Karlsruher Institut für Technologie

Verknüpfung von Textelementen zu Softwarearchitektur-Modellen mit Hilfe von Synsets

Bachelor's Thesis von

Theresa Heine

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

Erstgutachter:	Prof. Dr.-Ing. Anne Koziolek
Zweitgutachter:	Prof. Dr. Ralf H. Reussner
Betreuender Mitarbeiter:	M.Sc. Jan Keim
Zweiter betreuender Mitarbeiter:	M.Sc. Yves R. Schneider

08. Juli 2019 – 07. November 2019

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

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

Karlsruhe, den 07.11.2019

. . . . . .

(Theresa Heine)

Zusammenfassung

Eine gute Softwaredokumentation ist Grundlage für den Langzeiterfolg eines Softwaresystems. Änderungen an der Software ziehen unter Umständen Änderungen an der Dokumentation nach sich. Wird die Dokumentation nicht dementsprechend angepasst, kann es zu Inkonsistenzen bei der Benennung kommen. Trotz dieser Inkonsistenzen soll Rückverfolgbarkeit zwischen den Textelementen der Softwarearchitektur-Dokumentation (SAD) und den Modellelementen des Softwarearchitektur-Modells (SAM) gewährleistet werden. Statt einem direkten Vergleich zwischen den Bezeichnern der Textelemente und den Namen der Modellelemente, wird ein semantischer Vergleich auf Basis der Auflösung sprachlicher Mehrdeutigkeiten (WSD, Word Sense Disambiguation) durchgeführt. Mit dem WSD-Algorithmus Babelfy werden die Bedeutungen der Textelemente im Kontext der SAD in Form von Synsets bestimmt. Diese Synsets bilden Synonyme ab, die für das Erstellen von Verknüpfungen zwischen den Text- und Modellelementen verwendet werden. Unter Einbeziehung der Synonyme zum Erstellen der Verknüpfungen ist es möglich, Textelemente Modellelementen zuzuordnen, die semantisch dasselbe Element abbilden aber unterschiedlich benannt sind. Mit diesem Ansatz kann eine Verbesserung beim Erstellen von Verknüpfungen zwischen den Textelementen der SAD und den Modellelementen des SAM erzielt werden gegenüber einem Ansatz ohne Einbeziehung von Synonymen. Die Ausbeute kann beim Erstellen von Verknüpfungen um durchschnittlich 12 Prozentpunkte auf 53,1% erhöht werden, bei einer Präzision von 90,9%.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Nach Clements et al. [6] ist eine gute Softwarearchitektur-Dokumentation (SAD, Software Architecture Documentation) Grundlage für den Langzeiterfolg eines Softwaresystems. Wenn die Dokumentation nicht die Implementierung der Software widerspiegelt, ist die Wartbarkeit und damit die Lebenszeit der Software eingeschränkt. Änderungen an der Software ziehen unter Umständen Änderungen an der Dokumentation nach sich. Da Änderungen unabdingbar sind, muss auch die Dokumentation entsprechend angepasst werden. Trotzdem wird die Erstellung und Aktualisierung dieser gerne auf einen unbestimmten Zeitpunkt in der Zukunft verschoben und vernachlässigt. Dadurch kann es zu Inkonsistenzen zwischen Software und Dokumentation kommen. Ein Grund für dieses Problem ist, dass die Erstellung und das Aktualisieren einer SAD ein teurer und aufwändiger Prozess ist. Die Softwarearchitektur eines Systems ändert sich fortlaufend und teilweise schneller als diese dokumentiert werden kann (S. 18ff) [6]. Die SAD ist damit veraltet und/ oder unvollständig und stimmt nicht mit dem aktuellen Softwarearchitektur-Modell (SAM, Software Architecture Model) überein. Ein Teilproblem dieser Inkonsistenz bezieht sich auf uneinheitliche Verwendung von Namen für die Elemente eines Softwaresystems. Es kann vorkommen, dass die Bezeichner der SAD nicht mit den Namen des SAM übereinstimmen, obwohl sie dasselbe Modellelement abbilden. Diese inkonsistente Benennung führt dazu, dass eine Rückverfolgbarkeit zwischen diesen Text- und Modellelementen nicht mehr gewährleistet werden kann.

Zur Lösung dieses Problems gibt es mehrere Möglichkeiten, die Konsistenz zwischen SAD und SAM sicherzustellen. Beispielsweise kann die Dokumentation automatisch direkt aus dem Quellcode generiert werden, wodurch es nicht zu Inkonsistenzen kommen kann [15]. Existiert jedoch bereits eine SAD, ist die Ersetzung dieser durch eine neue, automatisch generierte SAD nicht zielführend, wenn Inkonsistenzen identifiziert werden sollen. Alternativ können Verknüpfungen (*Trace Links*) zwischen Dokumentation und Quellcode hergestellt werden, die der Rückverfolgbarkeit dienen [2]. Eine konsistente Dokumentation wird dadurch nicht explizit sichergestellt und das Problem inkonsistenter Benennung nicht aufgelöst. Es bleibt die Möglichkeit, die Dokumentation von Hand auf Inkonsistenzen zu überprüfen und diese zu korrigieren. Nach Dagenais und Robillard [7] liegt der Hauptaufwand bei dieser Vorgehensweise nicht bei der Durchführung von Änderungen in der Dokumentation, sondern beim Herausfinden an welchen Stellen Änderungen notwendig sind. Die händische Methode ist zeitaufwändig und ineffizient.

Aus diesem Grund ist die Idee von Keim und Koziolek [12], den Aufwand zum Erstellen und Aktualisieren einer SAD zu reduzieren, indem Teile des Dokumentationsprozesses automatisiert werden. In diesem Zusammenhang sollen Inkonsistenzen zwischen SAD und SAM automatisch aufgefunden werden. Ein Verarbeitungsschritt zum Auffinden dieser Inkonsistenzen ist herauszufinden, welche Teile der SAD mit dem SAM übereinstimmen. Dafür werden Verknüpfungen zwischen den Textelementen der SAD und den Modell-

elementen des SAM erstellt. Ein direkter Vergleich der Bezeichner der Textelemente mit den Namen der Modellelemente ist aufgrund der inkonsistenten Benennung nicht zielführend. Aus diesem Grund wird stattdessen ein semantischer Vergleich zum Erstellen der Verknüpfungen angewandt. Dieser semantische Vergleich wird mit Hilfe der WSD durchgeführt. Mit einem WSD-Algorithmus werden die Bedeutungen der Textelemente im Kontext der SAD in Form von Synsets bestimmt. Die Synonyme der Synsets werden dafür verwendet Verknüpfungen zwischen Bezeichnern von Textelementen der SAD und Namen von Modellelemente des SAM herzustellen, die dasselbe Element abbilden aber unterschiedlich benannt sind.

Das Hauptziel, welches mit dieser Arbeit erreicht werden soll, ist den aktuellen Stand einer SAD mit dem aktuellen Stand eines SAM vergleichen zu können. Durch das Erstellen von Verknüpfungen zwischen Text- und Modellelementen, die dasselbe Elemente abbilden aber unterschiedlich benannt sind, können Inkonsistenzen zwischen SAD und SAM aufgelöst werden. Damit wird die Rückverfolgbarkeit zwischen den Text- und Modellelementen ermöglicht. Dieses Wissen ermöglicht die Überarbeitung der SAD und verhindert, dass Leser der SAD falsche Informationen über das Softwaresystem erhalten. Darüber hinaus wird der Aufwand reduziert, der zum Aktualisieren einer SAD benötigt wird. Dies wird erreicht, indem Stellen der SAD identifiziert werden können, welche nicht konsistent mit dem SAM sind und damit Änderungen erforderlich machen.

2. Grundlagen

Dieses Kapitel gibt einen Überblick über die Grundlagen zur Verarbeitung von natürlicher Sprache. Eine Softwaredokumentation besteht neben Quellcode aus einer Vielzahl von Modellen, Tabellen und Diagrammen. Diese haben gemein, dass sie mit natürlicher Sprache beschrieben und erklärt werden. Was in diesem Zusammenhang unter einer Softwarearchitektur-Dokumentation (SAD, Software Architecture Documentation) zu verstehen ist, wird in Abschnitt 2.1 erläutert. Ein Großteil dieser SAD ist informell verfasst. Wie diese informelle Dokumentation softwaretechnisch verarbeitet werden kann und was dabei zu beachten ist, wird in Abschnitt 2.2 eingeführt. In Abschnitt 2.3 wird auf die Auflösung sprachlicher Mehrdeutigkeiten eingegangen. In diesem Zusammenhang werden in Abschnitt 2.4 Synsets eingeführt. Zur Bestimmung der Ähnlichkeit zwischen Zeichenketten werden in Abschnitt 2.5 zwei Verfahren erläutert, welche zum Erstellung der Verknüpfungen zwischen Text- und Modellelementen relevant sind. Zuletzt werden in Abschnitt 2.6 Ontologien eingeführt, welche das Softwarearchitektur-Modell (SAM, Software Architecture Model) abbilden.

2.1. Softwarearchitektur-Dokumentation

Bevor eine softwarebasierte Verarbeitung der informellen Dokumentation durchgeführt werden kann, ist es sinnvoll zu definieren, was man darunter versteht und warum der informelle Teil der Dokumentation wichtig ist.

In einer SAD werden Architekturentscheidungen und Problemlösungen, die während der Lebenszeit eines Softwaresystems in Bezug auf die Architektur getroffen werden festgehalten. Eine SAD wird zur Kommunikation dieser Entscheidungen und Lösungen an alle am Projekt beteiligten Personen angefertigt. Alle relevanten Informationen müssen für die jeweiligen Personen in der SAD enthalten sein, damit diese das System verwenden, warten und umsetzen können. Die relevanten Informationen der SAD sind für die verschiedenen Personengruppen sehr unterschiedlich. Beispielsweise benötigt ein Entwickler Informationen über Bedingungen, die gelten müssen und Freiheiten, die ihm bei der Implementierung überlassen werden. Ein Tester hingegen benötigt Informationen zur Erstellung der Testfälle, welche die gewünschte Funktionalität des Systems gewährleisten. Unter Umständen sind diese Personen organisatorisch, geografisch oder sogar zeitlich voneinander getrennt (S. 9ff) [6]. Der informelle Teil, also die textuelle Beschreibung der Softwarearchitektur, ist deshalb zur Erklärungen und eindeutigen Interpretation der Architekturmodelle wichtig (S. 40f) [6]. Darüber hinaus enthalten textuelle Beschreibungen Informationen, die in den Modellen, Tabellen und Diagrammen nicht enthalten sind. Beispielsweise ist eine mathematische Formel ein präziser Ausdruck, ohne zusätzliche Erklärung unter Umständen jedoch nur schwer verständlich [2].

2.2. Verarbeitung natürlicher Sprache

Die softwaretechnische Verarbeitung natürlicher Sprache (NLP, Natural Language Processing) beschreibt das Verstehen und die Manipulation von Daten in natürlicher Sprache, mit Hilfe von computergestützten Algorithmen. Die Datenmengen können dabei sowohl in gesprochener, als auch geschriebener Form vorliegen. Die Grundlagen von NLP basieren auf Computer- und Informationswissenschaften, Linguistik, Mathematik, Elektrotechnik, künstlicher Intelligenz, Robotik und Psychologie. Dadurch ist es beispielsweise möglich, Texte maschinell in eine andere Sprache zu übersetzten, Grammatikschreibweisen zu überprüfen, Assistenz- oder Dialogsysteme zu entwerfen, Wortvorhersagen zu machen und Gebrauchsgegenstände mit Sprachsteuerung zu bedienen [5, 11].

In Bezug auf die Verarbeitung von Textdokumenten (wie der SAD) ist das Ziel von NLP die Umwandlung informeller Dokumente, die beispielsweise Mehrdeutigkeiten enthalten, in formal eindeutige Dokumente [11]. Ein grundlegendes Teilgebiet von NLP ist das Verstehen natürlicher Sprache (NLU, Natural Language Understanding). Um den Inhalt eines Textes verstehen zu können, kommt es auf die Darstellung und den Bereich, beziehungsweise das Thema des Textes und auf das Allgemeinwissen des Verfassers an. Aus diesem Grund ist es sinnvoll, erst die Wörter einzeln zu betrachten, dann diese im Kontext des Satzes zu interpretieren und zum Schluss diese in Bezug auf das gesamte Dokument in Beziehung zu setzen [5, 11]. Um ein umfassendes Verständnis vom Inhalt eines Textes zu erlangen, werden die folgenden Analysen bei NLU berücksichtigt [11]:

- Morphologische Analyse (Struktur und Aufbau von Wörtern)
- Lexikalische Analyse (Bedeutung eines einzelnen Wortes)
- Syntaktische Analyse (Grammatik und Struktur von Sätzen)
- Semantische Analyse (Bedeutung von Wörtern und Sätzen)
- Diskursanalyse (Korreferenzen und Beziehungen über mehrere Sätze)
- Pragmatische Analyse (Weltwissen unabhängig des Textes)

In Bezug auf diese Arbeit sind besonders die syntaktische, semantische und Diskursanalyse relevant. Zur syntaktischen Analyse gehören, unter anderen, Lemmatisierung und die Zuordnung von Wörtern zu Wortarten (POS-Tagging, Part-Of-Speech-Tagging). Unter Lemmatisierung versteht man die Umformung eines Wortes in dessen Grundform. So wird beispielsweise *fetching* oder *fetches* in die Grundform *fetch* umgewandelt. Unter POS-Tagging versteht man die grammatikalische Zuordnung von Wörtern zu den Wortarten. Ein Wort kann beispielsweise ein Nomen, Verb oder Adjektiv sein und im Singular oder Plural vorliegen. Zur semantischen Analyse gehören, unter anderen, die Auflösung sprachlicher Mehrdeutigkeiten und das Erstellen von Entitätsverknüpfungen (EL, Entity Linking) [11]. Unter der Auflösung sprachlicher Mehrdeutigkeiten versteht man das Bestimmen von Wortbedeutungen abhängig vom Kontext. Eine ausführliche Einführung zu diesem Thema kommt im folgenden Abschnitt 2.3. Unter EL versteht man das Auffinden von Entitäten, denen ein passender Eintrag in einer Wissensbasis zugeordnet werden kann. Dadurch

kann beispielsweise von einem Fußballspieler Thomas auf den berühmten Thomas Müller geschlossen werden [18, 19]. Zur Diskursanalyse gehört, unter anderem, die Auflösung von Korreferenzen. Unter Korreferenzen versteht man, dass für ein Wort zwei verschiedene Bezeichner verwendet werden, um Wiederholungen zu vermeiden [11]. Beispielsweise sind im Satz „When a user uploads a file, it will be stored in the DataStorage“ (S.7) [23] die Wörter *file* und *it* korreferent.

2.3. Auflösung sprachlicher Mehrdeutigkeiten

Ein Teil der semantischen Analyse und damit eine Herausforderung von NLP, ist die Auflösung sprachlicher Mehrdeutigkeiten (WSD, Word Sense Disambiguation). Ein Wort kann, abhängig vom Kontext in dem es steht, mehrere Bedeutungen haben (S. 671ff) [11]. Das Bestimmen der Bedeutung eines Wortes in einem bestimmten Kontext nennt man *disambiguieren*. So hat beispielsweise das Wort *memory* in den folgenden zwei Sätzen, abhängig des Kontextes in dem es steht, unterschiedliche Bedeutungen:

1. He lost his memory after the accident.
2. He does not have enough memory to run the computer game.

Im ersten Satz ist mit *memory* das Gedächtnis, beziehungsweise der Gedächtnisverlust gemeint. Im zweiten Satz handelt es sich um Datenspeicher. Es gibt also mehrere Bedeutungen des Wortes *memory* [25].

Es gibt verschiedene WSD-Algorithmen, um die Bedeutung eines Wortes abhängig vom Kontext zu bestimmen. Diese Algorithmen werden in drei Kategorien unterschieden: *Supervised*, *Unsupervised* und *Knowledge Based* WSD [19]. Unter *Supervised* WSD versteht man die Auflösung von Mehrdeutigkeiten, mit Hilfe von Machine Learning-Algorithmen und annotierten Trainingsdaten. Die Trainingsdaten bilden die Bedeutungen von Wörtern in verschiedenen Kontexten ab. Das System bestimmt die Bedeutung eines Wortes auf Basis der zuvor gelernten Daten. Erfahrungsgemäß liefern diese Algorithmen die höchste Trefferquote bei der richtigen Interpretation der Bedeutung eines Wortes abhängig vom Kontext. Die Erstellung der Trainingsdaten ist jedoch aufwändig und teuer (S. 673f) [11, 21]. Hingegen werden bei *Unsupervised* WSD nicht annotierte Trainingsdaten verwendet, eine wesentliche Schwierigkeit liegt jedoch in der Auswertung der Ergebnisse [19]. Aus diesem Grund bieten sich alternativ wissensbasierte, beziehungsweise *Knowledge Based* WSD-Algorithmen an. Darunter versteht man die Auflösung sprachlicher Mehrdeutigkeiten mit Hilfe der Eigenschaften lexikalischer Ressourcen, wie Wörterbüchern und Thesaurus-Methoden. Darunter versteht man die Verwendung von Wortnetzen, die aus bedeutungsverwandten Wörtern und deren Beziehungen zueinander aufgebaut sind. Ein bekanntes Beispiel für eine lexikalische Ressource der englischen Sprache ist WordNet. Dieses Wortnetz ist aus sogenannten Synsets aufgebaut (S. 680f) [11, 21].

In den folgenden zwei Abschnitten werden zwei Knowledge Based WSD-Algorithmen vorgestellt und wie diese in ein Softwaresystem integriert werden können. Zum einen wird der Lesk-Algorithmus als klassischer WSD-Algorithmus vorgestellt. Zum anderen wird für einen Algorithmus auf dem aktuellen Stand der Technik der Babelfy-Algorithmus eingeführt.

2.3.1. Lesk-Algortihmus

Mit dem Lesk-Algorithmus wird die Bedeutung eines Wortes anhand der Definitionen der möglichen Bedeutungen bestimmt. Der Kontext, indem die Bedeutung des Wortes bestimmt wird, ist die Phrase oder der Satz selbst. Es wird diejenige Bedeutung für ein Wort ausgewählt, für die Wortüberschneidungen der Definitionen mit den Definitionen der anderen Wörter der Phrase oder des Satzes existieren [21]. Für die Phrase „memory drive“ werden mit dem Lesk-Algorithmus die Definitionen von *memory* mit den Definitionen von *drive* auf Wortüberschneidungen überprüft. Beispielsweise existieren die folgenden zwei Definitionen für mögliche Bedeutungen des Wortes *memory* [18, 19]:

D1: „The cognitive process whereby past experience is remembered“

D2: „An electronic memory **device**“

Die Wörter dieser Definitionen werden auf Wortüberschneidungen mit den Wörtern der möglichen Definitionen der Bedeutungen von *drive* untersucht [18, 19]:

D1: „A **device** that writes data onto or reads data from a storage medium“

D2: „A road leading up to a private house“

Für die Definition D2 von *memory* und die Definition D1 von *drive* existiert eine Überschneidung für das Wort *device*. Damit wird der Phrase *memory drive* die Bedeutung elektronisches Speichergerät durch den Lesk-Algorithmus zugeordnet.

Zur Integration des Lesk-Algorithmus in ein Softwaresystem stehen verschiedene Framworks für unterschiedliche Programmiersprachen zur Verfügung. *DKPro WSD* ist ein Java-Framework, welches verschiedene grundlegende WSD-Ansätze anbietet [16]. Beispielsweise stehen WSD-Algorithmen, wie Most Common Sense (MCS) und verschiedene Lesk-Varianten, unter der GNU General Public License, Version 3, zur Verfügung [16]. Als lexikalische Ressource stehen, unter anderen, WordNet, Wikipedia, Wiktionary und Google zur Auswahl. Der Simplified-Lesk-Algorithmus ist eine Abwandlung des Lesk-Algorithmus, mit der Änderung, dass der Kontext zur Disambiguierung explizit angegeben werden kann [21]. Zur Bestimmung der Bedeutung eines Wortes kann der Kontext unabhängig davon, beispielsweise in Form eines Satzes, angegeben werden. Das Wort selbst muss nicht Teil des Satzes sein. Die Bedeutung des Wortes in diesem Kontext wird, wie beim ursprünglichen Lesk-Algorithmus, anhand der Wortüberschneidungen der Definitionen bestimmt. Mit dem Algorithmus kann abhängig vom getesteten Datensatz ein F_1-Maß von 53,6% erreicht werden [21].

2.3.2. Babelfy-Algorithmus

Babelfy ist nach Moro et al. in [19] ein graphbasierter Ansatz für WSD und EL. Die zugrundeliegende lexikalische Ressource von Babelfy ist BabelNet. Mit Babelfy wird ein durchschnittliches F_1-Maß von 65,5% erreicht [21]. Sowohl für Babelfy, als auch für BabelNet steht eine HTTP-Programmierschnittstelle (API, Application Programming Interface) für Forschungszwecke frei zur Verfügung. Im Folgenden werden die Parameter für die API-Anfragen erläutert.

