4503-2879-1. DOI: 10.1145/2597008.2597793 (zitiert auf S. 23, 46).

[MM16] Paul W. McBurney und Collin McMillan. „Automatic Source Code Summarization of Context for Java Methods“. In: *IEEE Transactions on Software Engineering* 42.2 (Feb. 2016), S. 103–119. ISSN: 1939-3520. DOI: 10.1109/TSE.2015.2465386 (zitiert auf S. 22, 44).

[MN15] Andrea Moro und Roberto Navigli. „SemEval-2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking“. In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015).* Denver, Colorado: Association for Computational Linguistics, Juni 2015, S. 288–297. DOI: 10.18653/v1/S15-2049 (zitiert auf S. 20, 21, 39).

[Mor+13] Laura Moreno et al. „Automatic generation of natural language summaries for Java classes“. In: *2013 21st International Conference on Program Comprehension (ICPC).* 2013 21st International Conference on Program Comprehension (ICPC). Mai 2013, S. 23–32. DOI: 10.1109/ICPC.2013.6613830 (zitiert auf S. 22, 41, 44).

[MRN14] Andrea Moro, Alessandro Raganato und Roberto Navigli. „Entity Linking meets Word Sense Disambiguation: a Unified Approach“. In: *Transactions of the Association for Computational Linguistics* 2 (Dez. 2014), S. 231–244. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00179 (zitiert auf S. 20, 38, 39, 54).

[NJV13] Roberto Navigli, David Jurgens und Daniele Vannella. „SemEval-2013 Task 12: Multilingual Word Sense Disambiguation“. In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA: Association for Computational Linguistics, Juni 2013, S. 222–231. URL: https://aclanthology.org/S13-2040 (zitiert auf S. 20, 21, 39).

[NP12] Roberto Navigli und Simone Paolo Ponzetto. „BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network“. In: *Artificial Intelligence* 193 (Dez. 2012), S. 217–250. ISSN: 00043702. DOI: 10.1016/j.artint.2012.07.001 (zitiert auf S. 36).

[Oraa] Oracle. *Code Conventions for the Java Programming Language: 9. Naming Conventions*. URL: https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html (besucht am 02. 03. 2022) (zitiert auf S. 30–32).

[Orab] Oracle. *Documentation Comment Specification for the Standard Doclet*. URL: https://docs.oracle.com/javase/9/docs/specs/doc-comment-spec.html (besucht am 20. 12. 2021) (zitiert auf S. 10, 30).

[Orac] Oracle. *Java Language Specification*. URL: https://docs.oracle.com/javase/specs/jls/se9/html/ (besucht am 18. 12. 2021) (zitiert auf S. 10, 13, 30).

[Orad] Oracle. *Predefined Annotation Types (The Java™ Tutorials > Learning the Java Language > Annotations)*. URL: https://docs.oracle.com/javase/tutorial/java/annotations/predefined.html (besucht am 02. 03. 2022) (zitiert auf S. 29).

[Pria] University of Princeton. *Documentation: wninput(5WN) | WordNet*. URL: https://wordnet.princeton.edu/documentation/wninput5wn (besucht am 28. 02. 2022) (zitiert auf S. 37).

[Prib] University of Princeton. *News | WordNet*. URL: https://wordnet.princeton.edu/news-0 (besucht am 22. 02. 2022) (zitiert auf S. 36).

[Pric] University of Princeton. *WordNet | A Lexical Database for English*. URL: https://wordnet.princeton.edu/ (besucht am 01. 03. 2022) (zitiert auf S. 38).

[RCN17] Alessandro Raganato, Jose Camacho-Collados und Roberto Navigli. „Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison“. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. EACL 2017. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, S. 99–110. URL: https://aclanthology.org/E17-1010 (besucht am 10. 02. 2022) (zitiert auf S. 39).

[Som16] Ian Sommerville. *Software engineering*. 10. ed., global ed. Always learning. Boston Munich: Pearson, 2016. 810 S. ISBN: 978-1-292-09613-1 (zitiert auf S. 31, 32).

[SP14] Michael Schuhmacher und Simone Paolo Ponzetto. „Knowledge-based graph document modeling“. In: *Proceedings of the 7th ACM international conference on Web search and data mining*. WSDM 2014: Seventh ACM International Conference on Web Search and Data Mining. New York New York USA: ACM, 24. Feb. 2014, S. 543–552. ISBN: 978-1-4503-2351-2. DOI: 10.1145/2556195.2556250 (zitiert auf S. 25, 48).