Bei einer HTTP-Anfrage an Babelfy können verschiedene Parameter gesetzt werden, um die Wortbedeutungen eines Textes zu bestimmen, beziehungsweise um den Text zu disambiguieren. Erforderlich ist das Setzen der drei Parameter *text*, *lang* und *key*. In dem folgenden Beispiel 1 ist die URL für eine Anfrage an Babelfy mit Platzhaltern für die Parameter angegeben [18, 17]:

Beispiel 1: *https://babelfy.io/v1/disambiguate?**text**={text}&**lang**={lang}&**key**={key}*

Dabei entspricht *text* dem Satz der disambiguiert werden soll und *lang* der Sprache des Textes. Bei jeder Anfrage wird der API-Schlüssel *key* mitgegeben, welcher bei der Registrierung bei BabelNet zugeteilt wird. Zusätzlich zu den erforderlichen Parametern können optionale Parameter gesetzt werden. Von diesen optionalen Parametern wird im Folgenden eine Auswahl vorgestellt, eine ausführliche Liste ist auf der Website des Babelfy API Guides zu finden [18, 17]. Zu diesen optionalen Parametern gehören beispielsweise der semantische Typ der Annotation (*annType*) und die semantische Ressource der Annotation (*annRes*). Beim setzen von *annType=CONCEPTS* wird der Text nur mit WSD disambiguiert, also die Bedeutungen der Wörter abhängig vom Kontext bestimmt. Beim setzen von *annType=NAMED_ENTITIES* wird kein WSD, sondern nur EL durchgeführt. Es werden nicht die Bedeutungen der Wörter bestimmt, sondern versucht eine Zuordnung der Wörter zu berühmten Personen, Firmen oder Orten herzustellen. Mit dem Parameter *annRes* kann die lexikalische Ressource bestimmt werden, welche zur Disambiguierung der Wörter im Kontext des Textes verwendet wird. Es stehen die drei Ressourcen WordNet, BabelNet und Wikipedia zur Verfügung. Ein weiterer optionaler Parameter ist *cands*. Dieser ermöglicht entweder alle oder nur die bestmögliche Bedeutungen eines Wortes in einem Kontext zurückzugeben. Des Weiteren kann neben dem Babelfy-Algorithmus die Backup-Strategie MCS aktiviert oder deaktiviert werden. Liefert der Babelfy-Algorithmus für die Bedeutung eines Wortes im gegebenen Kontext kein Ergebnis, wird der MCS zurückgegeben. Implizit ist diese Backup-Strategie aktiviert, für explizites deaktivieren wird der Parameter *MSC=OFF* gesetzt [18, 17].

Die Bedeutungen der Wörter im Kontext werden mit Babelfy durch ein Synset mit einer *SynsetId* eindeutig identifiziert. Zur Auflösung dieser *SynsetId* wird nach der Babelfy-Anfrage eine weitere HTTP-Anfrage an BabelNet gestellt. Um die Synonyme eines Synsets einer gegebenen *SynsedId* zu erhalten, ist das Setzten von zwei Parametern bei einer BabelNet-Anfrage erforderlich. Diese zwei Parameter entsprechen der *SynsetId* und dem API-Schlüssel. In Beispiel 2 ist die URL für eine Anfrage an BabelNet mit Platzhaltern für die Parameter angegeben [18, 17]:

Beispiel 2: *https://babelnet.io/v5/getSynset?**id**={synsetId}&**key**={key}*

Für diese HTTP-Anfrage wird für eine *SynsetId* eine Liste von Synonymen für die durch das Synset dargestellte Bedeutung zurückgegeben. Zusätzlich zu den zwei erforderlichen Parametern kann der optionale Parameter *targetLang* gesetzt werden. Dadurch wird bestimmt, in welcher Sprache die Ergebnisse der Anfrage zurückgegeben werden sollen.

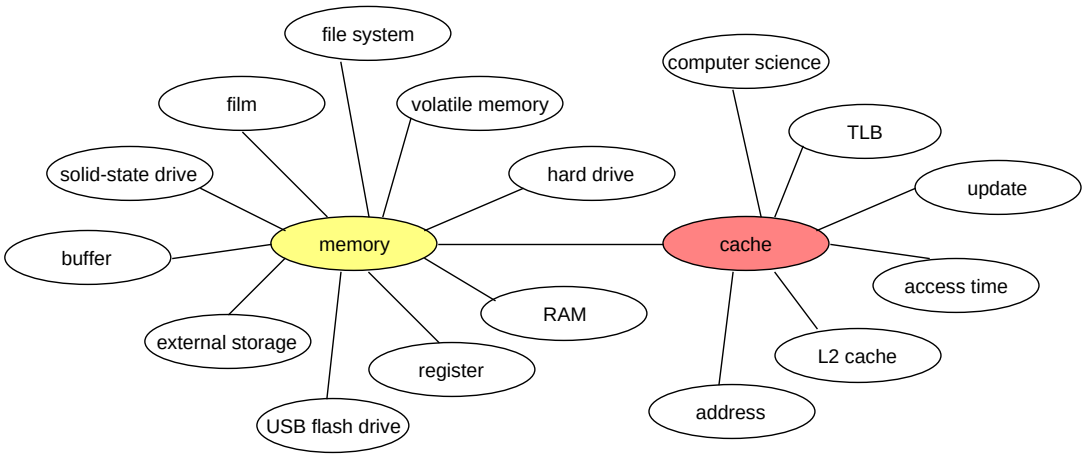


Abbildung 2.1.: Darstellung der Synsets mit denen das Synset des Wortes „memory“, für die Bedeutung elektronisches Speichergerät, in Relation steht [18]

An BabelNet können weitere Anfragen gestellt werden, als die hier vorgestellte Anfrage zum Erhalten von Informationen über ein Synset. Diese anderen Anfragen sind im Zusammenhang mit dieser Arbeit nicht relevant und sind auf der Website des API Guides für BabelNet aufgeführt [18, 17].

2.4. Synsets

Synsets drücken lexikalische und semantische Relationen zwischen Wörtern aus. Die Wörter eines Synsets stehen in direkter Beziehung zueinander und haben dieselbe Wortbedeutung. Das bedeutet, dass die Wörter eines Synsets einem Set aus Synonymen entsprechen [24]. Diese Sets stehen wiederum in Relation mit anderen Synsets und bilden damit ein Netzwerk aus Relationen zwischen Wörtern und deren Bedeutungen ab. Besteht das Synset eines Wortes nur aus dem Wort selbst, existieren keine Synonyme für dieses Wort. Es gibt damit keine Wörter, welche dieselbe Wortbedeutung haben. Für das Wort *architecture*, mit einer Bedeutung in Bezug auf die Architektur von Gebäuden existieren keine Synonyme, da das zugehörige Synset nur dieses eine Wort enthält [18, 19]. Hingegen existiert für das Wort *memory* ein Synset mit den Wörtern *memory*, *storage*, *computer memory*, *store* und *computer storage*. Die Wörter dieses Synsets bilden die Wortbedeutung *electronic memory device* (elektronisches Speichergerät) ab. Darüber hinaus wird das Synset den Kategorien Rechnerarchitektur, Computerspeicher und Datenmanagement zugeordnet [18, 17]. In Abbildung 2.1 ist dargestellt, mit welchen anderen Synsets dieses Synset in Relation steht. Jedes dieses Synsets steht wiederum in Relation mit anderen Synsets. Diese anderen Synsets stellen eine Generalisierung oder Spezialisierung der gegebenen Wortbedeutung für *memory* dar. Eine Generalisierung entspricht in diesem Fall einem übergeordneten Begriff, der Fachbegriff dafür ist Hyperonym. In diesem Zusammenhang ist *memory* ein Hyperonym von *cache*. Der Gegensatz zum Hyperonym ist Hyponym, das entspricht

einen untergeordneten Begriff und bezeichnet damit eine Spezialisierung. *L2 cache* ist beispielsweise ein Hyponym von cache [11, 19].

Bedeutungsverwandte Synsets sind durch Kanten verbunden und bilden ein Netzwerk aus Wörtern ab, die in Relation zueinander stehen. Das Wort *memory* mit der Bedeutung elektronisches Speichergerät steht beispielsweise nicht in direkter Relation mit Wörtern wie *remember*, *recall* oder *working memory*. Diese Wörter bilden ein Synset anderer Bedeutung ab, das mentale Erinnern an vergangene Erfahrungen [18, 19]. Damit gibt es mindestens zwei Bedeutungen für das Wort *memory*, welche unterschiedliche Bedeutung haben und nicht in direkter Relation zueinander stehen. Diese Wortbedeutungen stellen das Gegenteil von Synonymen dar und sind damit Homonyme. Hat ein Wort zwei Bedeutungen, die sich semantisch stark unterscheiden sind diese Bedeutungen homonym zueinander. Synonyme stellen das Gegenteil dazu dar, zwei unterschiedliche Wörter haben dieselbe beziehungsweise eine sehr ähnliche Bedeutung [11].

2.5. Ähnlichkeit von Zeichenketten

Im Folgenden werden zwei Verfahren zur Bestimmung der Ähnlichkeit von zwei Zeichenketten vorgestellt. Zum einen ist das die Levenshtein-Distanz, welche in Unterabschnitt 2.5.1 eingeführt wird. Zum anderen ist das die Jaro-Winkler-Distanz, welche in Unterabschnitt 2.5.2 erläutert wird. Diese Verfahren sind relevant in Bezug auf das Erstellen von Verknüpfungen zwischen Text- und Modellelementen.

2.5.1. Levenshtein-Distanz

Unter der Levenshtein-Distanz versteht man die minimale Anzahl an Änderungen die nötig sind, um eine Zeichenkette in eine andere zu überführen. Diese Änderungen umfassen das Löschen, Hinzufügen und Ersetzen einzelner Buchstaben der Zeichenkette. Diese Art zur Bestimmung der Ähnlichkeit zweier Zeichenketten ist auch unter dem Namen *Edit Distance* bekannt [28]. Sind zwei zu vergleichenden Zeichenketten identisch, ist die Levenshtein-Distanz Null. Für jede Änderung, die zum Überführen der einen Zeichenkette in die andere nötig ist, wird der Wert um Eins erhöht. Die Überführung des Wortes *media* in das Wort *medium* erfolgt beispielsweise folgendermaßen:

1. medi**a** -> medi**u** (Ersetzen von „a“ durch „u“)
2. mediu -> mediu**m** (Hinzufügen von „m“)

Um das Wort *media* in das Wort *medium* zu überführen, wird der Buchstabe „a“ durch „u“ ersetzt. Daraufhin wird der Buchstabe „m“ dem Wort hinzugefügt. Damit sind zwei Änderungen nötig, um das Wort *media* in das Wort *medium* umzuwandeln. Daraus folgt: Levenstein-Distanz(media, medium) = 2.

2.5.2. Jaro-Winkler-Distanz

Die Jaro-Winkler-Distanz wird über die Ähnlichkeit zweier Zeichenketten bestimmt, der sogenannten Jaro-Winkler-Ähnlichkeit. Die Jaro-Winkler-Distanz ist eine Abwandlung der

Jaro-Distanz, bei der Unterschiede am Anfang der Zeichenketten einen höheren Einfluss auf die Distanz haben als Unterschiede am Ende der Zeichenketten [14]. Die Jaro-Winkler-Distanz ist die Inverse der Jaro-Winkler-Ähnlichkeit und berechnet sich aus *1 - Jaro-Winkler-Ähnlichkeit*. Von der Jaro-Winkler-Distanz werden Werte zwischen Null und Eins angenommen, wobei ein Wert von Null bedeutet, dass die Wörter identisch sind und ein Wert von Eins, dass die Distanz maximal ist.

Zur Bestimmung der Jaro-Ähnlichkeit werden die Anzahl und Reihenfolge gemeinsamer Zeichen der zwei Zeichenketten bestimmt. Dafür werden die Wortlängen der Zeichenketten *s1* und *s2* bestimmt, sowie die Anzahl der übereinstimmenden Zeichen *m*. Zusätzlich wird die Variable *t* bestimmt, die der Anzahl an Vertauschungen entspricht, damit die Reihenfolge der übereinstimmenden Zeichen gleich wird. Dadurch ergibt sich die Jaro-Ähnlichkeit wie folgt [14]:

$$\text{Jaro-Ähnlichkeit: } sim_j = \frac{1}{3} * (\frac{m}{s1} + \frac{m}{s2} + \frac{m-t}{m})$$

Damit Unterschiede der Zeichenketten am Anfang einen höheren Einfluss auf die Ähnlichkeit haben als die am Ende, wird eine Präfix-Skalierung vorgenommen. Dadurch erhält man die Jaro-Winkler-Ähnlichkeit wie folgt [14]:

$$\text{Jaro-Winkler-Ähnlichkeit: } sim_{jw} = sim_j + (l * p * (1 - sim_j))$$

In der Formel steht *l* für die Länge des gemeinsamen Präfixes. Dieser Wert ist auf maximal vier Zeichen begrenzt. Und der Parameter *p* steht für einen konstanten Skalierungsfaktor mit einem Standardwert von *0,1* [14].

Im Folgenden wird die Jaro-Winkler-Distanz der Wörter *media* und *medium* bestimmt. Die Wortlänge von *media* wird mit *s1 = 5* und für *medium* mit *s2 = 6* bestimmt. Für die Anzahl übereinstimmender Zeichen gilt *m = 4*. Die Reihenfolge der übereinstimmenden Zeichen ist für beide Wörter gleich, es müssen keine Zeichen vertauscht werden und damit gilt *t = 0*. Die Jaro-Ähnlichkeit berechnet sich wie folgt:

$$sim_j = \frac{1}{3} * (\frac{4}{5} + \frac{4}{6} + \frac{4-0}{4}) = \frac{1}{3} * 2,47 \approx 0,82$$

Für die Jaro-Winkler-Ähnlichkeit ist die Länge des gemeinsamen Präfixes *l = 4* und der konstante Skalierungsfaktor *p = 0,1*. Damit ist die Jaro-Winkler-Ähnlichkeit folgende:

$$sim_{jw} = 0,82 + (4 * 0.1 * (1 - 0,82)) = 0,82 + 0,08 = 0,9$$

Die Inverse der Jaro-Winkler-Ähnlichkeit ergibt die Jaro-Winkler-Distanz von *media* und *medium*::

$$dist_{jw} = 1 - 0,9 = 0,1$$

Daraus folgt: Jaro-Winkler-Distanz(media, medium) = 0,1.

2.6. Ontologien

Unter Ontologien ist allgemein eine konzeptuelle Darstellung von Informationen zu verstehen. Ein System, beziehungsweise Wissen und Informationen über dieses System, werden mit Ontologien in formaler abstrakter Form dargestellt. Objekte, Konzepte, Entitäten und deren Relationen zueinander werden verwendet, um das System in formaler Form abzubilden [9].

In dieser Arbeit werden Ontologien für die abstrakte Darstellung von der Struktur und den Informationen eines Softwaresystems in digitaler formaler Form verwendet. Eigenschaften von und Relationen zwischen Softwaremodellen werden dabei in Form eines Netzwerks dargestellt (S.23ff) [13]. Explizit werden NLP und NLU unter Verwendung der in Ontologieform vorliegenden Wissensbasis durchgeführt. Diese Wissensbasis entspricht dem SAM in Ontologieform und wird mit Hilfe der Web-Ontologie-Sprache (OWL, Web Ontology Language) verarbeitet. Dadurch können Informationen über die Softwarearchitektur des Systems gewonnen werden. Die Modellelemente des SAM werden aus der Ontologie extrahiert und mit den in der SAD enthaltenen Textelemente verglichen. Jedes Modellelement wird über eine *URI* in der Ontologie eindeutig identifiziert. Von der *URI* kann auf den Namen des Modellelements geschlossen werden.

3. Verwandte Arbeiten

In diesem Kapitel wird in Abschnitt 3.1 auf verwandte Arbeiten in Bezug auf die Konsistenzhaltung von Softwaresystemen und Dokumentation eingegangen. Dabei werden die Herangehensweise, die verwendeten Methoden und die gewonnenen Erkenntnisse kurz erläutert. Zum anderen werden verschiedene WSD-Algorithmen in Abschnitt 3.2 vorgestellt, für die Bestimmung von Wortbedeutungen abhängig vom Kontext.

3.1. Konsistenzhaltung von Softwarearchitektur und Dokumentation

Im Zusammenhang mit der Konsistenzhaltung von Softwaresystemen ist *Traceability* (Rückverfolgbarkeit) von Daten, die miteinander in Beziehung stehen, von Bedeutung. *Traceability-Links* sind explizite Verbindungen von einem Quellartefakt zu einem Zielartefakt, die eine Rückverfolgbarkeit gewährleisten. Beispielsweise kann mit Traceability überprüft werden, ob die implementierten Testfälle die geplanten Anforderungen eines Softwaresystems abdecken. Dafür werden Traceability-Links zwischen den Testfällen und den Anforderungen erstellt, welche den Anforderungen Testfälle zuordnen. Traceability stellt sicher, dass das zu entwickelnde System das Design reflektiert [8]. Damit ist Traceability in Bezug auf die Konsistenzhaltung zwischen SAD und SAM relevant.

Antoniol et al. stellen in [2] einen Ansatz zur Wiederherstellung und Erhaltung von Traceability-Links zwischen Softwaredokumentation und Quellcode vor. Ihr Vorgehen basiert auf Information-Retrieval-Methoden. Dabei werden Teile der Dokumentation mit den Namen der Quellcodekomponenten verglichen und zugeordnet. Die Komponenten werden abhängig von der Übereinstimmung mit dem Textdokument, und damit der Wahrscheinlichkeit der korrekten Zuordnung, in Reihenfolge aufgelistet. Ein Klassifikator bestimmt die Ähnlichkeit zwischen dem Textdokument und den Quellcodekomponenten und stellt die Traceability-Links her. Zur Evaluation werden die Metriken Ausbeute (*Recall*) und Präzision (*Precision*) betrachtet. Je besser die Ausbeute desto, schlechter fällt normalerweise die Präzision aus [2]. Die Vorgehensweise von Antoniol et al. bezieht sich in erster Linie auf die Wiederherstellung von Traceability-Links und weniger auf die Konsistenzhaltung und das Auffinden von Inkonsistenzen. Darüber hinaus ist für das Auffinden von Inkonsistenzen sowohl eine hohe Präzision, als auch eine hohe Ausbeute erstrebenswert, beim Erstellen der Traceability-Links.

Guo et al. stellen in [10] einen Ansatz zur verbesserten Traceability von Softwareanforderungen vor. Dabei wenden sie *Deep-Learning*-Techniken an und beziehen bei der Erstellung der Traceability-Links die Semantik der Anforderungen mit ein. Ihre Idee basiert auf NLP-Technologien und Neuronalen Netzen. Informationen über das Softwaresystem werden dabei durch Wortvektoren repräsentiert, welche in Neuronalen Netzen verwendet

werden um die Semantik der Anforderungen zu erlernen. Der Ansatz liefert verbesserte Werte in Bezug auf Ausbeute und Präzision bei der Erstellung von Traceability-Links [10]. Durch Verwendung von Deep-Learning-Techniken werden zur Umsetzung dieses Ansatzes viele annotierte Daten und lange Trainingszeiten benötigt. Zum Erreichen der verbesserten Ergebnisse ist ein teurer und aufwändiger Prozess nötig.

Ein weiterer Ansatz von McBurney und McMillan [15], mit dem eine bessere Konsistenz von Softwaredokumentation und Quellcode erreicht werden kann, ist die automatische Generierung der Dokumentation aus Statements und Kommentaren des Quellcodes. Aus Schlüsselwörtern des Quellcodes werden Sätze in natürlicher Sprache generiert. Die Generierung von Kontextinformationen steht dabei im Vordergrund. Diese werden aus der Art und Weise, wie und unter welchen Umständen die Methoden aufgerufen werden gewonnen [15]. Bei McBurney und McMillan geht es in erster Linie um die Generierung und Verbesserung von Softwaredokumentation. Bei der Generierung von Dokumentation aus Quellcode, kann die Konsistenz zwischen diesen gewährleistet werden. Auf Basis einer existierenden SAD ist der Ansatz zur Sicherstellung der Konsistenz nicht anwendbar, ohne die existierende Dokumentation zu verwerfen.

Die genannten Ansätze liefern, in Bezug auf die Konsistenzhaltung zwischen SAD und SAM, noch keine zufriedenstellende Lösung. Besonders wird bei den Ansätzen nicht betrachtet, wie eine Zuordnung zwischen einem Textelement der SAD und einem Modellelement des SAM ermöglicht wird, wenn diese dasselbe Element abbilden, aber semantisch unterschiedlich benannt sind. Aus diesem Grund ist es interessant, allgemeine Knowledge Based und Supervised WSD Ansätze zu betrachten, welche nicht auf den spezifischen Fall einer SAD begrenzt sind. Im Gegensatz zu Supervised WSD-Ansätzen, die auf Machine Learning-Algorithmen basieren, werden für Knowledge Based WSD keine qualitativ hochwertigen Trainingsdaten benötigt (S. 673f) [11], [21].

3.2. Auflösung sprachlicher Mehrdeutigkeiten

Es gibt verschiedene Ansätze für WSD-Algorithmen. In diesem Abschnitt werden drei *Knowledge Based* WSD-Algorithmen und zwei *Supervised* WSD-Algorithmen vorgestellt. Wie diese in Bezug auf das F_1-Maß im Vergleich abschneiden ist in Tabelle 3.1 dargestellt. Die drei *Knowledge Based* WSD-Algorithmen, die nach Ruder in [22] und Raganato et al. in [21] die besten Ergebnisse liefern, sind Babelfy, UKB und WSD-TM. Diese werden im Folgenden erläutert.

Babelfy ist nach Moro et al. in [19] ein graphbasierter Ansatz für WSD und EL. Die zugrundeliegende lexikalische Ressource von Babelfy ist BabelNet. Bei der Verarbeitung eines Textes werden alle möglichen Bedeutungen eines Textelements aufgelistet und verknüpft. Die Verknüpfungen ergeben einen Graphen, der die semantische Interpretation des gesamten Textes darstellt. Die Bedeutung der einzelnen Elemente im gegebenen Kontext wird durch einen bestmöglichen Untergraphen bestimmt. Mit Babelfy wird bei englischer Sprache und unter Verwendung von BabelNet ein durchschnittliches F_1-Maß von 65,5% erreicht [21].

Ansatz	WSD-Algorithmus	F_1-Maß
Knowledge Based	Babelfy	65,5%
	UKB	53,2%
	Topic Model	66,9%
Supervised	IMS	68,8%
	MFS	62,9%

Tabelle 3.1.: Vergleich verschiedener WSD-Algorithmen in Bezug auf das F_1-Maß

UKB ist ein Open Source Projekt das nach Agirre et al. in [1] Knowledge Based WSD ausführen kann. WSD wird mit graphbasierten Algorithmen und in der englischen Sprache, unter Verwendung von WordNet durchgeführt. Für bestmögliche Ergebnisse bei Betrachtung eines Satzes werden sowohl der vorherige, als auch der nachfolgende Satz in Bezug auf den Kontext berücksichtigt. Die besten Ergebnisse bei der Verwendung von graphbasierten Algorithmen erhält man bei Anwendung von UKBppr_w2w. Mit diesem Vorgehen kann ein F_1-Maß von durchschnittlich 53,2% erreicht werden [21].

Ein weiterer Ansatz, der eine hohe Trefferquote bei WSD erreicht, ist das *Topic Model* WSD-TM von Chaplot und Salakhutdinov in [4]. Anstatt sich nur auf den Kontext einzelner oder naheliegender Wörter und Sätze zu beziehen, wird der Kontext des gesamten Dokuments bei WSD miteinbezogen. Dieses Topic Model basiert auf den WordNet Synsets der Wörter und der Anzahl an Überschneidungen zwischen diesen Synsets. Mit diesem Vorgehen kann bei englischer Sprache ein F_1-Maß von durchschnittlich 66,9% erreicht werden [4].

Im Gegensatz zu *Knowledge Based* WSD-Algorithmen, die auf Basis lexikalischer Ressourcen die Bedeutungen von Wörtern in einem Kontext bestimmen, gibt es auch *Supervised* WSD-Algorithmen. Diese überwachten Ansätze basieren auf Machine Learning-Algorithmen und benötigen annotierte Trainingsdaten. Im Folgenden werden die zwei *Supervised*-Ansätze MFS und IMS vorgestellt.

Beim MFS-Ansatz (Most Frequent Sense) wird für jedes Zielartefakt diejenige Bedeutung gewählt, die am Häufigsten in den Trainingsdaten auftaucht verwendet. Damit wird diejenige Bedeutung gewählt, die in den meisten Anwendungsfällen korrekt ist. Es wird ein durchschnittliches F_1-Maß von 62,9% erreicht [21].

IMS benutzt eine SVM-Klassifizierer (Support Vector Machine) um die Güte der Zuordnungen zu bestimmen. Eine SVM teilt die Bedeutungen in Klassen ein, sodass die Grenzen zwischen den Klassen möglichst breit sind. Bedeutungen in unterschiedlichen Klassen sind also möglichst fern. IMS ist erweiterbar und kann zum Beispiel auch mit Embeddings kombiniert werden. In der Standardimplementierung werden umliegende Wörter, POS-Tagging und Nachbarschaftshäufigkeit als Features zur Klassifizierung verwendet. Es wird ein durchschnittliches F_1-Maß von 68,8% erreicht [21].

Die genannten WSD-Algorithmen beziehen sich auf die Auflösung von Mehrdeutigkeiten in beliebigen, allgemeinen Texten. Bei Anwendung dieser Ansätze auf eine SAD werden nicht alle zur Verfügung stehenden Informationen genutzt. Die Art des Textes einer SAD ist bekannt. Die grundlegenden Ideen der WSD-Ansätze können angepasst für eine SAD übernommen werden.

4. Konzept

Das Auffinden von Inkonsistenzen zwischen SAD und SAM ist ein komplexes Problem, welches in mehrere Unterprobleme aufgespalten werden kann. In diesem Kapitel wird zum einen auf die bestehende Arbeit zum Auffinden dieser Inkonsistenzen eingegangen und zum anderen beschrieben, wie diese Arbeit zum Projekt beträgt und wo diese in den Gesamtkontext eingeordnet werden kann. In Abschnitt 4.1 wird die grundlegende Idee vorgestellt, um Konsistenz zwischen SAD und SAM sicherzustellen. Des Weiteren wird vorgestellt an welcher Stelle diese Arbeit dazu beiträgt. Es stehen verschiedene Projekte im Zusammenhang mit dieser Arbeit zur Verfügung. Diese werden in Abschnitt 4.2 vorgestellt. Der Ansatz und die von ihm angegangen Probleme werden in Abschnitt 4.3 beschrieben. In diesem Abschnitt werden auch die Ziele des Agenten formuliert. Zuletzt wird in Abschnitt 4.4 eine Beschreibung und Darstellung des UML-Diagramms des Ansatzes angegeben.

4.1. Einordnung der Arbeit in den Gesamtkontext

Diese Arbeit ist Teil eines größeren Gesamtkomplexes. Keim und Koziolek stellen in [12] eine Idee für die Sicherstellung von Konsistenz zwischen SAD und SAM vor. Nach ihnen ist der Mehrwert einer SAD nicht direkt für die Verfasser erkennbar und wird deshalb in vielen Fällen vernachlässigt. Eine schlechte SAD wirkt sich negativ auf die Evolution eines Softwaresystems aus. Aus diesem Grund haben sie eine Idee entwickelt, die das agenten-basierte Framework PARSE verwendet und in Abbildung 4.1 dargestellt ist. Mit diesem System soll zum einen das Problem fehlender und unzureichender Dokumentation eines Softwaresystems angegangen werden. Zum anderen soll der Aufwand reduziert werden, der zum Erstellen und Aktualisieren einer SAD nötig ist, indem Teile des Dokumentationsprozesses automatisiert werden. Das System besteht aus den folgenden Komponenten:

(I) Eine Wissensbasis (*Knowledge Base*) in Ontologieform. Diese enthält allgemeine Informationen zu Softwarearchitektur, Fachwissen und Informationen zur tatsächlichen Softwarearchitektur des Systems. Dazu gehören beispielsweise das Softwarearchitektur-Modell und der Quellcode des Systems.

(II a) Die Vorverarbeitung (*Pre-processing*) der SAD mit Hilfe eines auf ProNat basierenden Frameworks [26]. In diesem Schritt wird der Text mit Hilfe von NLP auf NLU vorbereitet. Darunter versteht man das Parsen und Aufspalten des Eingabedokuments in kleine Untereinheiten, die leichter verarbeitet werden können.

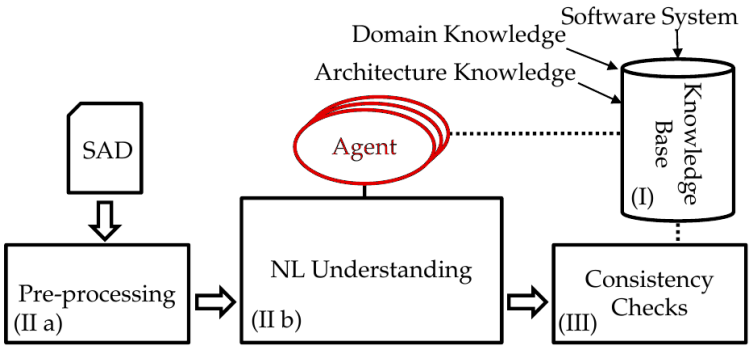


Abbildung 4.1.: Übersicht des Lösungsansatzes zur Konsistenzprüfung zwischen SAD und SAM [12]

(II b) NLU (*NL Understanding*) entspricht der Hauptverarbeitung. NLU ist ein komplexer Prozess, weshalb die Aufgabe des Verstehens der Eingabe in Unteraufgaben aufgeteilt wird, die jeweils von unabhängigen Agenten gelöst werden (diese sind in der Abbildung rot gekennzeichnet).

(III) Konsistenzprüfung (*Consistency Checks*) zwischen der Softwarearchitektur und den gewonnen Informationen aus der SAD.

Ziel dieser Arbeit ist es, einen Agenten zu implementieren, der zum NLU der Hauptverarbeitung beiträgt. Der Agent ist für das Auffinden bestehender Verknüpfungen zwischen SAD und SAM zuständig.

4.2. Bestehende Projekte

ProNat von Weigelt und Tichy [26] ist ein Agenten-basiertes Design, welches dafür ausgelegt ist, Haushaltsrobotern neue Befehle beizubringen. Dabei geht es um die Extraktion von Prozessabläufen aus gesprochenen Äußerungen, welche verwendet werden, um Skripte für diese Abläufe zu programmieren [26]. In diesem Zusammenhang wurde das Projekt *PARSE* umgesetzt. *PARSE* ist auf die Interpretation von sprachlichen Äußerungen ausgelegt. Nach Umwandlung dieser in eine textuelle Repräsentation, sind die Herausforderungen von *PARSE* und denen zur Konsistenzprüfung von SAD uns SAM sehr ähnlich.

Der Prototyp *ArchDocLink* basiert auf *ProNat* und stellt eine erste Umsetzung eines Agenten der Idee dar, welche in Abschnitt 4.1 erläutert ist. Die grundlegende Struktur dieses Agenten dient zur Orientierung für den Ansatz des Agenten zum Erstellen von Verknüpfungen zwischen SAD und SAM. Der Prototyp verwendet die drei Projekte *NLWrapper*, *OntologyAccess* und *GraphImprove*, welche auch im Kontext dieser Arbeit benötigt werden. *GraphImprove* wird zur Darstellung und Annotation des Eingabedokuments in Form eines Graphen benötigt. Damit können Knoten bearbeitet, neue definiert und Relationen erstellt werden. *GraphImprove* verwendet den *NLWrapper*. Dieser ermöglicht eine einfache und einheitliche Nutzung des NLP-Frameworks. Der NLWrapper ist zur Verarbeitung der Sätzen, Phrasen und Wörtern des Dokuments. Mit diesem kann auf die Attribute dieser Einheiten zugegriffen werden. *OntologyAccess* bietet einen einfachen und einheitlichen

Zugriff auf Ontologien. Mit dem *OntologyAccess* können Informationen aus der OWL-Datei ausgelesen, verändert und hinzugefügt werden. Für den Agenten werden nur zwei Funktionalitäten dieses Projekts benötigt, um Verknüpfungen zwischen Text- und Modellelementen zu erstellen. Zum einen wird der *OntologyAccess* dazu verwendet, die Elemente eines Klassennamens auszulesen. Unter diesem Klassennamen sind die Modellelemente, welche für die Verknüpfung betrachtet werden, in der Ontologie zu finden. Zum anderen wird die Funktionalität verwendet, den Namen eines Modellelements auf Basis dessen URI in der OWL-Datei zu bestimmen.

4.3. Konzeptioneller Ansatz

Der erste Schritt, zum Auffinden von Inkonsistenzen zwischen SAD und SAM ist, herauszufinden welche Teile übereinstimmen. Dafür werden Verknüpfungen zwischen den Textelementen der SAD und den Modellelementen des SAM erstellt.

Das Problem beim Erstellen dieser Verknüpfungen liegt darin, dass es Bezeichner der Textelemente der SAD gibt, die nicht mit den Namen der Modellelemente des SAM übereinstimmen, obwohl sie semantisch dasselbe Element abbilden. Diese inkonsistente Benennung führt dazu, dass eine Rückverfolgbarkeit zwischen diesen Text- und Modellelementen nicht mehr gewährleistet werden kann. Diese Unterschiede können in der Ausdrucksweise der Namenskonvention liegen [27], beispielsweise bei inkonsistenter Verwendung von CamelCase-, snake_case- oder kebab-case-Schreibweise. Darüber hinaus werden die Namen der Modellelemente in der betrachteten SAD teilweise auseinander geschrieben und mit Leerzeichen getrennt. Weitere Unterschiede entstehen aufgrund von Namensänderungen während der Lebenszeit des Softwaresystems. Zwei unterschiedliche Namen können bei einer veralteten Dokumentation semantisch dasselbe Element abbilden. Aus diesen Gründen ist eine Zuordnung von Text- und Modellelementen über einen direkten Vergleich der Bezeichnern der Textelemente und den Namen der Modellelemente nicht ausreichend. Um diese Probleme zu lösen, ist die Idee, statt einem direkten Vergleich einen semantischen Vergleich durchzuführen. Dafür wird ein WSD-Algorithmus eingesetzt, um die Bedeutung der Bezeichner im Kontext der SAD aufzulösen. Durch Auflösung dieser Bedeutungen werden Synsets generiert, die zum Erstellen von Verknüpfungen zwischen Text- und Modellelementen beitragen. Dadurch wird eine bessere Verknüpfung von SAD und SAM erreicht.

Zur Umsetzung dieser Idee wird ein Agent implementiert, der zum NLU der Hauptverarbeitung beiträgt (in Abbildung 4.1 rot gekennzeichnet). Der Agent ist zuständig für das Erstellen von Verknüpfungen zwischen den Textelementen der SAD und den Modellelementen des SAM. Die Hauptaufgabe besteht darin, Verknüpfungen zwischen Text- und Modellelementen zu erstellen, die unterschiedlich benannt sind, aber semantisch dasselbe Element abbilden. Dafür wird ein WSD-Algorithmus eingesetzt, der die Bedeutungen der Textelemente im Kontext der SAD in Form von Synsets bestimmt. Die Bezeichner und generierten Synonyme werden über einen direkten Vergleich den Namen der Modellelemente zugeordnet. Für das Vergleichsverfahren wird ein Schwellenwert definiert, der eine Toleranz zulässt. Das bedeutet, dass Verknüpfungen aufgelöst werden können, wenn Umbenennungen auf Basis von Synonymen stattgefunden haben.

„**Media access** is responsible for fetching information about audio files from the **Database** and the audio files themselves from **FileStorage**."

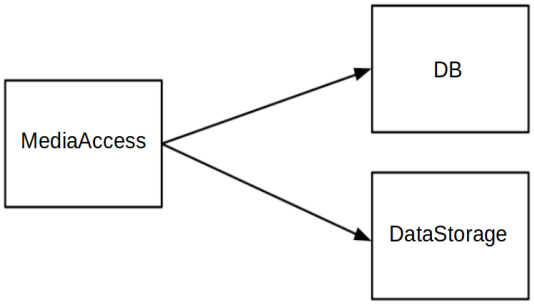


Abbildung 4.2.: Erstellen von Verknüpfungen zwischen SAD (links) und SAM (rechts)

In Abbildung 4.2 ist dargestellt, welche Art von Inkonsistenzen mit dem beschriebenen Synonym-basierten Ansatz aufzulösen sind. Dafür ist ein Beispielsatz angegeben, der einen Ausschnitt einer SAD darstellt und ein beispielhaftes Modell, welches das zugehörige SAM abbildet. Zwischen dieser SAD und diesem SAM werden Verknüpfungen erstellt. Statt dem Modellnamen *MediaAccess* im SAM, wird in der SAD der Bezeichner *Media access* durch ein Leerzeichen getrennt. Das Erstellen einer Verknüpfung erfolgt, indem die Phrase und nicht nur ein einzelnes Wort betrachtet wird. Eine Zuordnung von *Database* und *DB* ist mit Hilfe eines WSD-Algorithmus möglich, wenn die verwendete lexikalische Ressource Abkürzungen unterstützt (wie beispielsweise BabelNet). Das Erstellen einer Verknüpfung zwischen dem Bezeichner *FileStorage* und dem Namen *DataStorage* entspricht dem Hauptanwendungsfall des Synonym-basierten Vergleichsalgorithmus. Diese Elemente bilden semantisch dasselbe Element ab, obwohl der Bezeichner des Textelements nicht mit dem Namen des Modellelements übereinstimmt. Mit einem WSD-Algorithmus kann über das Synset des Wortes *File* auf das Wort *Data* geschlossen und eine Verknüpfung erstellt werden.

Die Ziele, welche mit der Umsetzung des Ansatzes erreicht werden sollen, sind Folgende:

Z1: Güte der Verknüpfung zwischen den Bezeichnern der Textelemente der SAD und den Namen der Modellelemente des SAM bestimmen, mit Hilfe des Synonym-basierten Vergleichsansatzes.

Z2: Güte der Verknüpfung zwischen den Bezeichnern der Textelemente der SAD und den Namen der Modellelemente des SAM bestimmen, ohne Anwendung des Synonym-basierten Vergleichsansatzes.

Z3: Güte der Methoden zur Ermittlung der Ähnlichkeit zwischen den Bezeichnern der Textelemente und den Namen der Modellelemente bestimmen.

Z4: Performance und Skalierbarkeit des Synonym-basierten Vergleichsansatzes abhängig von den einstellbaren Parameter bestimmen.

4.4. Softwarearchitektur-Modell des Ansatzes

In Abbildung 4.3 ist das UML-Diagramm des Synonym-basierten Vergleichsansatzes zur Erstellung der Verknüpfungen zwischen Text- und Modellelementen angegeben. Dieser Ansatz trägt in Form eines Agenten (*WsdLinkAgent*) zum NLU der Hauptverarbeitung der in Abbildung 4.1 dargestellten Idee bei. Bei Initialisierung des Agenten werden alle benötigen Informationen über Setter-Methoden gesetzt oder in einer Konfigurationsdatei definiert. Zur Erstellung der Verknüpfungen zwischen Text- und Modellelementen verfügt der Agent über einen *WsdLink*. Dieser entspricht der Hauptverarbeitungseinheit des Agenten. Der *WsdLink* verfügt über *OntologyProcessing*, *NaturalLanguageProcessing*, *BabelfyProcessing* und eine *StringSimilarityMethod*. *OntologyProcessingImpl* ist eine Instanziierung des Interfaces *OntologyProcessing*, welches den *OntologyAccess* nutzt. Damit werden die Modellelemente, welche für die Erstellung der Verknüpfungen mit den Textelementen betrachtet werden, aus der Ontologie extrahiert. *NaturalLanguageProcessing* steht zur Verarbeitung des Textdokuments der SAD zur Verfügung. Aus diesem Textdokument wird ein Graph aufgebaut, bei dem die Wörter eines Satzes Knoten des Graphen darstellen und durch Kanten verbunden sind. Damit werden die Sätze, Phrasen und Wörter des Graphen verarbeitet und neue Informationen, sowie Knoten und Kanten dem Graphen hinzugefügt. *BabelfyProcessing* ist eine Umsetzung des WSD-Algorithmus Babelfy zur Generierung der Synsets und der damit einhergehenden Synonyme, die bei der Erstellung der Verknüpfungen miteinbezogen werden. Die generierten Synsets und Synonyme für die Textelemente werden als *BabelfyResult* verarbeitet. *BabelfyProcessing* kann eine *BabelNetException* auslösen, wenn an einem Tag zu viele API-Anfragen an BabelNet gestellt werden. Die Klasse *ProcessingHelper* wird sowohl vom *WsdLink*, als auch von *BabelfyProcessing* zur Verarbeitung und Kombination der Bezeichner der Textelemente und deren Synonyme verwendet. Für die *StringSimilarityMethod* stehen dem *WsdLink* zwei verschiedene Verfahren zur Bestimmung der Ähnlichkeit zwischen den Bezeichnern der Textelemente und den Namen der Modellelemente zur Verfügung. Zum einen ist das die *LevenshteinDistance* und zum anderen die *JaroWinklerDistance*. Für beide Verfahren kann ein Schwellenwert definiert werden, ab dem eine Verknüpfung zwischen einem Text- und Modellelement erstellt wird. Die gefundenen Verknüpfungen zwischen Text- und Modellelementen werden als *ReferencedModelElement* vom *WsdLink* an den *WsdLinkAgent* zurückgegeben.

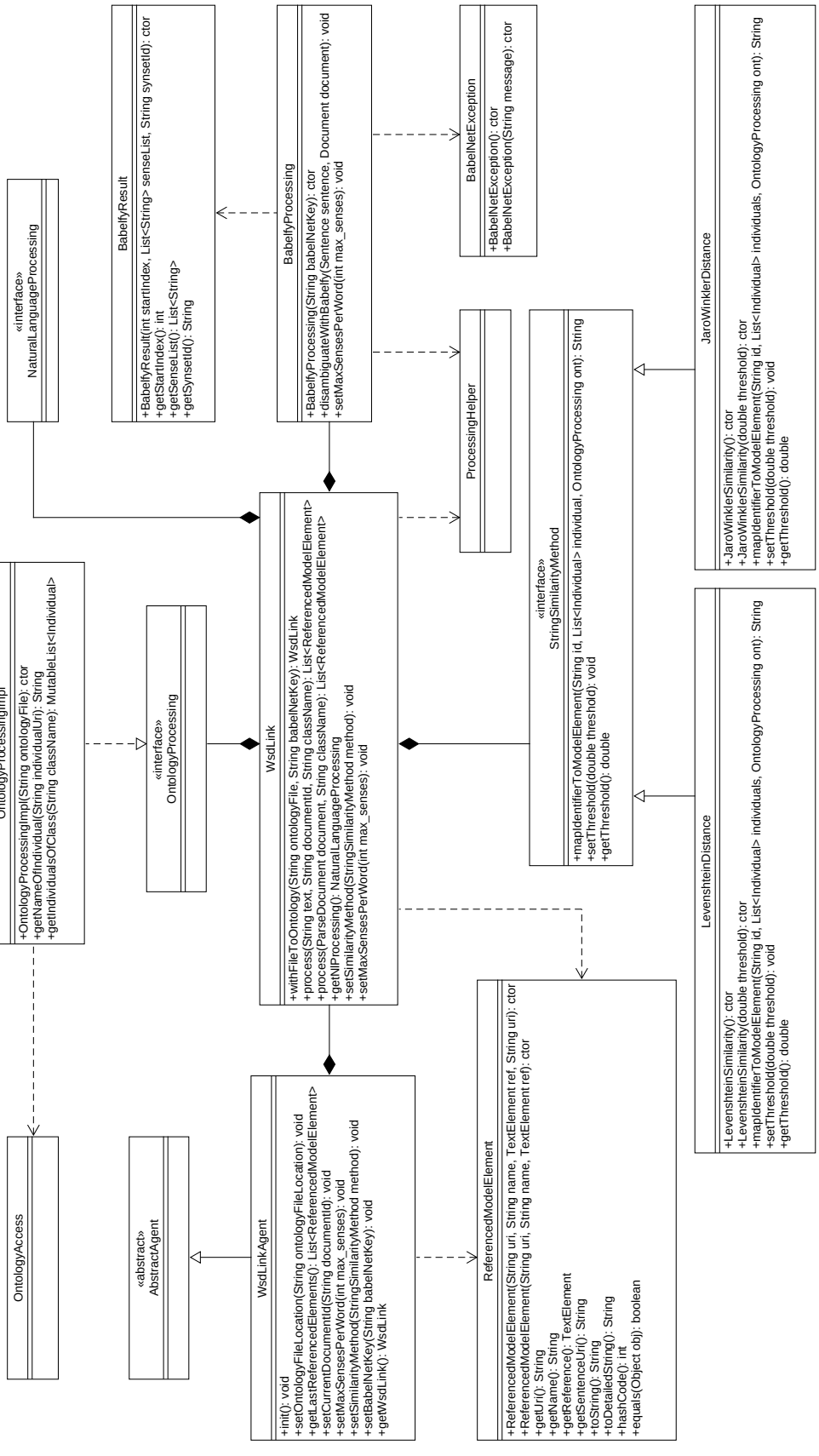


Abbildung 4.3.: UML-Diagramm des Ansatzes für den Agenten

5. Implementierung

In diesem Kapitel wird beschrieben, wie der Synonym-basierte Vergleichsansatz zum Erstellen der Verknüpfungen zwischen den Textelementen der SAD und den Modellelementen des SAM umgesetzt wird. Dafür wird überprüft, welche Textelemente der SAD welchen Modellelementen des SAM zugeordnet werden können. Dabei besteht die Hauptaufgabe darin, Bezeichnern von Textelementen den Namen von Modellelementen zuzuordnen. Eine Zuordnung zwischen Text- und Modellelementen, die dasselbe Element abbilden, aber semantisch unterschiedlich benannt sind, erfolgt über die in den Synsets enthaltenen Synonyme der Textelemente. Gefundene Verknüpfungen zwischen Text- und Modellelementen werden in einem aus dem Text aufgebauten Graphen annotiert. Die Umsetzung erfolgt in Java unter Verwendung von Maven.