[Ull20] Christian Ullenboom. *Java ist auch eine Insel: Einführung, Ausbildung, Praxis*. 15. Aufl. Rheinwerk Verlag, 1. Juli 2020. ISBN: 978-3-8362-7737-2. URL: http://openbook.rheinwerk-verlag.de/javainsel/ (besucht am 20. 12. 2021) (zitiert auf S. 11).

[Wei+19] Sebastian Weigelt et al. „Unsupervised Multi-Topic Labeling for Spoken Utterances“. In: *2019 IEEE International Conference on Humanized Computing and Communication (HCC)*. 2019 IEEE International Conference on Humanized Computing and Communication (HCC). Sep. 2019, S. 38–45. DOI: 10.1109/HCC46620.2019.00014 (zitiert auf S. 24, 37, 47, 48, 56, 63, 67, 74, 77).

[WHT17] Sebastian Weigelt, Tobias Hey und Walter F. Tichy. „Context Model Acquisition from Spoken Utterances“. In: *SEKE 2017. The 29th International Conference on Software Engineering & Knowledge Engineering, Pittsburgh, PA, July 5 - 7, 2017* (2017), S. 201. DOI: 10.18293/SEKE2017-083 (zitiert auf S. 23, 24, 46, 47).

[Wik] *Wikipedia:Article titles – Wikipedia*. URL: https://en.wikipedia.org/wiki/Wikipedia:Article_titles#Article_title_format (besucht am 21. 11. 2021) (zitiert auf S. 8).

[WT15] Sebastian Weigelt und Walter F. Tichy. „ProNat: An Agent-Based System Design for Programming in Spoken Natural Language“. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE). Florence, Italy: IEEE, Mai 2015, S. 819–820. ISBN: 978-1-4799-1934-5. DOI: 10.1109/ICSE.2015.264 (zitiert auf S. 18).

[WWF20] Yinglin Wang, Ming Wang und Hamido Fujita. „Word Sense Disambiguation: A comprehensive knowledge exploitation framework“. In: *Knowledge-Based Systems* 190 (Feb. 2020), S. 105030. ISSN: 09507051. DOI: 10.1016/j.knosys.2019.105030 (zitiert auf S. 20, 38–40).

Anhang

A. Evaluations-Ergebnisse

Präzision (mit dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	**.268**	.249	.254	.252	.241	.199	.214	.200	.158
+mc+uct(+cd)	**.255**	.187	.242	.189	.239	.163	.209	.164	.164
eTour +mc	.296	.258	**.312**	.257	.290	.258	.290	.257	.276
+mc+cd	.292	.255	**.312**	.254	.284	.255	.290	.255	.270
+mc+uct	.394	.366	.401	.366	.395	.366	**.403**	.366	.374
+mc+cd+uct	.408	.373	.409	.373	.407	.373	**.414**	.373	.378

Präzision (ohne dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	.258	.232	**.264**	.231	.248	.185	.246	.184	.158
+mc+uct(+cd)	.278	.185	**.279**	.185	.269	.160	.264	.159	.164
eTour +mc	**.289**	.267	**.289**	.266	.284	.264	.278	.265	.276
+mc+cd	**.288**	.264	.285	.264	.282	.263	.278	.263	.270
+mc+uct	.415	.354	**.422**	.354	.412	.354	.407	.353	.374
+mc+cd+uct	.414	.362	**.425**	.361	.421	.360	.421	.360	.378

Tabelle A.1.: oben: Präzisions-Werte für Anreicherung bei WSD mit/ohne Wörterbuchgewichtung (dw).
unten: Differenz ΔPräzision derselben Präzisions-Werte in Bezug auf die BL.
„+uB“: mit ursprünglichen Bedeutungen, „−uB“: ohne.
„Dupl. verh./erlaub.“: Duplikate verhindern/erlauben.
„BL“: Baseline (dt. *Referenzwert* (ohne Anreicherung)).
Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.
Die Option +cd ist bei eAnci wirkungslos und daher eingeklammert.