In Abschnitt 5.1 wird die Vorgehensweise des Synonym-basierten Vergleichsansatzes erläutert. Die grundlegenden Teilschritte werden in Abschnitt 5.2 und Abschnitt 5.3 im Detail ausgeführt. Dazu gehört zum einen die Bestimmung der Wortbedeutungen im Kontext der SAD mit Babelfy. Und zum anderen das Erstellen der Verknüpfungen zwischen den Text- und Modellelementen mit den annotierten Synonymen der jeweiligen Wortbedeutung.

5.1. Vorgehensweise des Ansatzes

Der Agent des Synonym-basierten Vergleichsansatz verfügt über die SAD, die durch ein Textdokument abgebildet und in Form eines Graphen verarbeitet wird. Die Wörter des Dokuments werden durch Knoten des Graphen dargestellt. Das Vorgehen des Agenten (*WsdLinkAgent*) lässt sich in drei Schritte unterteilen. Diese sind in Abbildung 5.1 in Form eines Aktivitätsdiagramms dargestellt und farblich gekennzeichnet. Im ersten Schritt (gelb gekennzeichnet) wird der Agent mit dem SAM und einem API-Schlüssel für den WSD-Algorithmus initialisiert. Das SAM wird in Form einer Ontologie als OWL-Datei dem Agenten übergeben. Der Agent verfügt über einen *WsdLink*, welcher den Einstiegspunkt zur Verarbeitung des Textdokuments und der OWL-Datei darstellt. Dieser wird vom Agenten mit der OWL-Datei und dem *BabelNetKey* initialisiert. Bevor Verknüpfungen zwischen den Text- und Modellelementen erstellt werden, gilt es die Modellelemente aus der OWL-Datei zu extrahieren. Das Auslesen dieser Informationen der Ontologie erfolgt mit OntologyProcessing. Mit einem vom Anwender spezifizierten Klassennamen werden die für die Verknüpfung zu betrachtenden Modellelemente der OWL-Datei entnommen. Im zweiten Schritt (in Abbildung 5.1 blau gekennzeichnet) erfolgt die Generierung der Bedeutungen in Form der Synsets, unter Verwendung des WSD-Algorithmus Babelfy mit *BabelfyProcessing*. Die Bedeutungen der Wörter des Textdokuments werden im Kontext des Satzes mit diesem WSD-Algorithmus bestimmt. Die in den Synsets enthaltenden Synonyme werden unter dem Attributnamen *„senses“* des Wortknotens im Graphen an-

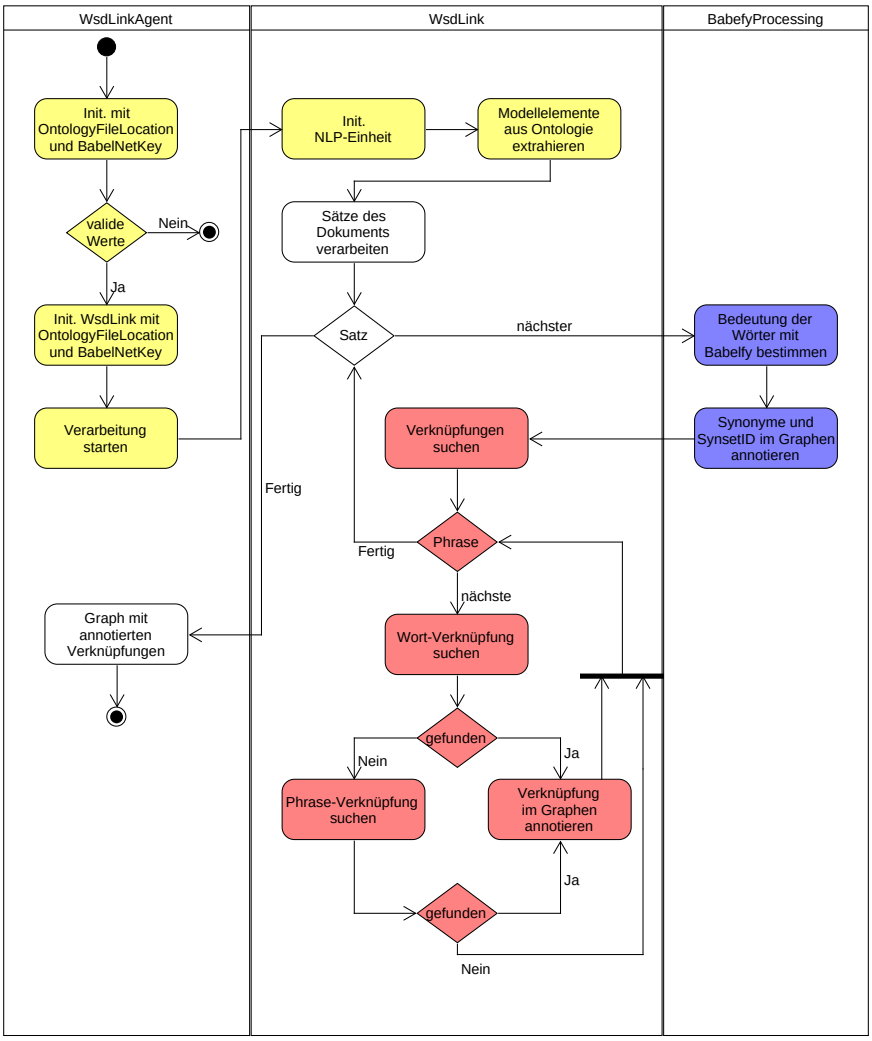


Abbildung 5.1.: Aktivitätsdiagramm für den Synonym-basierten Vergleichsansatz mit den wichtigsten Verarbeitungsstufen farblich gekennzeichnet

notiert. Das genaue Vorgehen wird in Abschnitt 5.2 beschrieben. Im dritten Schritt (in Abbildung 5.1 rot gekennzeichnet) werden die expliziten Verknüpfungen zwischen Text- und Modellelementen gesucht und gegebenenfalls im Graphen annotiert. Dafür werden die Phrasen und die Wörter der Sätze des Textdokuments der SAD betrachtet. Zuerst wird versucht eine Verknüpfung zwischen einem Wort und einem Modellelement herzustellen. Ist dies nicht erfolgreich, wird versucht eine Verknüpfung zwischen der gesamten Phrase und einem Modellelement herzustellen. Wird eine Verknüpfung gefunden, wird diese im Graphen annotiert und danach die nächste Phrase oder der nächste Satz der SAD betrachtet. Dieser Schritt wird in Abschnitt 5.3 im Detail erläutert. Für jeden Satz wird erst Schritt zwei und dann für jede Phrase des Satzes, die mindestens ein Nomen enthält, Schritt drei ausgeführt. Die Verarbeitung durch den Agenten ist abgeschlossen, nachdem für alle Sätze des Textdokuments abwechselnd Schritt zwei und drei ausgeführt wurden. Der Graph mit

den Annotationen steht dem Agenten für die Weiterverarbeitung zur Verfügung und kann an andere Agenten weitergegeben werden.

5.2. Auflösung sprachlicher Mehrdeutigkeiten mit Babelfy

In diesem Abschnitt wird Schritt zwei, der Vorgehensweise des Synonym-basierten Vergleichsansatzes erläutert. Dieser ist im Aktivitätsdiagramm in Abbildung 5.1 blau gekennzeichnet. Dabei wird im Folgenden erklärt, wie die Bedeutungen der Wörter des Textdokuments bestimmt und im Graphen annotiert werden.

Zur Umsetzung des Synonym-basierten Vergleichsansatzes wird ein WSD-Algorithmus benötigt. Es gibt verschiedene Verfahren zur Auflösung sprachlicher Mehrdeutigkeiten, wie in Abschnitt 3.2 beschrieben. Deshalb ist die Implementierung so aufgebaut, dass der konkrete WSD-Algorithmus durch einen anderen ersetzt werden kann. In dieses System ist ein WSD-Algorithmus integriert, der auf lexikalischen Ressourcen basiert. Während der Implementierung wurden zwei verschiedene WSD-Algorithmen zur Auflösung der Wortbedeutungen, abhängig des Kontextes, in das System integriert. Die Anzahl frei zur Verfügung stehender Java-Frameworks für WSD ist beschränkt. Aus diesem Grund wurde als klassischer Algorithmus *Simplified-Lesk* und als State-of-the-art-Algorithmus *Babelfy* für WSD in Betracht gezogen.

Beim Simplified-Lesk-Algorithmus wird der Kontext, in dem ein Wort disambiguiert werden soll, explizit angegeben. Die Bedeutung des Wortes im angegebenen Kontext wird über die Definitionen der möglichen Bedeutungen des Wortes bestimmt (wie in Unterabschnitt 2.3.1 beschrieben) [21]. Mit dem Lesk-Algorithmus kann abhängig des getesteten Datensatzes ein F_1-Maß von 53,6% erreicht werden [21]. Der Babelfy-Algorithmus von Moro et al. [19] liefert bessere Ergebnisse. Mit diesem Algorithmus werden F_1-Werte von 69,2% [19] beziehungsweise 70,3% [21] gemessen. Babelfy ist ein graphbasierter WSD-Ansatz, der als zugrundeliegende lexikalischer Ressource BabelNet verwendet. Bei der Disambiguierung werden zufällige Pfade des Graphen durchlaufen und Verbindungen zwischen den Synsets bestimmt (wie bereits in Unterabschnitt 2.3.2 erwähnt). Aufgrund der besseren F_1-Werte erfolgt die Umsetzung des Synonym-basierten Vergleichsansatzes unter Verwendung des Babelfy-Algorithmus für WSD.

Bei einer Anfrage an Babelfy werden die Parameter *text*, *lang*, *key*, *annTypes* und *cands* beim Synonym-basierten Vergleichsansatz gesetzt. Der Parameter *text* entspricht dem Satz, der mit Babelfy disambiguiert werden soll. Der Satz dient dabei zusätzlich als Kontext, in dem die Bedeutungen der im Satz enthaltenen Wörter bestimmt werden. Der Parameter *lang* entspricht der Sprache der lexikalischen Ressource mit der die Bedeutungen aufgelöst werden. Der Synonym-basierte Vergleichsansatz ist auf Texte ausgelegt, die in englischer Sprache verfasst sind. Die SAD des Agenten muss somit in englischer Sprache vorliegen und es wird der Parameter *lang = en* gesetzt. Bei jeder Anfrage wird ein BabelNet-API-Schlüssel benötigt, der mit dem Parameter *key* übergeben wird. Abgesehen von diesen Pflichtparametern, werden zwei optionale Parameter bei den Babelfy-Anfragen im Agenten gesetzt. Die Ergebnisse werden mit *annType=CONCEPTS* auf WSD-Ergebnisse eingeschränkt, weshalb dadurch kein EL durchgeführt wird. Nach eigenen Erfahrungen sind die EL-Ergebnisse für eine Verknüpfung zwischen Text- und Modellelementen nicht zielführend. Mit *cands=TOP*

werden die Ergebnisse von Babelfy geordnet und nur das beste Ergebnis zurückgegeben [18, 17]. Weitere optionale Parameter, die beim Synonym-basierten Vergleichsansatz nicht gesetzt werden, sind in den Grundlagen in Unterabschnitt 2.3.2 aufgelistet. Eine HTTP-Anfrage an Babelfy mit den erwähnten Parametern ist zur Disambiguierung eines Satzes in Beispiel 1 gegeben:

Beispiel 1: *„https://babelfy.io/v1/disambiguate?*
***text**=Media Access is responsible for fetching information about audio files from the Database and the audio files themselves from Data Storage.*
*&**lang**=en*
*&**annType**=CONCEPTS*
*&**cands**=TOP*
*&**key**=60f97c66-b785-4fb3-a48c-12c650ee8383“*

Die beschriebenen Parameter *text*, *lang*, *annType*, *cands* und *key* werden in der HTTP-Anfrage gesetzt. Der Satz des Parameters *text* ist aus der SAD des Media Stores [23] entnommen. Mit Babelfy wird bestimmt, welche Bedeutungen die einzelnen Wörter im Kontext dieses Satzes haben. Beispielsweise wird bestimmt, welche Bedeutung das Wort *Access* im angegeben Satz hat. Die Wörter *Access*, *Data* und *Storage* weisen in Kombination auf eine Bedeutung im Kontext der Informatik hin. Babelfy weist dem Wort *access* das Synset mit der BabelNetID *bn:00000665n* zu. Die Definition dieser Bedeutung ist: „The operation of reading or writing stored information (computer science)„ [18, 17]. Als Synonym wird das Wort *memory access* zurückgegeben.

Die Babelfy-Ergebnisse für alle Wörter dieser Anfrage sind in Tabelle 5.1 aufgelistet [18, 17]. Mit Babelfy werden den Nomen, Verben, Adjektiven und Adverbien des Satzes jeweils die bestgeeignete Bedeutung im Kontext des Satzes zugewiesen. Jede dieser Bedeutungen wird durch ein Synset dargestellt und über eine ID eindeutig identifiziert (zweite Spalte der Tabelle 5.1). Abhängig vom POS-Tagging durch den Babelfy-Algorithmus steht der letzte Buchstabe der Synset-ID für die jeweilige Wortart. Dementsprechend steht (n) für Nomen, (v) für Verben, (a) für Adjektive und (r) für Adverbien [19]. Der Babelfy-Algorithmus liefert als Ergebnis eine Bedeutung, die durch eine *BabelSynsetID* eindeutig bestimmt ist. Mit einer weiteren HTTP-Anfrage an BabelNet wird diese ID aufgelöst und die Synonyme des Synsets in Form einer Liste zurückgegeben. Die BabelSynsetID entspricht dabei dem Synset, der durch den Babelfy-Algorithmus zugewiesenen Wortbedeutung in BabelNet. Und die Synonyme entsprechen den Wörtern des Synsets (dritte Spalte der Tabelle 5.1).

Bei der Benennung von Modellelementen wird teilweise der Name aus mehreren Teilwörtern zusammengesetzt und beispielsweise in der *CamelCase*-Schreibweise angegeben. Ein Beispiel dafür sind die Namen *MediaAccess* und *DataStorage*. Diese Namen sind aus mehreren Teilwörtern gebildet und jedes Teilwort wird mit einem Großbuchstaben begonnen. Für die HTTP-Anfrage an Babelfy ist es nötig, diese Wörter in ihre Teilwörter aufzutrennen. In der Anfrage werden diese in Form von *Media Access* und *Data Storage* übergeben. Nachdem die Synonyme der Teilwörter mit Babelfy und BabelNet bestimmt sind, werden die Synonyme wieder zusammengeführt. Dafür wird jedes Synonym des ersten Teilwortes mit jedem Synonym des zweiten Teilwortes kombiniert. Dabei erhält man

Wort	BabelSynsetID	Synonyme
Media	bn:00048459n	journalism, news media, media
Access	bn:00000665n	access, memory access
responsible	bn:00109742a	responsible
fetching	bn:00084061v	bring, get, convey, fetching
information	bn:00046698n	information, info
audio	bn:13731704a, bn:01627315n	audio, audio file format, audio file
files	bn:00021475n	computer file, data file, file, files
Database	bn:00025333n	database, db, database management system
Data	bn:00025327n	datum, data point, data points, data
Storage	bn:00074446n	storage

Tabelle 5.1.: Ergebnisse des Babelfy-Algorithmus für einen Beispielsatz [18, 17]

Bezeichner	BabelSynsetID	Kombinierte Synonyme
MediaAccess	bn:00048459n, bn:00000665n	journalism access, journalism memory access, news media access, news media memory access, media access, media memory access
DataStorage	bn:00025327n, bn:00074446n	datum storage, data point storage, data points storage, data storage

Tabelle 5.2.: Kombinierte Synonyme für Bezeichner, die aus mehreren Teilwörtern zusammengesetzt sind

die in Tabelle 5.2 dargestellten kombinierten Ergebnisse. Zu welchen Problemen dieses Verfahren führen kann und was dabei beachtet werden muss, wird in Unterabschnitt 5.2.1 behandelt.

In Abbildung 5.2 ist dargestellt, wie der Graph aus den Wörtern des Satzes aufgebaut ist. Ein Wort stellt einen Knoten vom Typ „token“ dar. Die einzelnen Knoten stehen in Relation zueinander und jeder dieser Knoten verfügt über verschiedene Attribute, wie *value*, *sentence*, *pos* (für POS-Tagging) und *position*. Das Attribut *value* entspricht dabei dem Wort selbst, *sentence* im wievielten Satz sich das Wort in der SAD befindet und *position* gibt die Wortposition im Satz an. Dem Knoten sind noch weitere Attribute zugeordnet und es können auch neue Attribute definiert werden. Für die Ergebnisse des Babelfy-Algorithmus werden die neuen Attribute „senses“ und „synsetID“ definiert. Die von Babelfy bestimmten BabelSynsetID's werden im Graphen, als Attribute des Wortknotens mit dem Attributnamen „synsetID“, annotiert. Die Synonyme, beziehungsweise die kombinierten Synonyme der zusammengesetzten Wörtern, werden als Attribute des Wortknotens unter dem Attributnamen „senses“ annotiert. Für Wörter des Satzes, für die keine Bedeutung im Kontext des Satzes bestimmt werden kann, ist der Wert der Attribute „senses“ und „synsetID“ ein leerer String.

Nachdem alle Wörter des Satzes mit den Synonymen annotiert sind, ist der Verarbeitungsschritt zwei für diesen Satz abgeschlossen. Daraufhin wird Verarbeitungsschritt drei, das Erstellen von Verknüpfungen eingeleitet. In den folgenden zwei Unterkapiteln wird auf Details bei der Auflösung von sprachlichen Mehrdeutigkeiten mit Babelfy eingegangen.

Abbildung 5.2.: Wortknoten des Graphen nach Annotation mit „senses“ und „synsetID“

5.2.1. Anzahl Bedeutungen pro Wort

Die Anzahl Synonyme pro Wort, die im Graphen unter „senses“ annotiert werden, wird im Synonym-basierten Vergleichsansatz durch das Attribut „*maxSensesPerWord*“ begrenzt. Die Anzahl an Synonymen pro Wort, die durch den Babelfy-Algorithmus generiert werden, ist nach oben nicht beschränkt. Je mehr Synonyme pro Wort annotiert werden, desto mehr Begriffe müssen vom Vergleichsansatz mit den Namen der Modellelemente verglichen werden.

Die Bezeichner der Textelemente in der SAD bestehen nicht notwendigerweise aus einem, sondern können aus mehreren zusammengesetzten Teilwörtern bestehen. Beispielsweise besteht der Bezeichner des Textelements *MediaAccess* aus den zwei Teilwörtern *Media* und *Access*. Für eine obere Grenze von drei Synonymen pro Wort wird sowohl das Wort *Media*, als auch das Wort *Access* mit bis zu drei Synonymen versehen. Angenommen die Wörter *memory, journalism* und *medium* sind die Synonyme für das Wort *Media* und die Wörter *admittance, entrance* und *approach* für das Wort *Access*. Wird jedes Synonym von *Media* mit jedem Synonym von *Access* kombiniert, erhält man nicht drei, sondern neun mögliche Kombinationen, die unter „senses“ im Graphen annotiert werden, dargestellt in Beispiel 2.

Beispiel 2: *„[memory admittance, memory entrance, memory approach, medium admittance, medium entrance, medium approach, journalism admittance, journalism entrance, journalism approach]“*

Ist das Teilwort selbst nicht Teilmenge der Synonyme, wird es diesen noch hinzugefügt. Da die Wörter *media* und *access* nicht Teilmenge der zuvor genannten Synonyme sind, müssen der Liste aus Beispiel 2 die fehlenden Kombinationen aus Beispiel 3 hinzugefügt werden.

Beispiel 3: *„[media admittance, media entrance, media approach, media access, memory access, medium access, journalism access]“*

Damit ergeben sich insgesamt sechzehn mögliche Kombinationen, die unter dem Attributnamen „senses“ im Graphen für den Wortknoten *MediaAccess* annotiert werden. Wie man in der Tabelle 5.3 sieht, existieren für ein Modellelement dessen Name aus zwei Teilwörtern besteht bei drei Synonymen pro Wort, bis zu sechzehn mögliche Kombinationen dieser Synonyme. Dieser Wert kann der Tabelle 5.3 in Zeile vier (Synonyme/ Wort = 3) und Spalte drei (zwei Teilwörter) entnommen werden. In der Tabelle 5.3 ist dargestellt, wie sich die Anzahl an Kombinationen, abhängig der Anzahl Synonyme pro Wort und Teilwörter pro Bezeichner verhält. Besteht der Name des Modellelements nicht aus zwei, sondern aus drei

Synonyme/ Wort	Ein Wort	Zwei Teilwörter	drei Teilwörter	vier Teilwörter
0	1	1	1	1
1	2	4	8	16
2	3	9	27	81
3	4	16	64	256
4	5	25	125	625
s	$(s+1)^1$	$(s+1)^2$	$(s+1)^3$	$(s+1)^4$

Tabelle 5.3.: Anzahl Synonyme pro Wort in Bezug auf die Anzahl Teilwörter pro Bezeichner im Text

Teilwörtern, wie beispielsweise *UserDBAdapter*, erhält man nicht *16*, sondern *64* mögliche Kombinationen. Für einen Namen mit a Teilwörtern und s Synonyme pro Wort, existieren maximal n mögliche Kombinationen:

$$1 \leq n \leq (s+1)^a \qquad \forall s > 0$$

Diese Formel gibt die maximale Anzahl möglicher Kombinationen für alle $s > 0$ an. Es gilt $s+1$ für den Fall, dass das zu disambiguierende Wort nicht Teilmenge der Synonyme ist. Ist das der Fall, wird es der Menge zu kombinierender Begriffe noch hinzugefügt und damit existieren endgültig $s + 1$ Synonyme pro Wort. Nach eigenen Erfahrungen entspricht dies dem *Worst-Case*-Szenario. In den meisten Fällen lässt sich die maximale Anzahl Kombinationen pro Wort mit $n \leq s^a$ bestimmen, da das Wort selbst in der Menge der Synonyme enthalten ist. Abhängig der durch BabelNet zurückgegebenen Reihenfolge der Liste mit den Synonymen kann es sein, dass das Wort selbst erst an dritter oder vierter Stelle steht und damit nicht Teilmenge der Synonyme ist. Für ein kleines s kann dies nach eigenen Beobachtungen häufiger vorkommen. Alle Wörter, die den Wortarten Nomen, Verben, Adjektiven und Adverbien angehören, werden immer mindestens mit sich selbst als „senses“ annotiert. Für den Sonderfall *s = 0* folgt, dass der Wortknoten von *MediaAccess* im Graphen unter dem Attributnamen „senses“ mit „[*media access*]“ annotiert wird. Damit existiert für *s = 0* eine Kombination, nämlich der Bezeichner selbst, unabhängig der Anzahl Teilwörter a.

Damit lässt sich festhalten, dass beim Setzen des Wertes für die Anzahl Synonyme pro Wort beachtet werden muss, wie sich dieser Wert auf die Anzahl der als „senses“ annotierten Begriffe auswirkt. Besonders bei der Verarbeitung von Bezeichnern, die aus mehreren Teilwörtern bestehen, wirkt sich eine Erhöhung der Anzahl Synonyme pro Wort negativ auf die Laufzeit des Programms aus. Beispielsweise bewirkt eine Erhöhung von zwei auf zehn Synonyme pro Wort eine Laufzeiterhöhung von einer auf drei Minuten[1]. Die Festlegung dieses Wertes stellt ein Kompromiss aus Laufzeit und Ausbeute dar. Je höher der Wert für die Anzahl Synonyme pro Wort, desto wahrscheinlicher ist es, eine Verknüpfung zwischen einem Text- und einem Modellelement herzustellen (wie die Evaluation in Abschnitt 6.2 zeigt).

[1]Für Version 1 des Media Stores mit der Levenshtein-Distanz, Single-Threaded unter Verwendung eines Ryzen 5 2600 Prozessors

5.2.2. Extraktion und Rückordnung der Babelfy-Ergebnisse

Bei einer HTTP-Anfrage an Babelfy wird mit dem Parameter *text* in der URL der zu disambiguierende Satz übergeben. Dieser Satz enthält Leerzeichen, wie in Beispiel 1 dargestellt wurde. Diese Leerzeichen werden kodiert, sodass eine valide URL entsteht. Bezeichner des Textdokuments, die aus mehreren Teilwörtern zusammengesetzt sind, werden bei Anfragen an Babelfy nicht durch ein Leerzeichen, sondern durch einen Tab-Character *\t* getrennt. Beispielsweise wird das Wort *MediaAccess* als *Media\t Access* und das Wort *UserDBAdapter* als *User\t DB\t Adapter* in der HTTP-Anfrage an Babelfy übergeben. Babelfy generiert für das zusammengeschriebene Wort *MediaAccess* keine Ergebnisse, für die Teilwörter *Media* und *Access* hingegen schon. Das Einfügen des Tab-Characters ermöglicht, dass die Teilwörter eines Bezeichners von Babelfy einzeln disambiguiert werden und gleichzeitig die Ergebnisse der Teilwörter wieder dem ursprünglichen Wortknoten zurück geordnet werden können.

Die Ergebnisse des Babelfy-Algorithmus werden über einen Start- und Endindex eindeutig identifiziert. Babelfy ordnet den Wörtern des angefragten Satzes eine Bedeutung im Kontext dieses Satzes zu. Die Bedeutungen der Babelfy-Antwort werden durch das Attribut *babelSynsetID* eindeutig einem Synset in BabelNet zugeordnet. Für jede *babelSynsetID* wird eine weitere HTTP-Anfrage an BabelNet gestellt. Das Ergebnis dieser Anfrage ist eine Liste von Synonymen für die Bedeutung des Wortes im Kontext des angefragten Satzes. Diese Synonyme werden für die Verknüpfung von Bezeichnern der Textelemente mit den Namen der Modellelementen verwendet.

Die Ergebnisse werden über den Start- und Endindex der Babelfy-Antwort den Wortknoten im Graphen zugeordnet und annotiert. Die Rückordnung der Bedeutungen zu den Wortknoten erfolgt über den Startindex. Dieser entspricht der Indexposition des Wortes im übergebenen Satz der HTTP-Anfrage mit Tab-Characters. Diese Tab-Characters helfen, um von dieser Indexposition wieder auf den ursprünglichen Startindex des Wortes im Satz zu kommen, bevor eine Auftrennung der Bezeichner in deren Teilwörter stattgefunden hat. Dafür wird die Anzahl an Tab-Characters, die im Satz vor einem Wort eingefügt wurden vom aktuellen Startindex abgezogen. Damit ist es möglich die Babelfy-Ergebnisse der Teilwörter *Media* und *Access* dem Wortknoten *MediaAccess* zuzuordnen. Dafür werden Bezeichner, die aus mehreren Teilwörtern zusammengesetzt sind, über die jeweiligen Startindexe der Ergebnisse der Teilwörter dem gleichen Wortknoten zugeordnet. Die BabelSynsetID und Synonyme der Teilwörter werden, wie in Unterabschnitt 5.2.1 beschrieben, miteinander kombiniert und daraufhin im Graphen annotiert. Für Bezeichner, die nicht aus mehreren Teilwörtern zusammengesetzt sind, kann über den Startindex direkt ein Wortknoten zugeordnet werden. Der Wortknoten wird im Graphen mit der *babelSynsetID* unter dem Attributnamen „synsetID“ und den Synonymen unter dem Attributnamen „senses“ annotiert. Die Anzahl an Synonymen die annotiert werden, richtet sich nach dem Wert, der vom Anwender für die maximale Anzahl an Synonymen pro Wort definiert wurde.

5.3. Verknüpfung von Text- und Modellelementen

In diesem Abschnitt wird der dritte Schritt der Vorgehensweise des Synonym-basierten Vergleichsansatzes erläutert. Dieser Schritt ist im Aktivitätsdiagramm in Abbildung 5.1 rot gekennzeichnet. Dabei wird im Folgenden erklärt, wie das Vorgehen des Synonym-basierten Vergleichsansatzes abläuft. Zum einen wird erläutert, wie Verknüpfungen zwischen den Text- und Modellelementen gefunden werden und zum anderen, wie diese Verknüpfungen im Graphen annotiert werden.

Die Verknüpfung von Text- und Modellelementen erfolgt, nachdem die Synonyme der Wörter im Graphen in Schritt zwei annotiert wurden. In jedem Satz werden diejenigen Phrasen betrachtet, die mindestens ein Nomen enthalten. Das Überprüfen von möglichen Verknüpfungen zwischen Text- und Modellelementen lässt sich in zwei Stufen gliedern. In der ersten Stufe wird für jedes Wort einer Phrase versucht, eine Verknüpfung mit einem Modellelement herzustellen. Ist dies erfolgreich, wird die weitere Suche für diese Phrase beendet und die gefundene Verknüpfung im Graphen annotiert. Hat die erste Stufe zu keiner erfolgreichen Verknüpfung geführt, wird die zweite Stufe eingeleitet. In der zweiten Stufe wird nicht für ein Wort der Phrase, sondern für die gesamte Phrase versucht eine Verknüpfung mit einem Modellelement herzustellen. Dafür werden die Synonyme der einzelnen Wörter miteinander kombiniert, wie bei Bezeichnern die aus mehreren Teilwörtern bestehen (siehe Tabelle 5.2 in Abschnitt 5.2). Ist die Suche nach einer Verknüpfung mit einem der kombinierten Wörtern erfolgreich, wird die gefundene Verknüpfung, wie in der ersten Stufe, im Graphen annotiert. Die erste Stufe, das Erstellen von Verknüpfungen zwischen einzelnen Wörtern und Modellelementen wird vor dem Erstellen von Verknüpfungen zwischen der gesamten Phrase und den Modellelementen ausgeführt. Diese Reihenfolge ist auf Einsparungen bei der Laufzeit des Synonym-basierten Vergleichsalgorithmus begründet. Die Anzahl Begriffe, die für einen Vergleich mit den Namen der Modellelemente betrachtet werden, wächst mit der Anzahl Wörter pro Phrase und damit die Laufzeit des Ansatzes (siehe die Evaluation in Abschnitt 6.2). Ist die Suche nach einer Verknüpfung nicht erfolgreich, wird die nächste Phrase des Satzes oder der nächste Satz des Dokuments betrachtet. Diese zwei Stufen werden im Folgenden im Detail erläutert.

Für das Erstellen von Verknüpfungen zwischen Text- und Modellelementen wird der Satz in Beispiel 4 betrachtet. Die Phrasen dieses Satzes, die mindestens ein Nomen enthalten und für eine Verknüpfungen zwischen Text- und Modellelementen betrachtet werden, sind unterstrichen.

Beispiel 4: *„The UserDBArranger component queries the database and the Account Management component answers the requests for registration and authentication.“*

Angenommen es existieren die Modellelemente *UserManagement*, *DB* und *UserDBAdapter*. Die unterstrichenen Textelemente des Satzes sollen diesen Modellelementen zugeordnet werden. Dafür werden für die Wörter der Phrasen, die im Graphen unter „senses“ annotierten Synonyme ausgelesen. Bei Betrachtung der ersten Phrase *The UserDBArranger component* ist das Vorgehen folgendermaßen: Dem Bezeichner *The* sind keine Synonyme zugeordnet, da dieser keiner der Wortarten Nomen, Verben, Adjektive und Adverbien angehört. Als nächstes wird der Bezeichner *UserDBArranger* betrachtet. Im Kontext des

Wort	BabelSynsetID	Synonyme
User	bn:000079373n	account, user
DB	bn:00025333n	database, db
Arranger	bn:000001269n	adapter, arranger
UserDBArranger	bn:000079373n, bn:00025333n, bn:000001269n	account database adapter, account db adapter, account database arranger, account db arranger, user database adapter, user db adapter, user database arranger, user db arranger

Tabelle 5.4.: Bedeutungen der Teilwörter ergeben kombiniert die annotierten Synonyme des Wortes

gegebenen Satzes wurden im vorherigen Arbeitsprozess die in Tabelle 5.4 dargestellten Synonyme mit Babelfy bestimmt [18, 17]. Die letzte Zeile der Tabelle entspricht den im Graphen annotierten Synonymen des Wortknotens *UserDBArranger*, die für eine Verknüpfung mit den Modellelementen betrachtet werden.

Bevor versucht wird eine Verknüpfung mit den synonymen Begriffen und den Modellelementen herzustellen, werden dieser Liste zusätzlich alle Teilwörter der synonymen Begriffe hinzugefügt. Dies entspricht der Potenzmenge, die über die Wörter eines synonymen Begriffs gebildet wird. Damit existieren Begriffe in der Liste die aus drei, zwei oder nur einem Synonym der Teilwörter von *UserDBAdapter* bestehen. Damit ergibt sich die folgende Liste mit den Begriffen in Beispiel 5, die verwendet wird, um eine Verknüpfung zwischen dem Textelement *UserDBArranger* und den Modellelementen herzustellen:

Beispiel 5: *[account database adapter, account database arranger, user database arranger, user database adapter, account db arranger, account db adapter, database arranger, user db arranger, account database, account arranger, database adapter, user db adapter, account adapter, user database, user arranger, user adapter, db arranger, account db, db adapter, database, arranger, user db, account, adapter, user, db,]*

Die Elemente der Liste werden der Wortlänge nach sortiert und der Reihe nach für eine Verknüpfung mit den Modellelementen betrachtet. Dadurch wird sichergestellt, dass für ein Teilwort keine Verknüpfung mit einem Modellelement erstellt wird, wenn für die kombinierten Teilwörter eine Verknüpfung mit einem anderen Modellelement existiert. Beispielsweise gehört das Wort *db* zur Liste der Begriffe, die für eine Verknüpfung mit dem Textelement *UserDBArranger* und den Modellelementen betrachtet werden. Das Wort *db* kann dem Modellelement *DB* zugeordnet werden. Die Wörter *user db adapter* gehören auch zur Liste der Begriffe, die für eine Verknüpfung des Textelements *UserDBArranger* und den Modellelementen betrachtet werden. Die Wörter *user db adapter* können dem Modellelement *UserDBAdapter* zugeordnet werden. Damit gibt es zwei mögliche Modellelemente (*DB* und *UserDBAdapter*), die dem Textelement *UserDBArranger* zugeordnet werden können. Da der Bezeichner des Textelements *UserDBArranger* eine höhere Ähnlichkeit zu dem Namen des Modellelements *UserDBAdapter*, als zu dem Namen des Modellelements *DB* aufweist, wird eine Verknüpfung zwischen dem Textelement *UserDBArranger* und dem Modellelement *UserDBAdapter* erstellt. Es wird keine Verknüpfung zwischen dem Textelement *UserDBArranger* und dem Modellelement *DB* erstellt. Die Wörter *user db*

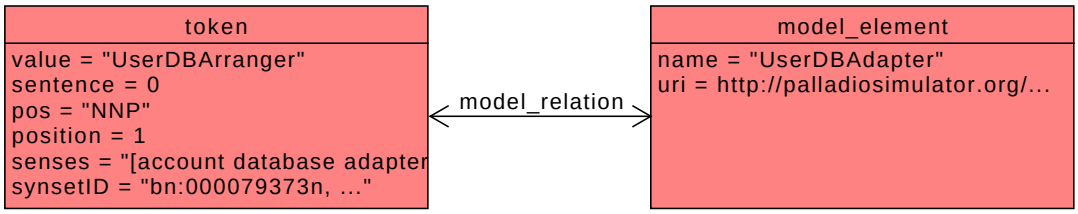


Abbildung 5.3.: Verknüpfung im Graphen zwischen einem Text- und Modellelement

adapter stehen in der Liste weiter vorne, als das Wort *db* und werden zuerst für einen Verknüpfung betrachtet. Damit soll erreicht werden, dass eine Verknüpfung mit dem Modellelement erstellt wird, für das die Übereinstimmung mit dem ursprünglichen Bezeichner am Höchsten ist. Dieses Vorgehen beruht auf der Idee, dass wenn für einen Bezeichner keine Verknüpfung gefunden wird, die Teilwörter des Bezeichners für eine Verknüpfung betrachtet werden. Stimmt ein Teilwort des Bezeichners mit dem Namen eines Modellelements überein, beziehungsweise ist diesem sehr ähnlich, wird eine Verknüpfung zwischen dem Bezeichner und dem Modellelement erstellt. Existiert beispielsweise nur das Modellelement *DBAdapter* und das Textelement *UserDBArranger*, wird für die Teilwörter *db adpater* eine Verknüpfung mit dem Modellelement erstellt. Damit wird eine maximale Ausbeute erreicht, was jedoch unter Umständen zu Einbüßen bei der Präzision führt. Die Verfahren zur Bestimmung der Ähnlichkeit sind die Levenshtein-Distanz und die Jaro-Winkler-Distanz. Ist die Ähnlichkeit kleiner als der definierten Schwellenwert des jeweiligen Verfahrens, wird eine Verknüpfungen zwischen dem Text- und Modellelement erstellt.

Zum Erstellen einer Verknüpfung zwischen einem Text- und Modellelement, wird das Modellelement als neuer Knoten dem Graphen hinzugefügt. Modellelemente erhalten den Knotentypen „model_element“ und werden dem Graphen neu hinzugefügt, wenn sie noch nicht enthalten sind. Ein Modellelement verfügt über die Attribute *name* und *uri*. Das Attribut *name* entspricht dem Namen des Modellelements im SAM und das Attribut *uri* entspricht der eindeutigen Identifikation des Elements in der Ontologie. Wörter werden im Graphen unter dem Knotentypen „token“ geführt. Die Verknüpfung wird durch eine Kante im Graphen zwischen den zwei Knoten umgesetzt. Kanten zwischen Text- und Modellelementen haben den Typ „model_relation“. In Abbildung 5.3 ist eine Verknüpfung zwischen dem Textelement *UserDBArranger* und dem Modellelement *UserDBAdapter* dargestellt.

Für den Bezeichner *UserDBArranger* der Phrase *The UserDBArranger component*, kann eine Verknüpfung mit einem Modellelement hergestellt werden. Aus diesem Grund wird nicht weiter versucht, mit dem Bezeichner *component* eine Verknüpfung mit einem Modellelement herzustellen. Stattdessen wird die nächste Phrase des Satzes, *the database*, betrachtet. Für das Textelement *database* wird, mit demselben Verfahren wie für *UserDBArranger*, eine Verknüpfung zu dem Modellelement *DB* hergestellt. In beiden Fällen ist die Suche nach einer Verknüpfung nach der ersten Stufe abgeschlossen.

In der nächsten Phrase *the Account Management component* kann mit keinem der Wörter eine Verknüpfung zu einem Modellelement hergestellt werden. Aus diesem Grund wird die

zweite Stufe eingeleitet. Es wird versucht, eine Verknüpfung zwischen der gesamten Phrase und einem Modellelement herzustellen. Die Wörter der Phrase werden behandelt wie die Teilwörter eines Bezeichners von einem Textelement. Die Phrase *the Account Management component* wird einem Bezeichner *AccountManagementComponent* gleichgesetzt. Dafür werden die Synonyme der einzelnen Wörter miteinander kombiniert und die Potenzmenge der dadurch generierten Begriffe gebildet. Das folgende Vorgehen unterscheidet sich nicht vom Vorgehen der Stufe zwei für einen Bezeichner der aus mehreren Teilwörtern besteht. Der einzige Unterschied besteht bei der Annotation im Graphen, wenn eine Verknüpfung für eine Phrase, statt einem Wort mit einem Modellelement hergestellt wird. Der Begriff *user* ist im Graphen als ein Synonym des Bezeichners *Account* gelistet. Die Kombination des Synonyms *user* mit *management* lässt auf eine Verknüpfung mit dem Modellelement *UserManagement* schließen. In diesem Fall wird die Phrase als neuer Knoten mit dem Knotentyp „phrase“ dem Graphen hinzugefügt. Statt zwischen einem „token“ und einem „model_element“ wird zwischen einer „phrase“ und einem „model_element“ eine Kante im Graphen erstellt.

Wird weder für die Wörter einer Phrase, noch für die Phrase selbst eine Verknüpfung zu einem Modellelement gefunden, wird die nächste Phrase oder der nächste Satz betrachtet. Ist das in diesem Abschnitt beschriebene Verfahren für alle Phrasen aller Sätze des Dokuments abgeschlossen, ist die Verarbeitung des Synonym-basierten Vergleichsansatzes durch den *WsdLink* abgeschlossen. Der Graph mit den gefundenen Verknüpfungen steht dem Agenten zur Weitergabe an andere Agenten zur Verfügung.

Ein wichtiger Aspekt, auf den in diesem Abschnitt bisher nicht eingegangen wurde, ist, mit welchem Verfahren die Ähnlichkeit zwischen den Bezeichnern der Textelemente und den Namen der Modellelemente bestimmt wird. Im Aktivitätsdiagramm in Abbildung 5.1 findet der Vergleich zwischen den Bezeichnern der Textelemente und den Namen der Modellelemente in den zwei Aktivitäten *Wort-Verknüpfung suchen* und *Phrase-Verknüpfung suchen* statt. Im System sind zwei Verfahren zur Bestimmung der Ähnlichkeit zwischen zwei Zeichenketten integriert. Zum einen ist das die Levenshtein-Distanz und zum anderen die Jaro-Winkler-Distanz. Diese Verfahren wurden in den Grundlagen in Abschnitt 2.5 eingeführt. Für beide Verfahren ist ein Schwellenwert definiert. Liegt die Ähnlichkeit zweier Zeichenketten unterhalb des Schwellenwertes, wird eine Verknüpfung erstellt. Sowohl das zu verwendende Verfahren, als auch der Schwellenwert können vom Anwender definiert werden.