ΔPräzision (mit dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	+**.110**	+.090	+.096	+.093	+.082	+.041	+.055	+.042
+mc+uct(+cd)	+**.092**	+.023	+.078	+.026	+.075	−.000	+.045	.000
eTour +mc	+.020	−.018	+**.037**	−.019	+.014	−.018	+.014	−.019
+mc+cd	+.022	−.015	+**.042**	−.016	+.014	−.015	+.020	−.015
+mc+uct	+.020	−.008	+.027	−.008	+.020	−.008	+**.028**	−.008
+mc+cd+uct	+.029	−.005	+.031	−.005	+.029	−.005	+**.036**	−.005

ΔPräzision (ohne dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	+.100	+.073	+**.105**	+.072	+.090	+.027	+.088	+.025
+mc+uct(+cd)	+.114	+.022	+**.115**	+.021	+.105	−.004	+.100	−.005
eTour +mc	+**.013**	−.009	+**.013**	−.010	+.008	−.012	+.002	−.011
+mc+cd	+**.018**	−.006	+.015	−.006	+.012	−.007	+.008	−.007
+mc+uct	+.040	−.020	+**.048**	−.020	+.038	−.020	+.032	−.021
+mc+cd+uct	+.036	−.017	+**.047**	−.018	+.043	−.018	+.042	−.019

Ausbeute (mit dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	.118	.168	.109	.173	.116	.173	.106	**.175**	.159
+mc+uct(+cd)	.129	.196	.120	**.201**	.129	.196	.116	.198	.194
eTour +mc	.399	.474	.390	.471	.396	.481	.386	.477	**.484**
+mc+cd	.383	.464	.380	.461	.377	**.474**	.377	**.474**	.471
+mc+uct	.494	.584	.494	.584	.516	.588	.523	.588	**.633**
+mc+cd+uct	.494	.578	.487	.578	.519	.581	.523	.581	**.630**

Ausbeute (ohne dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	.125	**.169**	.129	**.169**	.134	**.169**	.134	**.169**	.159
+mc+uct(+cd)	.127	**.201**	.141	**.201**	.138	.199	.150	.199	.194
eTour +mc	.399	.494	.399	.494	.396	**.497**	.383	**.497**	.484
+mc+cd	.396	.487	.393	.487	.390	**.490**	.380	**.490**	.471
+mc+uct	.519	.562	.519	.565	.526	.571	.510	.571	**.633**
+mc+cd+uct	.506	.555	.513	.558	.526	.562	.516	.562	**.630**

Tabelle A.2.: oben: Ausbeute-Werte für Anreicherung bei WSD mit/ohne Wörterbuchgewichtung (dw).
unten: Differenz ΔAusbeute derselben Ausbeute-Werte in Bezug auf die BL.
„+uB“: mit ursprünglichen Bedeutungen, „−uB“: ohne.
„Dupl. verh./erlaub.“: Duplikate verhindern/erlauben.
„BL“: Baseline (dt. *Referenzwert* (ohne Anreicherung)).
Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.
Die Option +cd ist bei eAnci wirkungslos und daher eingeklammert.

ΔAusbeute (mit dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	−.041	+.009	−.049	+.014	−.042	+.014	−.053	**+.016**
+mc+uct(+cd)	−.065	+.002	−.074	**+.007**	−.065	+.002	−.078	+.004
eTour +mc	−.084	−.010	−.094	−.013	−.088	**−.003**	−.097	−.006
+mc+cd	−.088	−.006	−.091	−.010	−.094	**+.003**	−.094	**+.003**
+mc+uct	−.140	−.049	−.140	−.049	−.117	**−.045**	−.110	**−.045**
+mc+cd+uct	−.136	−.052	−.143	−.052	−.110	**−.049**	−.107	**−.049**

ΔAusbeute (ohne dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	−.034	**+.011**	−.030	**+.011**	−.025	**+.011**	−.025	**+.011**
+mc+uct(+cd)	−.067	**+.007**	−.053	**+.007**	−.056	+.005	−.044	+.005
eTour +mc	−.084	+.010	−.084	+.010	−.088	**+.013**	−.101	**+.013**
+mc+cd	−.075	+.016	−.078	+.016	−.081	**+.019**	−.091	**+.019**
+mc+uct	−.114	−.071	−.114	−.068	−.107	**−.062**	−.123	**−.062**
+mc+cd+uct	−.123	−.075	−.117	−.071	−.104	**−.068**	−.114	**−.068**

F_1 (mit dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	.164	.200	.153	**.205**	.157	.185	.142	.186	.159
+mc+uct(+cd)	.171	.191	.160	**.195**	.167	.178	.149	.179	.178
eTour +mc	.340	.334	.347	.332	.335	.336	.331	.334	**.351**
+mc+cd	.331	.329	**.343**	.327	.324	.331	.328	.331	**.343**
+mc+uct	.438	.450	.443	.450	.447	.451	.455	.451	**.470**
+mc+cd+uct	.446	.454	.444	.454	.456	.454	.462	.454	**.473**