Im Synonym-basierten Vergleichsansatz kann die Levenshtein-Distanz als eine der zwei Methoden zur Bestimmung der Ähnlichkeit zweier Zeichenketten und damit zur Erstellung von Verknüpfungen zwischen Text- und Modellelementen gewählt werden. Darüber hinaus wird ein Schwellenwert definiert. Liegt die Levenshtein-Distanz vom Bezeichner eines Textelements und dem Namen eines Modellelements unterhalb des Schwellenwertes, wird eine Verknüpfung erstellt. Für den Bezeichner *MediaAccess* und den Namen *MediumAccess* wird vom Agenten eine Verknüpfung für jeden Schwellenwert größer gleich zwei erstellt, da die Levenshtein-Distanz dieser Wörter zwei beträgt. Im Agenten ist der Standardwert für den Schwellenwert der Levenshtein-Distanz auf drei eingestellt. Für Wörter, deren Wortlänge kleiner als der doppelte Schwellenwert ist, in diesem Fall eine Wortlänge kleiner als sechs, wird der Schwellenwert auf die halbe Wortlänge reduziert. Damit wird verhindert, dass der Schwellenwert beim Vergleich von zwei kurzen Worten nicht zu hoch ist.

Werden beispielsweise zwei Wörter der Wortlänge drei verglichen, würde sonst immer eine Verknüpfung erstellt werden, da mit maximal drei Änderungen das eine Wort in das andere überführt werden kann. Der Standardschwellenwert von drei ist begründet auf der Idee, dass Wortendungen beim Vergleich keine Bedeutung zugewiesen werden soll. Für diese Entscheidung wurde die englische Wortendung *-ing* betrachtet. Diese besteht aus drei Buchstaben und somit ist der Schwellenwert auf drei gesetzt. Beispielsweise sind die Wörter *Reencoder* und *Reencoding* identisch, bis auf die Wortendungen *-er* und *-ing*, und haben eine Levenshtein-Distanz von drei. Ein Schwellenwert größer null erhöht damit die Anzahl korrekter Verknüpfungen aber auch die Anzahl falscher Verknüpfungen zwischen Text- und Modellelementen. Die Wahl des Schwellenwertes ist ein Kompromiss aus Ausbeute und Präzision. Je höher der Schwellenwert, desto mehr Verknüpfungen zwischen Text- und Modellelementen werden gefunden. Die Wahl eines guten Schwellenwertes ist entscheidend. Ob ein Schwellenwert von *3* geeignet ist, muss empirisch ermittelt werden.

Die andere Methode zur Bestimmung der Ähnlichkeit zweier Zeichenketten und damit zur Erstellung der Verknüpfungen des Synonym-basierten Vergleichsansatzes ist die Jaro-Winkler-Distanz. Der Schwellenwert, zur Erstellung der Verknüpfungen, ist mit einem Standardschwellenwert von *0,09* voreingestellt. Beispielsweise beträgt die Levenshtein-Distanz der zwei Wörter *media* und *medium 0,1*. Für diesen Wert wird keine Verknüpfung erstellt, da dieser oberhalb des Schwellenwertes liegt. Ist die Jaro-Winkler-Distanz, von einem Bezeichner der SAD und einem Namen des SAM, kleiner gleich dem Schwellenwert wird eine Verknüpfung erstellt. Eine Erhöhung des Schwellenwertes, beispielsweise von *0,09* auf *0,1*, hat eine hohe Zuordnung fälschlicherweise als positiv bestimmter Verknüpfungen zur Folge. Ein geeigneter Schwellenwert muss empirisch ermittelt werden.

6. Evaluation

In diesem Kapitel wird der Synonym-basierte Vergleichsalgorithmus evaluiert, indem er auf unterschiedliche Versionen einer SAD des Media Stores [23] angewandt wird. Die zugehörige Ontologie, die das SAM abbildet bleibt dabei konstant. Die Ergebnisse des Synonym-basierten Vergleichsansatzes sind abhängig von den Ergebnissen des verwendeten WSD-Algorithmus Babelfy. Aus diesem Grund wird in Abschnitt 6.1 der WSD-Algorithmus im Kontext der SAD unabhängig des Vergleichsansatzes evaluiert. In Abschnitt 6.2 wird der Synonym-basierte Vergleichsansatz unter Einbeziehung dieser Evaluationsergebnisse evaluiert. In Abschnitt 6.3 werden die Ergebnisse der gesamten Evaluation zusammengefasst und die Fragestellungen zum Erreichen der Ziele übergreifend beantwortet. Zum Schluss werden die Ergebnisse in Abschnitt 6.4 in Bezug auf deren Allgemeingültigkeit bewertet.

6.1. Evaluation von Babelfy

Die Idee zum Erstellen von Verknüpfungen zwischen inkonsistenten Bezeichnern und Namen von Text- und Modellelementen basiert auf der Idee, dass die Bedeutung der Elemente gleich ist. Aus diesem Grund werden mit Babelfy die Bedeutungen der Textelemente im Kontext der SAD in Form von Synsets bestimmt. Diese Synsets enthalten Synonyme, die für die Verknüpfung der Textelemente mit den Namen der Modellelementen verwendet werden. Die korrekte Bestimmung der Bedeutung der Textelemente im Kontext der SAD ist ausschlaggebend für das Abschneiden des Synonym-basierten Vergleichsansatzes. Wird bei inkonsistenter Benennung für ein Textelement die falsche Bedeutung bestimmt, ist es unwahrscheinlich, dass durch den Vergleichsansatz eine korrekte Verknüpfung erstellt werden kann.

Der WSD-Algorithmus Babelfy erreicht nach Raganato et al. [21] ein durchschnittliches F_1-Maß von 65,5% bei Evaluation verschiedener Datensätze. Abhängig des getesteten Datensatzes werden maximale F_1-Werte von 70,3% erreicht. Diese Datensätze bilden unterschiedliche Themenbereiche ab, wie Lern- und Nachrichtenartikel, Romanliteratur und biomedizinische und mathematische Berichte [21]. In diesem Abschnitt wird evaluiert, wie Babelfy im expliziten Kontext der SAD abschneidet. Für den Synonym-basierten Vergleichsansatz sind nur Textdokumente dieses Dokumententyps relevant. Der Babelfy-Algorithmus hat für 306 Wörter eine Bedeutung im Kontext eines Satzes bestimmt. Von diesen 306 Bedeutungen sind 126 als korrekt und 180 als inkorrekt im Kontext der SAD eingestuft. Damit ergibt sich eine Präzision von 41,2%, bei einer Ausbeute von 100%. Daraus folgt ein F_1-Maß von 58,3%. Dieses Ergebnis ist schlechter als die Babelfy-Ergebnisse der durchschnittlichen und maximal erreichten Vergleichswerte von Raganato et al. [21],

WSD-Ansatz	F_1-Maß in SAD	durchschnittliches F_1-Maß	maximales F_1-Maß
Babelfy	58,3%	65,5%	70,3%

Tabelle 6.1.: Babelfy-Ergebnisse im Kontext der SAD im Vergleich zu den durchschnittlich und maximal erreichten Babelfy-Ergebnissen von Raganato et al. [21].

dargestellt in Tabelle 6.1. Im Anhang in Tabelle A.1 sind die detaillierten Ergebnisse der Evaluation gelistet.

Auf Basis dieser Ergebnisse des Babelfy-Algorithmus im Kontext der SAD kann zu 41,2% die Bedeutung eines Bezeichners im Kontext des Satzes korrekt bestimmt werden. Daraus folgt, dass in dieser SAD zu 41,2% die richtigen Synonyme zur Erstellung der Verknüpfungen verwendet werden, zur Auflösung inkonsistenter Benennungen zwischen Text- und Modellelementen. In den anderen 58,8% sind die Bedeutungen falsch bestimmt und eine Auflösung der Inkonsistenzen mit den Wörtern der Synsets ist nicht zielführend. Mit einer Ausbeute von 100% wird für jedes Wort für das eine Bedeutung bestimmt werden kann, mit Babelfy auch ein Ergebnis zurückgeliefert. Dieses Verhalten beruht teilweise auf der Backup-Strategie MCS. Wird von Babelfy keine Bedeutung für ein Wort bestimmt, wird der MCS für dieses Wort zurückgegeben. Diese Ergebnisse müssen bei Interpretation der Evaluationsergebnisse des Synonym-basierten Vergleichsansatzes bedacht werden.

6.2. Evaluation des Ansatzes

In diesem Abschnitt wird der Synonym-basierter Vergleichsansatz zum Erstellen von Verknüpfungen zwischen Text- und Modellelementen evaluiert. Zum Erreichen der in Abschnitt 4.3 definierten Ziele, werden mit dieser Evaluation die folgenden Fragestellungen unter Messung verschiedener Metriken beantwortet:

F1: Wie hoch ist die Ausbeute und wie präzise können Verknüpfungen zwischen den Bezeichnern der Textelemente der SAD und den Namen der Modellelemente des SAM erkannt werden, unter Anwendung des Synonym-basierten Vergleichsansatzes?

F2: Wie gut ist die Verknüpfung zwischen den Bezeichnern der Textelemente der SAD und den Namen der Modellelemente des SAM, im Vergleich zu einem naiven Vergleichsansatz ohne Einbeziehung von Synonymen?

F3: Wie gut ist die Levenshtein-Distanz im Vergleich zur Jaro-Winkler-Distanz, zur Bestimmung der Ähnlichkeit zwischen den Bezeichnern der Textelemente der SAD und den Namen der Modellelemente des SAM und welche Schwellenwerte sind geeignet?

F4: Wie stark wird die Laufzeit des Synonym-basierten Vergleichsansatzes durch die Anzahl Synonyme pro Wort beeinflusst?

Zur Beantwortung der Fragestellungen und dem damit einhergehenden Erreichen der Ziele, werden die Information-Retrieval-Metriken Ausbeute und Präzision, beziehungsweise deren Kombination in Form des F_1-Maß und des $F_{\sqrt{2}}$-Maß betrachtet. Ausbeute und

Präzision können Werte von 0 bis 1 annehmen. Ein Ausbeute-Wert von 1 bedeutet, dass alle korrekten Verknüpfungen (True Positive, TP) zwischen SAD und SAM gefunden werden. Unter den gefunden Verknüpfungen können Verknüpfungen sein, die nicht korrekt (False Positive, FP) sind. Ein Präzisions-Wert von 1 bedeutet, dass alle gefundenen Verknüpfungen korrekt sind. Es kann jedoch sein, dass nicht alle korrekten Verknüpfungen gefunden wurden und fälschlicherweise als negativ (False Negative, FN) bestimmt werden[2]. Die Formeln zur Berechnung von Ausbeute und Präzision sind folgende:

$$\text{Ausbeute} = \frac{\sum TP}{\sum (TP+FN)} \qquad \text{Präzision} = \frac{\sum TP}{\sum (TP+FP)}$$

Die Ausbeute berechnet sich aus der Anzahl korrekt bestimmter Verknüpfungen, die durch die Summe der Anzahl korrekt bestimmter Verknüpfungen und nicht gefundenen Verknüpfungen geteilt wird. Werden alle existierenden Verknüpfungen gefunden, ergibt sich der Ausbeute-Wert von 1. Die Präzision berechnet sich aus der Anzahl korrekt bestimmter Verknüpfungen, die durch die Summe der Anzahl korrekt bestimmter Verknüpfungen und fälschlicherweise als positiv bestimmten Verknüpfungen geteilt wird. Werden keine falschen Verknüpfungen bestimmt, ergibt sich der Präzisions-Wert von 1. Das F_1-Maß ist das harmonische Mittel aus Ausbeute und Präzision. Im Gegensatz dazu wird beim $F_{\sqrt{2}}$-Maß die Ausbeute zweimal höher gewichtet als die Präzision. Die Ausbeute wird höher gewertet als die Präzision, da zum Auffinden von Inkonsistenzen möglichst alle existierenden Verknüpfungen auch erstellt werden müssen. Die allgemeine Formel zur Berechnung des F_β-Maßes ist folgende:

$$F_\beta\text{-Maß} = (1+\beta^2) * \frac{Präzision * Ausbeute}{(\beta^2 * Präzision) + Ausbeute}$$

Daraus ergeben sich die Formeln zur Berechnung von F_1-Maß und $F_{\sqrt{2}}$-Maß:

$$F_1\text{-Maß} = 2 * \frac{Präzision * Ausbeute}{Präzision + Ausbeute} \qquad F_{\sqrt{2}}\text{-Maß} = (1+\sqrt{2}^2) * \frac{Präzision * Ausbeute}{(\sqrt{2}^2 * Präzision) + Ausbeute}$$

Zur Ermittlung der Metriken wird der Synonym-basierte Vergleichsansatz auf eine Fallstudie angewandt. Dafür wird ein Softwaresystem evaluiert, für welches mehrere Versionen einer SAD im Textformat und ein SAM in Ontologieform vorliegt. Die durch den Vergleichsansatz bestimmten Verknüpfungen zwischen Text- und Modellelementen werden mit den im Text existierenden Verknüpfungen verglichen. Die Vergleichswerte zur Bewertung des Synonym-basierten Vergleichsansatzes werden von Hand aus dem Textdokument extrahiert. Die von Hand erstellten Verknüpfungen entsprechen dem Goldstandard. Es werden Ausbeute, Präzision, F_1-Maß und $F_{\sqrt{2}}$-Maß für die Trefferquote bei der Zuordnung zwischen den Bezeichnern der Textelemente und den Namen der Modellelemente bestimmt. Die Levenshtein- und Jaro-Winkler-Distanz werden zur Bestimmung der Ähnlichkeit zwischen den Bezeichnern der Textelemente und den Namen der Modellelemente eingesetzt. Der naive Vergleichsansatz entspricht einer Anwendung des Synonym-basierten Vergleichsansatzes mit null Synonymen pro Wort zur Erstellung der Verknüpfungen. Dabei werden folgende Konfigurationen gemessen und miteinander, beziehungsweise mit dem optimalen Ergebnis des Goldstandards verglichen:

Version	Bearbeitungsstand der SAD	Anzahl Elemente
Version 1	Media Store unbearbeitet	52
Version 2	Media Store bearbeitet ohne Bias	41
Version 3	Media Store bearbeitet mit Bias	49

Tabelle 6.2.: Überblick der verschiedenen Versionen des Media Stores die evaluiert werden

M1: Ergebnis des naiven Vergleichsansatzes (Synonym-basierter Vergleichsansatz mit null Synonymen pro Wort) und der Levenshtein-Distanz in Form des F_1-Maß und $F_{\sqrt{2}}$-Maß.

M2: Ergebnis des naiven Vergleichsansatzes (Synonym-basierter Vergleichsansatz mit null Synonymen pro Wort) und der Jaro-Winkler-Distanz in Form des F_1-Maß und $F_{\sqrt{2}}$-Maß.

M3: Ergebnis des Synonym-basierten Vergleichsansatzes und der Levenshtein-Distanz für ein bis zehn Synonyme pro Wort in Form des F_1-Maß und $F_{\sqrt{2}}$-Maß.

M4: Ergebnis des Synonym-basierten Vergleichsansatzes und der Jaro-Winkler-Distanz für ein bis zehn Synonyme pro Wort in Form des F_1-Maß und $F_{\sqrt{2}}$-Maß.

In den folgenden Unterabschnitten werden drei verschiedene Versionen der SAD des Media Stores [23] evaluiert. Für alle drei Versionen wird dasselbe zugrundeliegende SAM in Form einer Ontologie verwendet, zur Extraktion der Modellelemente. Es stehen 14 Namen von Modellelementen zur Verfügung, die für eine Verknüpfung mit den Textelementen betrachtet werden. Die Schwellenwerte zur Bestimmung der Ähnlichkeit zwischen den Bezeichnern der Textelemente und Namen der Modellelemente, entsprechen den Standardschwellenwerten des jeweiligen Verfahrens. Für die Levenshtein-Distanz ist dieser Wert auf *3* festgelegt und für die Jaro-Winkler-Distanz auf *0.09* (siehe Unterabschnitt 2.5.1 und Unterabschnitt 2.5.2). In Tabelle 6.2 ist dargestellt, welche Versionen der Media Stores evaluiert werden und für wie viele Textelemente eine Verknüpfung zu einem Modellelement existiert. In Unterabschnitt 6.2.1 wird der Synonym-basierte Vergleichsansatz auf Version 1 des Media Stores angewandt, dies entspricht der offiziellen Version der SAD des Media Stores. In Unterabschnitt 6.2.2 wird der Synonym-basierte Vergleichsansatz auf eine bearbeitete Version der SAD des Media Stores angewandt, dies entspricht Version 2. Die Bearbeitung dieser Version wurde von einer unabhängigen Person durchgeführt und enthält keinen Bias. In Unterabschnitt 6.2.3 wird der Synonym-basierte Vergleichsansatz auf eine weitere bearbeitete Version der SAD des Media Stores angewandt, dies entspricht Version 3. Diese Version wurde selbst bearbeitet und enthält einen Bias.

6.2.1. Media Store Version 1

Zur Beantwortung der Fragestellungen wird der Ansatz auf die offizielle Version der SAD des Media Stores angewandt. Diese Version enthält 600 Wörter, wovon für 52 Textelemente eine mögliche Verknüpfung mit einem Modellelement existiert. Die durch WSD-Algorithmus generierten Synonyme des Ansatzes können keinen positiven Einfluss auf

Verfahren	Synonyme /Wort	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
Levenshtein	0	65,4%	100%	79,1%	73,9%
	2	69,2%	94,7%	**80,0%**	**76,1%**
	5	69,2%	78,3%	73,5%	72,0%
Jaro-Winkler	0	53,9%	100%	70,0%	63,6%
	2	59,6%	100%	**74,7%**	**68,9%**
	5	59,6%	88,6%	71,3%	66,9%

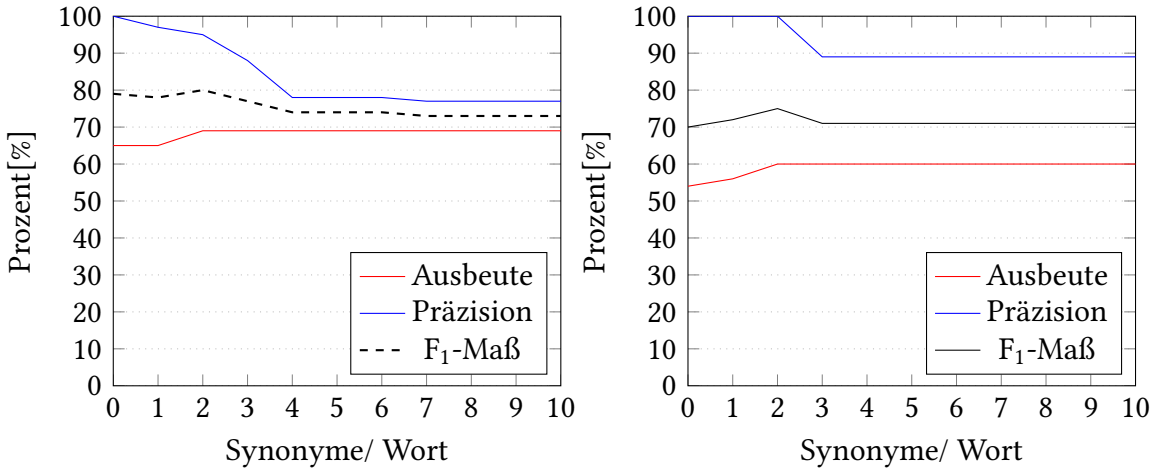
Tabelle 6.3.: Übersicht der Evaluationsergebnisse des Ansatzes für Version 1 des Media Stores

Erstellung von Verknüpfungen zwischen Text- und Modellelemente ausüben, wenn keine Inkonsistenzen bei der Benennung vorliegen. In dieser offiziellen Version der SAD existieren Inkonsistenzen bei der Benennung zwischen SAD und SAM.

Im ersten Teil wird evaluiert, wie gut Verknüpfungen zwischen den Text- und Modellelementen bei einem naiven Vergleichsansatz ohne Einbeziehung von Synonymen erstellt werden können. Dafür wird die Anzahl an Synonymen die pro Wort im Graphen annotiert werden, auf den Wert null gesetzt. In Tabelle 6.3 in der ersten Zeile des jeweiligen Verfahrens ist zu erkennen, wie hoch Ausbeute und Präzision und das damit einhergehende F_1-Maß und $F_{\sqrt{2}}$-Maß für den naiven Vergleichsansatz ausfallen. Bei Verwendung der Levenshtein-Distanz können 65,4% der Textelemente, für die eine Verknüpfung mit einem Modellelement existiert, korrekt zugeordnet werden. Damit beträgt das F_1-Maß 79,1%. Im Vergleich dazu fällt die Zuordnungsrate bei der Jaro-Winkler-Distanz mit 53,9% schlechter aus, woraus sich ein F_1-Maß von 70,0% ergibt. Ohne Anwendung von Synonymen schneidet die Levenshtein-Distanz besser ab als die Jaro-Winkler-Distanz. Im Anhang in Tabelle A.2 und Tabelle A.5 sind die detaillierten Ergebnisse dieser Evaluation gelistet.

Bei Erhöhung der Anzahl Synonyme pro Wort von null auf zwei (Spalte zwei in Tabelle 6.3 für das jeweilige Verfahren) wird die Ausbeute bei der Levenshtein-Distanz von 65,4% auf 69,2% erhöht. Auch bei der Jaro-Winkler-Distanz erhöht sich für zwei Synonyme pro Wort die Ausbeute von 53,9% auf 59,6%. Die Ausbeute der Levenshtein-Distanz liegt knapp 10 Prozentpunkte über der Ausbeute der Jaro-Winkler-Distanz. Mit dem Synonym-basierten Vergleichsansatz unter Verwendung der Levenshtein-Distanz können mehr korrekte Verknüpfungen erstellt werden als mit der Jaro-Winkler-Distanz. Gleichzeitig sinkt die Präzision mit der Anzahl Synonyme pro Wort, bis sie für drei bis vier Synonyme pro Wort einen konstanten Wert annimmt. Das Sinken der Präzision mit der Anzahl Synonyme pro Wort beruht darauf, dass es wahrscheinlicher wird ein Synonym zu finden welches ähnlich zu dem Namen eines Modellelements ist, ohne semantisch dieses Element abzubilden. Unter Anwendung des Ansatzes mit der Jaro-Winkler-Distanz fällt die Präzision besser aus. Es werden weniger falsche Verknüpfungen erstellt, als mit der Levenshtein-Distanz. Dies wird auch von den F_1-Werten widergespiegelt.

In Abbildung 6.1 ist dargestellt, wie sich Ausbeute, Präzision und das F_1-Maß verhalten, abhängig der Anzahl Synonyme pro Wort. Die Werte sind dargestellt für null bis zehn Synonyme pro Wort. Es lässt sich erkennen, dass die Ausbeute mit der Levenshtein-Distanz und die Präzision mit der Jaro-Winkler-Distanz besser ausfällt. Für mehr als zwei bis drei



(a) Levenshtein-Distanz

(b) Jaro-Winkler-Distanz

Abbildung 6.1.: Evaluation der SAD von Version 1 des Media Stores, abhängig von der Anzahl Synonyme pro Wort

Synonyme pro Wort ist keine Erhöhung der Ausbeute feststellbar. Gleichzeitig nimmt die Präzision im Bereich von null bis vier Synonyme pro Wort für beide Verfahren am meisten ab. Die Laufzeit des Ansatzes ist dabei bei einer Erhöhung von zwei auf zehn Synonyme pro Wort von einer Minute auf das dreifache angestiegen[1].

In Abbildung 6.2 sind die expliziten F_1-Werte der beiden Verfahren im Vergleich dargestellt. Die Levenshtein-Distanz erreicht mit höherer Ausbeute bei geringerer Präzision übergreifend bessere F_1-Werte, als die Jaro-Winkler-Distanz.

Für diese Version der SAD lässt sich zusammenfassend festhalten, dass die Anwendung des Synonym-basierten Vergleichsansatzes für zwei Synonymen pro Wort das beste F_1-Maß liefert. Für diesen Wert wird eine durchschnittliche Ausbeute von 64,4% bei einer Präzision von 97,4% erreicht. Das Erstellen von Verknüpfungen zwischen Text- und Modellelementen verbessert sich, bei Anwendung des Synonym-basierten Vergleichsansatzes, gegenüber Anwendung des Vergleichsansatzes ohne Synonyme. Für diese Version der SAD schneidet die Levenshtein-Distanz im Kontext der SAD besser ab als die Jaro-Winkler-Distanz. Die Laufzeit des Vergleichsansatzes wächst linear mit der Anzahl Synonyme für null bis zehn Synonyme pro Wort von einer auf drei Minuten an.

6.2.2. Media Store Version 2

In der betrachteten SAD des vorherigen Abschnitts ist die Anzahl möglicher Textelemente, für die eine Verknüpfung mit den Modellelementen hergestellt werden kann begrenzt. Aus diesem Grund wird in diesem Unterabschnitt eine bearbeitete Version der SAD des Media Stores evaluiert, bei der mehr Inkonsistenzen bei der Benennung der Bezeichner vorliegen. Die Bearbeitung des Textdokuments der SAD hat eine Person vorgenommen, die nicht über die detaillierte Vorgehensweise des Ansatzes zur Erstellung der Verknüpfungen

[1]Single-Threaded, bei Ausführung mit einem AMD Ryzen 5 2600 Prozessor

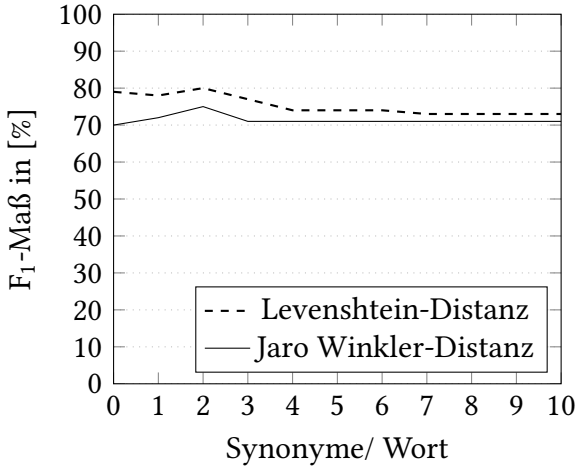


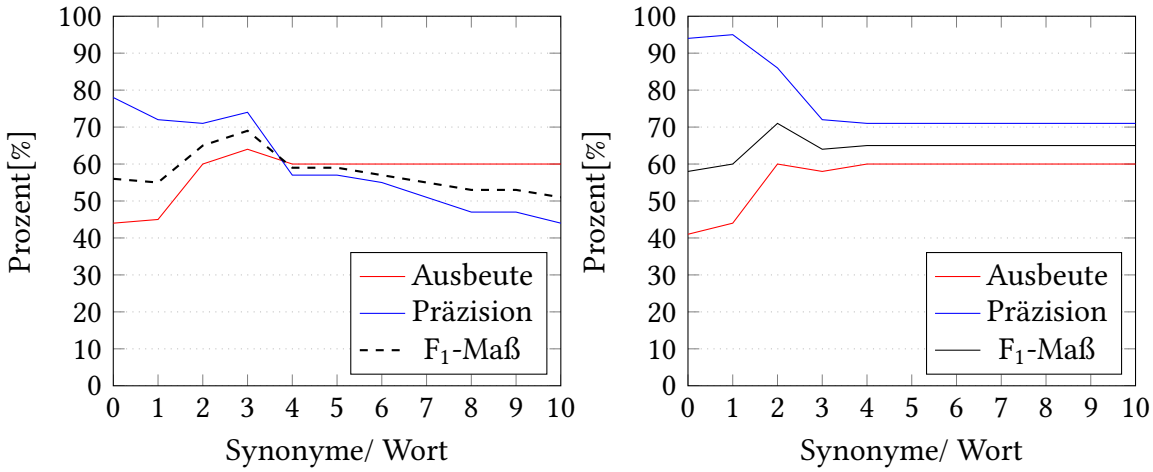
Abbildung 6.2.: Vergleich der F_1-Werte von Version 1 des Media Stores, abhängig Methoden zur Bestimmung der Ähnlichkeit von zwei Zeichenketten

Verfahren	Synonyme /Wort	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
Levenshtein	0	43,9%	78,3%	56,3%	51,4%
	2	60,0%	70,1%	**64,9%**	**63,2%**
	5	60,0%	57,1%	58,5%	59,0%
Jaro-Winkler	0	41,5%	94,4%	57,6%	51,0%
	2	60,0%	85,7%	**70,6%**	**66,7%**
	5	60,0%	70,6%	64,9%	63,2%

Tabelle 6.4.: Übersicht der Evaluationsergebnisse des Ansatzes für Version 2 des Media Stores

Bescheid weiß, um einen Bias zu vermeiden. Das Dokument umfasst 700 Wörter und enthält 41 Textelemente, für die eine mögliche Verknüpfung mit einem Modellelement existiert.

In Tabelle 6.4 ist dargestellt, wie sich die Levenshtein-Distanz gegenüber der Jaro-Winkler-Distanz für unterschiedliche Anzahlen an Synonymen pro Wort verhält. Für die Levenshtein-Distanz steigt die Ausbeute von 43,9% auf 60,0%, für eine Erhöhung der Anzahl an Synonymen pro Wort von null auf zwei. Für die Jaro-Winkler-Distanz steigt die Ausbeute von 41,5% auf 60,0%, für das gleiche Vorgehen. Die Präzision der Levenshtein-Distanz ist mit 70,1%, für zwei Synonyme pro Wort schlechter als die Präzision der Jaro-Winkler-Distanz mit 85,7%. Daraus resultiert ein besserer F_1-Wert für die Jaro-Winkler-Distanz mit 70,6% als für die Levenshtein-Distanz mit 64,9%. Für dieses Dokument schneidet die Jaro-Winkler-Distanz besser ab als die Levenshtein-Distanz, statt wie in Version 1 bei der die Levenshtein-Distanz besser abschneidet. Die Anzahl Verknüpfungen die zwischen den Text- und Modellelementen erstellt werden können unterscheidet sich für beide Verfahren kaum, wobei die Jaro-Winkler-Distanz mit einer höheren Präzision die Verknüpfungen bestimmen kann. Im Anhang in Tabelle A.3 und Tabelle A.6 sind die detaillierten Ergebnisse dieser Evaluation gelistet.



(a) Levenshtein-Distanz

(b) Jaro-Winkler-Distanz

Abbildung 6.3.: Evaluation der SAD von Version 2 des Media Stores, abhängig von der Anzahl Synonyme pro Wort

In Abbildung 6.3 sind zwei Diagramme, welche die Entwicklung von Ausbeute, Präzision und F_1-Maß, abhängig der Anzahl Synonyme pro Wort, darstellen. Es ist erkennbar, dass die Präzision der Levenshtein-Distanz mit der Anzahl Synonyme pro Wort abnimmt. Während die Präzision der Jaro-Winkler-Distanz konstant bleibt, wenn die Ausbeute einen konstanten Wert annimmt. Für mehr als drei bis vier Synonyme pro Wort ist keine Erhöhung der Ausbeute feststellbar. Bei Erhöhung der Anzahl Synonyme pro Wort von zwei auf zehn, hat sich die Laufzeit des Ansatzes mit der Levenshtein-Distanz von einer Minute auf das zehnfache erhöht[2]. Für die Jaro-Winkler-Distanz fällt die Laufzeit für zehn Synonyme pro Wort schlechter aus.

In Abbildung 6.4 ist der explizite Vergleich des Ansatzes mit der Levenshtein-Distanz gegenüber der Jaro-Winkler-Distanz mit dem F_1-Maß dargestellt. Für drei Synonyme pro Wort ist die Levenshtein-Distanz besser. In allen anderen Fällen schneidet die Jaro-Winkler-Distanz besser ab und hält für eine hohe Anzahl Synonyme pro Wort ein konstantes F_1-Maß bei. Ab sieben Synonymen pro Wort liegt das F_1-Maß des Synonym-basierten Vergleichsansatzes mit der Levenshtein-Distanz unterhalb dem F_1-Maß des naiven Vergleichsansatz mit null Synonymen pro Wort. Damit schneidet der Synonym-basierte Vergleichsansatz schlechter ab als der naive Vergleichsansatz ohne Synonyme, wenn die Anzahl Synonyme pro Wort zu hoch angesetzt ist. Dieses Verhalten beruht auf einer konstanten Ausbeute und gleichzeitig abnehmender Präzision, je mehr Synonyme pro Wort annotiert werden.

Für diese Version der SAD lässt sich zusammenfassend festhalten, dass bei Anwendung des Synonym-basierten Vergleichsansatzes mit zwei bis drei Synonymen pro Wort das F_1-Maß maximal wird. Es wird für diese Werte eine durchschnittliche Ausbeute von 60,4% und Präzision von 75,3% erreicht. Das Erstellen von Verknüpfungen zwischen Text- und Modellelementen verbessert sich, Synonym-basierten Vergleichsansatzes gegenüber Anwendung des naiven Vergleichsansatzes ohne Synonyme, solange die Anzahl Synonyme

[2]Single-Threaded, bei Ausführung mit einem AMD Ryzen 5 2600 Prozessor

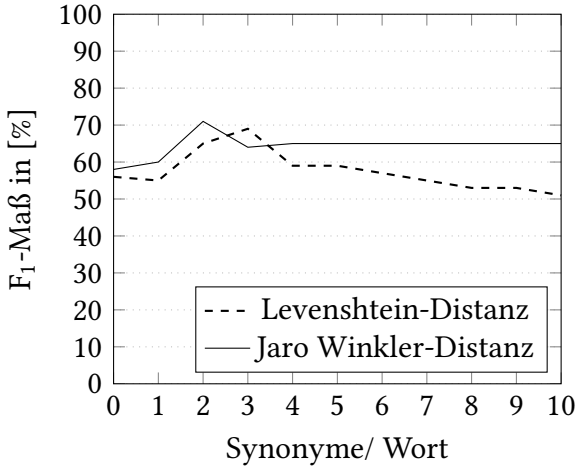


Abbildung 6.4.: Vergleich der F_1-Werte von Version 2 des Media Stores, abhängig der Methoden zur Bestimmung der Ähnlichkeit von zwei Zeichenketten

pro Wort kleiner sieben ist. Für zwei Synonyme pro Wort schneidet der Synonym-basierte Vergleichsansatz mit der Jaro-Winkler-Distanz genauso gut, wie für drei Synonyme pro Wort mit der Levenshtein-Distanz ab. Je höher die Anzahl an Synonymen pro Wort bestimmt wird, desto schlechter fällt das Ergebnis der Levenshtein-Distanz aus, während das Ergebnis der Jaro-Winkler-Distanz konstant bleibt. Die Laufzeit der beiden Verfahren steigt für null bis acht Synonyme linear von 70 auf 300 Sekunden an. Für neun und zehn Synonyme ist eine starke Erhöhung der Laufzeit zu verzeichnen, die nicht linear verläuft.

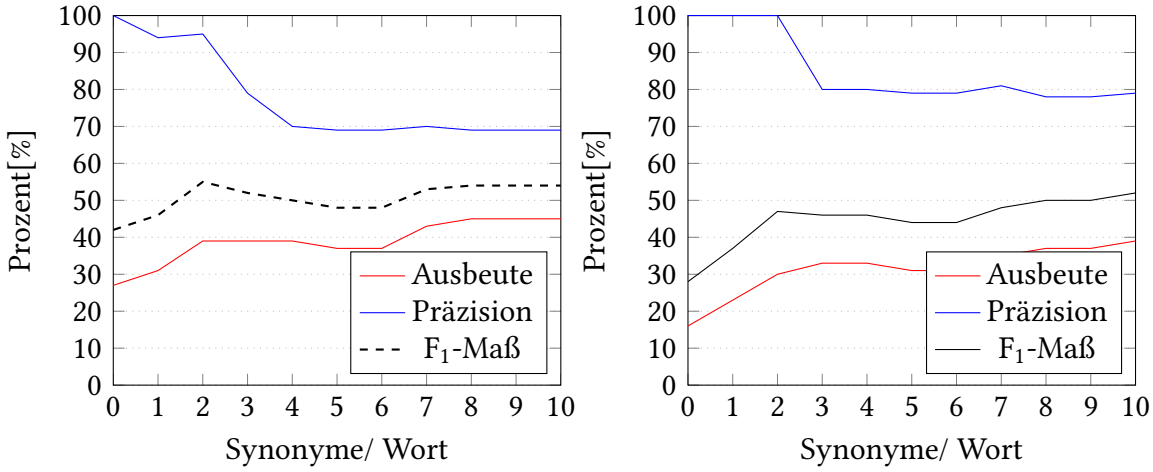
6.2.3. Media Store Version 3

Die in diesem Abschnitt betrachtete Version der SAD des Media Stores ist eine bearbeitete Version der SAD aus Unterabschnitt 6.2.1. Diese Version wurde selbst bearbeitet und enthält damit einen Bias. Die Bearbeitung, beziehungsweise Umbenennung der Textelemente basiert auf den Synonymen der Bezeichner. Das Ziel bei Evaluation dieser Version ist herauszufinden, wie gut der Ansatz abschneidet, wenn die Annahmen bei der Umbenennung eingehalten werden. Damit wird evaluiert, wie der Ansatz abschneidet wenn die Bedingung gilt, dass die Bedeutung eines Bezeichners bei Umbenennung nicht verändert wird. Die Bezeichner der Textelemente sind in dieser Version Synonyme der Namen der Modellelemente. Das Dokument umfasst 600 Wörter, wovon für 49 Textelemente eine mögliche Verknüpfung mit einem Modellelement existiert.

In Tabelle 6.5 ist dargestellt, wie Ausbeute, Präzision und das F_1- und $F_{\sqrt{2}}$-Maß ausfallen, abhängig des verwendeten Verfahrens zur Bestimmung der Ähnlichkeit. Für den naiven Vergleichsansatz ohne Einbeziehung von Synonymen bei der Erstellung von Verknüpfungen kann bei Anwendung mit der Levenshtein-Distanz eine Ausbeute von 26,5% und mit der Jaro-Winkler-Distanz eine geringere Ausbeute von 16,3% erreicht werden. Bei Erhöhung der Anzahl Synonyme pro Wort auf zwei bei Verwendung des Synonym-basierten Vergleichsansatzes kann die Ausbeute mit der Levenshtein-Distanz um mehr als 12 Prozentpunkte auf 38,8% erhöht werden. Unter Einbüßen von 5% der Präzision

Verfahren	Synonyme /Wort	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
Levenshtein	0	26,5%	100%	41,9%	35,1%
	2	38,8%	95,0%	**55,1%**	**48,3%**
	5	36,7%	69,2%	48,0%	43,6%
Jaro-Winkler	0	16,3%	100%	28,1%	22,6%
	2	30,6%	100%	**46,9%**	**39,8%**
	5	30,6%	79,0%	44,1%	38,5%

Tabelle 6.5.: Übersicht der Evaluationsergebnisse des Ansatzes für Version 3 des Media Stores



(a) Levenshtein-Distanz

(b) Jaro-Winkler-Distanz

Abbildung 6.5.: Evaluation der SAD von Version 3 des Media Stores, abhängig von der Anzahl Synonyme pro Wort

ergibt das ein F_1-Maß von 55,1%. Mit der Jaro-Winkler-Distanz kann die Ausbeute bei zwei Synonymen pro Wort um mehr als 14 Prozentpunkte auf 30,6% erhöht werden, ohne Abnahme der Präzision. Damit wird mit 46,9% ein über 8 Prozentpunkte schlechteres F_1-Maß erreicht als mit der Levenshtein-Distanz. Im Anhang in Tabelle A.4 und Tabelle A.7 sind die detaillierten Ergebnisse dieser Evaluation gelistet.

Beim Vergleich der Diagramme in Abbildung 6.5 ist zu erkennen, dass die Ausbeute mit steigender Anzahl an Synonymen pro Wort für den Synonym-basierten Vergleichsansatz steigt. Ab drei bis vier Synonyme pro Wort ist gleichzeitig keine Abnahme der Präzision feststellbar. Für eine Erhöhung von null auf zwei bis drei Synonyme pro Wort ist der Anstieg der Ausbeute und des F_1-Maßes am steilsten. Gleichzeit ist der Abstieg der Präzision für zwei bis vier Synonyme pro Wort am steilsten und danach konstant. Dadurch ist ein lokales Maximum für das F_1-Maß bei zwei Synonymen pro Wort zu verzeichnen. Die Präzision der Jaro-Winkler-Distanz ist um circa 10 Prozentpunkte besser als die der Levenshtein-Distanz.

In Abbildung 6.6 ist zu sehen, dass das F_1-Maß der Levenshtein-Distanz, unabhängig der Anzahl Synonyme pro Wort, oberhalb dem F_1-Maß der Jaro-Winkler-Distanz liegt.

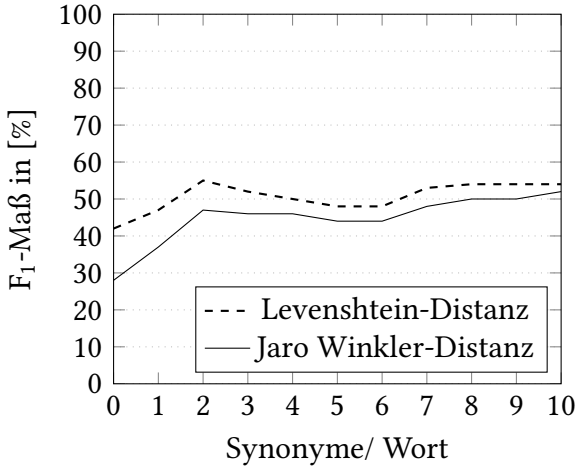


Abbildung 6.6.: Vergleich der F_1-Werte von Version 3 des Media Stores, abhängig Methoden zur Bestimmung der Ähnlichkeit von zwei Zeichenketten

Für diese Version der SAD lässt sich festhalten, dass bei Anwendung des Synonym-basierten Vergleichsansatzes mit zwei oder acht bis zehn Synonymen pro Wort das F_1-Maß maximal ist. Es wird für diese Werte eine durchschnittliche Ausbeute von 36,8% und Präzision von 89,5% erreicht. Das Erstellen der Verknüpfungen zwischen Text- und Modellelementen verbessert sich bei Anwendung des Synonym-basierten Vergleichsansatzes gegenüber Anwendung des naiven Vergleichsansatzes ohne Synonyme. Je höher die Anzahl Synonyme pro Wort bestimmt wird, desto höher fällt die Ausbeute aus. Dies ist unabhängig von den verwendeten Verfahren für die Bestimmung der Ähnlichkeit von Zeichenketten. Unabhängig der Anzahl Synonyme pro Wort schneidet der Ansatz mit der Levenshtein-Distanz besser ab, als mit der Jaro-Winkler-Distanz. Die Präzision der Jaro-Winkler-Distanz ist besser als die Präzision der Levenshtein-Distanz. Die Laufzeit steigt linear von knapp einer Minute auf etwas über drei Minuten an, bei einer Erhöhung von null auf zehn Synonyme pro Wort.