F_1 (ohne dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	.169	**.196**	.173	.195	.174	.177	.174	.176	.159
+mc+uct(+cd)	.174	**.193**	.187	**.193**	.182	.178	.191	.177	.178
eTour +mc	.336	.346	.335	.345	.331	.345	.322	.345	**.351**
+mc+cd	.334	.342	.331	.342	.327	.342	.321	.342	**.343**
+mc+uct	.461	.434	.466	.436	.462	.437	.452	.437	**.470**
+mc+cd+uct	.455	.438	.465	.438	.468	.439	.464	.439	**.473**

Tabelle A.3.: oben: F_1-Werte für Anreicherung bei WSD mit/ohne Wörterbuchgewichtung (dw).
unten: Differenz ΔF_1 derselben F_1-Werte in Bezug auf die BL.
„+uB“: mit ursprünglichen Bedeutungen, „−uB“: ohne.
„Dupl. verh./erlaub.“: Duplikate verhindern/erlauben.
„BL“: Baseline (dt. *Referenzwert* (ohne Anreicherung)).
Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.
Die Option +cd ist bei eAnci wirkungslos und daher eingeklammert.

ΔF_1 (mit dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	+.005	+.042	−.006	**+.046**	−.002	+.026	−.017	+.028
+mc+uct(+cd)	−.006	+.013	−.017	**+.017**	−.010	+.001	−.028	+.002
eTour +mc	−.012	−.017	**−.005**	−.019	−.017	−.016	−.020	−.018
+mc+cd	−.012	−.014	**−.001**	−.016	−.019	−.012	−.016	−.012
+mc+uct	−.032	−.020	−.028	−.020	−.023	−.019	**−.016**	−.019
+mc+cd+uct	−.026	−.019	−.028	−.019	−.016	−.018	**−.011**	−.018

ΔF_1 (ohne dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	+.010	**+.037**	+.014	**+.037**	+.016	+.018	+.015	+.018
+mc+uct(+cd)	−.003	**+.015**	+.010	**+.015**	+.004	−.000	+.014	−.001
eTour +mc	−.016	**−.005**	−.016	−.006	−.020	−.006	−.029	−.006
+mc+cd	−.009	**−.001**	−.013	**−.001**	−.016	**−.001**	−.022	**−.001**
+mc+uct	−.009	−.036	**−.005**	−.035	−.008	−.033	−.018	−.034
+mc+cd+uct	−.017	−.035	−.008	−.034	**−.005**	−.034	−.009	−.034

MAP (mit dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	**.151**	**.151**	.148	**.151**	.144	.143	.142	.143	.135
+mc+uct(+cd)	.145	**.146**	.143	**.146**	.140	.139	.138	.138	.135
eTour +mc	**.303**	.279	.297	.278	.297	.275	.292	.275	.285
+mc+cd	**.307**	.286	.297	.285	.302	.278	.294	.278	.296
+mc+uct	.436	.441	.434	.440	.431	.438	.432	.439	**.492**
+mc+cd+uct	.433	.434	.431	.433	.428	.432	.430	.432	**.489**

MAP (ohne dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	**.153**	.147	.149	.146	.147	.143	.144	.142	.135
+mc+uct(+cd)	**.148**	.146	.145	.146	.142	.141	.141	.141	.135
eTour +mc	**.289**	.270	.286	.270	.284	.272	.280	.271	.285
+mc+cd	.293	.270	.287	.273	.292	.276	.286	.276	**.296**
+mc+uct	.449	.437	.436	.435	.440	.438	.432	.436	**.492**
+mc+cd+uct	.443	.435	.428	.436	.434	.436	.426	.434	**.489**

Tabelle A.4.: oben: MAP-Werte für Anreicherung bei WSD mit/ohne Wörterbuchgewichtung (dw).
unten: Differenz ΔMAP derselben MAP-Werte in Bezug auf die BL.
„+uB“: mit ursprünglichen Bedeutungen, „−uB“: ohne.
„Dupl. verh./erlaub.“: Duplikate verhindern/erlauben.
„BL“: Baseline (dt. *Referenzwert* (ohne Anreicherung)).
Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.
Die Option +cd ist bei eAnci wirkungslos und daher eingeklammert.

ΔMAP (mit dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	**+.016**	**+.016**	+.013	**+.016**	+.010	+.009	+.007	+.008
+mc+uct(+cd)	+.010	**+.012**	+.008	**+.012**	+.005	+.004	+.003	+.003
eTour +mc	**+.017**	−.006	+.012	−.008	+.011	−.010	+.007	−.010
+mc+cd	**+.011**	−.010	+.001	−.011	+.006	−.018	−.002	−.018
+mc+uct	−.057	**−.051**	−.058	−.053	−.061	−.054	−.061	−.053
+mc+cd+uct	−.056	**−.055**	−.058	−.056	−.061	−.057	−.059	−.056

ΔMAP (ohne dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	**+.018**	+.012	+.015	+.012	+.012	+.008	+.009	+.007
+mc+uct(+cd)	**+.013**	+.012	+.010	+.011	+.007	+.006	+.006	+.006
eTour +mc	**+.003**	−.015	+.001	−.015	−.001	−.013	−.005	−.014
+mc+cd	**−.003**	−.026	−.009	−.023	−.004	−.020	−.010	−.020
+mc+uct	**−.044**	−.055	−.056	−.057	−.052	−.054	−.060	−.057
+mc+cd+uct	**−.046**	−.054	−.061	−.053	−.055	−.053	−.063	−.055

F_1^{OPT} (mit dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	**.240**	.226	.233	.226	.225	.197	.220	.198	.159
+mc+uct(+cd)	**.220**	.197	.213	.201	.202	.181	.201	.183	.178
eTour +mc	.376	.370	.379	.368	.381	.373	**.387**	.373	.351
+mc+cd	.376	.369	.376	.368	**.382**	.371	.380	.371	.343
+mc+uct	.456	**.474**	.451	.473	.457	.469	.464	.469	.470
+mc+cd+uct	.466	.472	.464	.472	.466	**.478**	.471	.477	.473

F_1^{OPT} (ohne dw)	top connectivity strategy				max coverage strategy				BL
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.		
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB	
eAnci +mc(+cd)	**.253**	.222	.242	.221	.234	.203	.221	.203	.159
+mc+uct(+cd)	.231	.193	**.235**	.194	.226	.182	.217	.182	.178
eTour +mc	**.385**	.376	.374	.379	.378	.384	.368	.384	.351
+mc+cd	.380	.372	.368	.370	.383	**.388**	.370	.385	.343
+mc+uct	.464	.469	.466	.472	.462	**.473**	.459	.468	.470
+mc+cd+uct	.461	.472	.465	.467	.468	.474	.467	**.476**	.473

Tabelle A.5.: oben: F_1^{OPT}: F_1-Werte für Anreicherung mit jeweils pro Tabellenzelle optimalen Schwellenwerten bei WSD mit/ohne Wörterbuchgewichtung (dw).
unten: Differenz ΔF_1^{OPT} derselben F_1^{OPT}-Werte in Bezug auf die BL.
„+uB“: mit ursprünglichen Bedeutungen, „−uB“: ohne.
„Dupl. verh./erlaub.“: Duplikate verhindern/erlauben.
„BL“: Baseline (dt. *Referenzwert* (ohne Anreicherung)).
Zeilenmaxima sind fett markiert, Zeilen-Zweitbeste unterstrichen.
Die Option +cd ist bei eAnci wirkungslos und daher eingeklammert.

ΔF_1^{OPT} (mit dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	+**.081**	+.067	+.075	+.067	+.066	+.039	+.062	+.040
+mc+uct(+cd)	+**.042**	+.020	+.036	+.023	+.024	+.003	+.024	+.005
eTour +mc	+.025	+.018	+.028	+.017	+.030	+.022	+**.035**	+.021
+mc+cd	+.033	+.025	+.032	+.025	+**.039**	+.028	+.037	+.028
+mc+uct	−.014	+**.004**	−.019	+.003	−.014	−.002	−.007	−.001
+mc+cd+uct	−.007	−.001	−.008	−.001	−.006	+**.005**	−.002	+.004

ΔF_1^{OPT} (ohne dw)	top connectivity strategy				max coverage strategy			
	Dupl. verh.		Dupl. erlaub.		Dupl. verh.		Dupl. erlaub.	
	+uB	−uB	+uB	−uB	+uB	−uB	+uB	−uB
eAnci +mc(+cd)	+**.094**	+.063	+.083	+.063	+.076	+.044	+.062	+.044
+mc+uct(+cd)	+.053	+.015	+**.057**	+.016	+.048	+.004	+.039	+.005
eTour +mc	+**.033**	+.025	+.022	+.028	+.027	+**.033**	+.016	+.032
+mc+cd	+.037	+.029	+.024	+.027	+.039	+**.045**	+.026	+.041
+mc+uct	−.006	−.002	−.005	+.001	−.008	+**.003**	−.011	−.002
+mc+cd+uct	−.012	−.001	−.008	−.006	−.005	+.002	−.005	+**.004**