6.3. Zusammenfassung der Ergebnisse

Die verschiedenen Versionen des Media Stores führen bei der Evaluation des Synonym-basierten Vergleichsansatzes zu unterschiedlichen Ergebnissen. In diesem Abschnitt werden diese Ergebnisse zusammengefasst und übergreifend interpretiert. Mit den Messungen der Evaluation lassen sich die zuvor definierten Fragestellungen zusammenfassend beantworten:

F1: Wie hoch ist die Ausbeute und wie präzise können Verknüpfungen zwischen den Bezeichnern der Textelemente der SAD und den Namen der Modellelemente des SAM erkannt werden unter Anwendung des Synonym-basierten Vergleichsansatzes? In Tabelle 6.6 sind die durchschnittlichen Ergebnisse des Ansatzes für zwei Synonyme pro Wort, abhängig des gewählten Vergleichsverfahrens dargestellt. Mit der Levenshtein-Distanz ist

Verfahren	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
Levenshtein	56,0%	86,6%	66,7%	62,5%
Jaro-Winkler	50,1%	95,2%	64,1%	58,5%

Tabelle 6.6.: Durchschnittliche Ergebnisse des Ansatzes für zwei Synonyme pro Wort

Verfahren	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
Levenshtein	+10,7	-6,2	+7,6	+9,1
Jaro-Winkler	+12,8	-2,9	+12,1	+12,7

Tabelle 6.7.: Durchschnittliche absolute Änderung der Ergebnisse des Synonym-basierten Vergleichsansatzes für zwei Synonyme pro Wort gegenüber null Synonyme pro Wort (in Prozentpunkten)

eine durchschnittliche Ausbeute von 56,0% möglich, bei einer Präzision von 86,6%. Das bedeutet, dass 56% aller möglichen Verknüpfungen zwischen Text- und Modellelementen erkannt werden und, dass 86,6% dieser erkannten Verknüpfungen korrekt sind. Mit der Jaro-Winkler-Distanz ist eine durchschnittliche Ausbeute von 50,1% möglich, bei einer Präzision von 95,2%. Damit werden die Hälfte aller möglichen Verknüpfungen zwischen Text- und Modellelementen erkannt, wobei die erkannten Verknüpfungen zu 95,2% korrekt sind. Die Wahl von zwei bis drei Synonymen pro Wort liefert durchschnittlich die besten Ergebnisse für beide Verfahren, bei Anwendung des Synonym-basierten Vergleichsansatzes.

F2: Wie gut ist die Verknüpfung zwischen den Bezeichnern der Textelemente der SAD und den Namen der Modellelemente des SAM, im Vergleich zu einem naiven Vergleichsansatz ohne Einbeziehung von Synonymen? In Tabelle 6.7 ist die durchschnittliche absolute Änderung der Ergebnisse des Synonym-basierten Vergleichsansatzes in Prozentpunkten angegeben, bei einer Erhöhung der Anzahl an Synonymen pro Wort von null auf zwei. Die dargestellten Zahlen bilden die absolute Änderung gegenüber dem naiven Vergleichsansatz ohne Synonyme ab. Es ist zu erkennen, dass die Ausbeute mit dem Synonym-basierten Vergleichsansatz für beide Verfahren gegenüber dem naiven Vergleichsansatz um 10 Prozentpunkte erhöht werden kann. Damit können Verknüpfungen aufgelöst werden, für die eine inkonsistente Benennung zwischen SAD und SAM vorliegt. Die Anwendung des Synonym-basierten Vergleichsansatzes für das Erstellen von Verknüpfung zwischen Text- und Modellelementen liefert bessere Ergebnisse als der naive Vergleichsansatz ohne Synonyme.

F3: Wie gut ist die Levenshtein-Distanz im Vergleich zur Jaro-Winkler-Distanz, zur Bestimmung der Ähnlichkeit zwischen den Bezeichnern der Textelemente der SAD und den Namen der Modellelemente des SAM und welche Schwellenwerte sind geeignet? In Tabelle 6.8 ist zu erkennen, dass die Levenshtein-Distanz eine höhere Ausbeute erreicht als die Jaro-Winkler-Distanz. Für den naiven Vergleichsansatz erreicht die Levenshtein-Distanz eine 8 Prozentpunkte höhere Ausbeute, als die Jaro-Winkler-Distanz. Mit dem Synonym-basierten Vergleichsansatz und zwei Synonymen pro Wort ist die Ausbeute der

Bedeutungen/ Wort	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
0	+8	-5,4	+7,2	+7,7
2	+5,9	-6,8	+2,6	+4,1

Tabelle 6.8.: Absoluter Vergleich der Ergebnisse der Levenshtein-Distanz mit denen der Jaro-Winkler-Distanz (in Prozentpunkten)

Dokument	Mögliche Verknüpfungen	Schwellenwert	TP	FP
Media Store Version 1	52	2	31	0
		3	36	5
		4	35	10
Media Store Version 3	49	2	13	0
		3	19	5
		4	17	10

Tabelle 6.9.: Menge der richtigerweise (TP) und fälschlicherweise (FP) als positiv bestimmten Verknüpfungen abhängig des Schwellenwertes der Levenshtein-Distanz.

Levenshtein-Distanz um 5,9 Prozentpunkte besser. In Bezug auf die Präzision schneidet die Jaro-Winkler-Distanz besser ab als die Levenshtein-Distanz. Bei Anwendung des naiven Ansatzes ist diese bei der Jaro-Winkler-Distanz um 5,4 Prozentpunkte besser und bei zwei Synonyme pro Wort um 6,8 Prozentpunkte besser. Damit lässt sich festhalten, dass die Levenshtein-Distanz besser geeignet ist, wenn die Ausbeute im Vordergrund steht und maximiert werden soll. Liegt der Fokus auf der Präzision, ist die Jaro-Winkler-Distanz unter Einbüßen der Ausbeute die bessere Wahl.

Diese Ergebnisse sind stark vom gewählten Schwellenwert des jeweiligen Verfahrens abhängig. Für andere Schwellenwerte können andere Ergebnisse generiert werden. Für die Levenshtein-Distanz hat sich ein Schwellenwert von *3* bewährt. In Tabelle 6.9 ist die Menge der richtigerweise und fälschlicherweise als positiv bestimmten Verknüpfungen zwischen Text- und Modellelementen für eine Menge möglicher Verknüpfungen angegeben. Die Werte sind abhängig der drei Schwellenwerte *2*, *3* und *4* für Version 1 und 3 des Media Stores dargestellt, wobei die Anzahl Synonyme pro Wort auf drei festgelegt ist. Die betrachteten Dokumente weisen auf, dass mit einem Schwellenwert von *3* die höchste Ausbeute erreicht werden kann. Die Präzision fällt für einen Schwellenwert von *2* besser aus als für einen der höheren Schwellenwerte. Für eine bessere Präzision, unter Einbüßen der Ausbeute ist auch ein Schwellenwert von *2* für die Levenshtein-Distanz empfehlenswert. Ein Schwellenwert von 4 ist nicht empfehlenswert, in den evaluierten Versionen des Media Stores fällt sowohl die Ausbeute, als auch die Präzision schlechter aus als für einen Schwellenwert von 3. Die Idee der Wahl eines Schwellenwertes von *3* beruht auf der Länge der Wortendung *-ing* von drei. In Bezug auf die Synonyme der Bezeichner der SAD kann dies zu Problemen führen. Je mehr Synonyme auf Ähnlichkeit mit den Namen der Modellelemente überprüft werden, desto wahrscheinlicher ist es, dass eine falsche Verknüpfung erstellt wird. Ein Synonym kann eine Levenshtein-Distanz von kleiner gleich drei aufweisen, ohne von der Bedeutung her mit dem Namen des Modellelements in Relation zu stehen. Beispielsweise

Dokument	Mögliche Verknüpfungen	Schwellenwert	TP	FP
Media Store Version 1	52	0,09	31	4
		0,1	30	16
Media Store Version 3	49	0,09	16	4
		0,1	16	15

Tabelle 6.10.: Menge der richtigerweise (TP) und fälschlicherweise (FP) als positiv bestimmten Verknüpfungen abhängig des Schwellenwertes der Jaro-Winkler-Distanz.

entspricht die Levenshtein-Distanz von *Factor* und *Facade* dem Standardschwellenwert von drei. Auch zwischen den Wörtern *Case* und *Cache* wird eine falsche Verknüpfung erstellt. Diese Wörter sind von der Bedeutung her nicht miteinander verwandt, weisen aber gleiche Buchstaben und eine ähnliche Reihenfolge auf.

Für die Jaro-Winkler-Distanz hat sich ein Schwellenwert von *0,09* bewährt. In Tabelle 6.10 ist die Menge der richtigerweise und fälschlicherweise als positiv bestimmten Verknüpfungen zwischen Text- und Modellelementen, für eine Menge möglicher Verknüpfungen für Version 1 und 3 des Media Stores angegeben. Die Ergebnisse sind abhängig der zwei Schwellenwerte *0,09* und *0,1* für drei Synonyme pro Wort unter Anwendung des Synonym-basierten Vergleichsansatzes. Für die betrachteten Dokumente ist feststellbar, dass für den Schwellenwert *0,09* weniger fälschlicherweise positive Verknüpfungen bestimmt werden, ohne Einbußen der richtigerweise als positiv bestimmten Verknüpfungen. Für einen Schwellenwert von *0,1* kann die Ausbeute nicht erhöht werden und gleichzeitig nimmt die Präzision ab. Ein Schwellenwert größer gleich *0,1* scheint ungeeignet. Die Idee der Jaro-Winkler-Distanz ist, dass Wortendungen weniger Einfluss auf die Ähnlichkeit beim Vergleich zweier Zeichenketten haben. Dieses Vorgehen kann in Bezug auf die Namen von Modellelementen, die aus mehreren Teilwörtern bestehen zu Problemen führen. Beispielsweise für die Modellnamen *FileStore* und *MediaStore* ist es wahrscheinlich, dass diese im Text der SAD mit den Wörtern *files* und *media* beschrieben werden. Die Jaro-Winkler-Distanz von *FileStore* und *files* beträgt *0,074* und die von *MediaStore* und *media* beträgt *0,089*. Diese Werte liegen unterhalb des Schwellenwertes von *0,09* und es werden Verknüpfungen zwischen den Text- und Modellelementen erstellt. In diesem Fall hat die geringe Bedeutung der Wortendungen bei Bestimmung der Ähnlichkeit einen negativen Einfluss auf die Menge der fälschlicherweise als positiv bestimmten Verknüpfungen.

F4: Wie stark wird die Laufzeit des Synonym-basierten Vergleichsansatzes durch die Anzahl Synonyme pro Wort beeinflusst? Die Laufzeit der evaluierten Versionen des Media Stores beträgt durchschnittlich 60 Sekunden für den naiven Vergleichsansatz mit null Synonymen pro Wort. Bei einer Erhöhung auf drei Synonyme pro Wort beträgt die durchschnittliche Laufzeiterhöhung zehn Sekunden bei Verwendung des Synonym-basierten Vergleichsansatzes mit der Levenshtein-Distanz. Bei der Jaro-Winkler-Distanz beträgt die durchschnittliche Laufzeiterhöhung vier Sekunden[3]. Durchschnittlich steigt die Laufzeit

[3]Single-Threaded, bei Ausführung mit einem AMD Ryzen 5 2600 Prozessor

für null bis acht Synonyme pro Wort linear und es besteht kaum ein Unterschied zwischen der Levensthein- und der Jaro-Winkler-Distanz. Die Laufzeiterhöhung steigt acht bis zehn Synonyme stark an und bleibt nicht im linearen Bereich. Für die Jaro-Winkler-Distanz steigt die Laufzeit bei einer Erhöhung von 8 auf 10 Synonyme pro Wort von 6 auf 51 Minuten. Gleichzeitig hat die Evaluation ergeben, dass für zwei bis drei Synonyme pro Wort die Besten F_1-Werte erzielt werden. Für diesen Bereich skaliert die Laufzeit des Synonym-basierten Vergleichsansatzes linear. Damit sind getroffenen Entwurfsentscheidungen in Bezug auf die Laufzeit zu vernachlässigen. Es ist nicht nötig, die Suche nach Verknüpfungen in zwei Stufen aufzuteilen, wie in Abschnitt 5.3 beschrieben. Eine Suche nach Verknüpfungen mit den Wörtern einer Phrase, bevor versucht wird die gesamte Phrase einem Modellelement zuordnen, kann entfernt werden.

6.4. Beurteilung der Evaluationsergebnisse

In der Evaluation hat sich ergeben, dass der Synonym-basierte Vergleichsansatz für die Verknüpfung bessere Ergebnisse liefert als ein naiver Vergleichsansatz ohne Einbeziehung der Synonyme. Die Ausbeute kann durchschnittlich unter Einbeziehung der Synonyme um mehr als 10 Prozentpunkte erhöht werden, obwohl die Bedeutungen der Textelement der SAD durch den Babelfy-Algorithmus nur zu 41,2% korrekt bestimmt werden. Bei expliziter Betrachtung, für welche Elemente der Synonym-basierte Vergleichsansatz Verknüpfungen erstellt fällt auf, dass in vielen Fällen eine Verknüpfung zwischen *Database* und der Abkürzung *DB* erstellt wird. Die Auflösung dieser Abkürzung macht einen Teil der Erhöhung der Ausbeute aus. Bei Verwendung einer lexikalischen Ressource, welche keine Abkürzungen auflösen kann, würde die Ausbeute schlechter ausfallen. BabelNet kann diese Abkürzungen auflösen, in WordNet existiert die Abkürzung für *Database* nicht [18, 25].

Darüber hinaus kann mit dem Synonym-basierten Vergleichsansatz keine Verknüpfung zwischen den Elementen *DataStorage* und *FileStorage* erstellt werden. Es existiert kein Synset von *Data*, welches als Synonym das Wort *file* enthält. Für das Wort *File* hingegen existiert ein Synset, welches das Synonym *data file* enthält [18, 19]. Das bedeutet, dass von *FileStorage* auf *DataStorage* geschlossen werden kann. Hingegen kann von *DataStorage* nicht auf *FileStorage* geschlossen werden, obwohl es auf Basis der Herangehensweise des Synonym-basierten Vergleichsansatzes zu vermuten wäre. Es wird keine Verknüpfung erstellt, obwohl die Umbenennung des Modellnamens auf einem bedeutungsähnlichen Wort basiert. Dieses Beispiel ist ein Grund dafür, warum die Ausbeute nicht besser ausfällt, als in der Evaluation festgestellt wird. Weitere Ursachen, warum die Ausbeute nicht höher ausfällt, sind Wörter die umbenannt werden aber nicht bedeutungsverwandt sind. Beispielsweise die Umbenennung von *Web GUI* zu *Facade*. Mit dem Synonym-basierten Vergleichsalgorithmus besteht bei diesem Beispiel keine Möglichkeit eine Verknüpfung zu erstellen, da die Wörter nicht synonym zueinander sind.

Bei der Evaluation des Babelfy-Algorithmus ist anzumerken, dass die korrekten Bedeutungen im Kontext der SAD von einer Person bestimmt werden, die Englisch auf C1-Level beherrscht. Das bedeutet, dass unter Umständen die korrekten Bedeutungen

falsch bestimmt sind und das F_1-Maß gegebenenfalls besser oder schlechter ausfällt als angegeben. Darüber hinaus ist für die Evaluation von Babelfy nur ein Textdokument mit den korrekten Bedeutungen annotiert und evaluiert. Für allgemeine Ergebnisse ist es nötig, mehrere verschiedene Textdokumente einer SAD mit Babelfy zu evaluieren.

Die Evaluation des Synonym-basierten Vergleichsansatzes umfasst mehrere Versionen der SAD des Media Stores. Für allgemeine Ergebnisse ist es nötig, mehrere verschiedene SAD zu evaluieren, welche die Realität bei der Umbenennung von Elementen widerspiegeln. Die Ergebnisse der zwei bearbeiteten Versionen des Media Stores, Version 2 und Version 3, sind nicht so aussagekräftig wie die unbearbeitete Version 1. Darüber hinaus ist die SAD des Media Stores von Personen verfasst, deren Muttersprache nicht englisch ist. In der SAD liegen Syntaxfehler vor, die einen Einfluss auf die Ergebnisse des Synonym-basierten Vergleichsansatzes haben. Beispielsweise kommt es vor, dass die Phrasen eines Satzes durch den NLWrapper falsch bestimmt werden und somit zwei Modellnamen in einer Phrase enthalten sind. Eine Phrase kann im Vergleichsansatz nicht zwei, sondern nur einem Modellelement zugeordnet werden. Die Ausbeute fällt aufgrund dieses Fehlers schlechter aus.

7. Fazit

In diesem Kapitel wird zusammenfassend festgehalten, wie der Synonym-basierten Vergleichsansatz beim Erstellen der Verknüpfungen zwischen den Textelementen der SAD und den Modellelementen des SAM vorgeht. Darüber hinaus werden die Annahmen und Einschränkungen des Vergleichsansatzes benannt und es wird darauf eingegangen, wie der Ansatz weiterentwickelt werden kann. In Abschnitt 7.1 werden das Vorgehen des Synonym-basierten Vergleichsansatzes und die Ergebnisse der Evaluation zusammengefasst. In Abschnitt 7.2 werden Probleme bei der Verarbeitung durch den Synonym-basierten Vergleichsansatz aufgezeigt, sowie Lösungsvorschläge für diese eingebracht. In Abschnitt 7.3 wird ein Ausblick für das mögliche Vorgehen für den nächsten Schritt erläutert, um dem Ziel näher zu kommen, Inkonsistenzen zwischen SAD und SAM auffinden zu können. Darüber hinaus wird ein Verbesserungsvorschlag zur die Bestimmung der Ähnlichkeit zwischen den Text- und Modellelementen genannt.

7.1. Zusammenfassung

Änderungen an der Software ziehen unter Umständen Änderungen an der Dokumentation nach sich. Wird die Dokumentation nicht dementsprechend angepasst, kann es zu Inkonsistenzen bei der Benennung kommen. Trotz dieser Inkonsistenzen soll Rückverfolgbarkeit zwischen den Textelementen der SAD und den Modellelementen des SAM gewährleistet werden. Statt einem direkten Vergleich zwischen den Bezeichnern der Textelemente und den Namen der Modellelemente wird ein semantischer Vergleich auf Basis der Auflösung sprachlicher Mehrdeutigkeiten mit WSD durchgeführt. Mit dem WSD-Algorithmus Babelfy werden die Bedeutungen der Textelemente im Kontext der SAD in Form von Synsets bestimmt. Diese Synsets bilden Synonyme ab, die für das Erstellen von Verknüpfungen zwischen den Text- und Modellelementen verwendet werden.

Zur Umsetzung dieser Idee wird ein Synonym-basierter Vergleichsansatz in Form eines Agenten implementiert. Die Vorgehensweise dieses Ansatzes lässt sich in drei grundlegende Schritte unterteilen. Im ersten Schritt wird der Agent mit der SAD und dem SAM initialisiert. Im zweiten Schritt werden die Wörter der SAD im Kontext dieses Textdokuments mit dem WSD-Algorithmus Babelfy disambiguiert. Die Bedeutungen der Wörter im Kontext der SAD werden in Form von Synsets dargestellt. Diese Synsets enthalten Synonyme, welche im aus dem Textdokument aufgebauten Graphen als Attribute der Wortknoten annotiert werden. Besteht der Bezeichner eines Textelements aus mehreren Teilwörtern, werden auch die Synonyme der Teilwörter miteinander kombiniert und annotiert. Im dritten Schritt werden die Synonyme der Bezeichner der Textelementen verwendet um eine Verknüpfung mit den Namen der Modellelementen herzustellen. Dafür werden für einen Satz der SAD die einzelnen Phrasen betrachtet und versucht eine Verknüpfung mit einem

Modellelement herzustellen. Zur Bestimmung der Ähnlichkeit zwischen dem Bezeichner eines Textelements und dem Namen eines Modellelements stehen die Levenshtein- und die Jaro-Winkler-Distanz zur Verfügung. Liegt die Distanz von einem Bezeichner und einem Namen unterhalb des Schwellenwertes, wird eine Verknüpfung zwischen dem Text- und Modellelement erstellt. Diese Verknüpfung wird im Graphen durch eine Kante zwischen dem Wort- und Modellknoten vom Typ *model_relation* umgesetzt. Schritt 2 und Schritt 3 werden abwechselnd für alle Sätze der SAD durchgeführt.

Mit dem Synonym-basierten Vergleichsalgorithmus ist es möglich, Textelemente Modellelementen zuzuordnen, die semantisch dasselbe Element abbilden aber unterschiedlich benannt sind. Die Ausbeute kann beim Erstellen von Verknüpfungen um durchschnittlich 12 Prozentpunkte auf 53,1% erhöht werden, bei einer Präzision von 90,9%. Eine Wahl von zwei bis drei Synonymen pro Wort liefert durchschnittlich das beste F_1-Maß beim Erstellen der Verknüpfungen bei Anwendung des Synonym-basierten Vergleichsansatzes. Die Levenshtein-Distanz erreicht in den evaluierten Versionen des Media Stores übergreifend eine bessere Ausbeute als die Jaro-Winkler-Distanz. In Bezug auf die Präzision schneidet die Jaro-Winkler-Distanz besser ab. In diesem Zusammenhang hat sich für die Levenshtein-Distanz ein Schwellenwert von *3* und für die Jaro-Winkler-Distanz ein Schwellenwert von *0,09* bewährt. Die Laufzeit des Synonym-basierten Vergleichsansatzes steigt linear mit der Anzahl Synonyme pro Wort für bis zu acht Synonyme pro Wort.

Diese Ergebnisse des Synonym-basierten Vergleichsansatzes sind abhängig vom gewählten WSD-Algorithmus, der verwendeten lexikalischen Ressource, der Anzahl Synonyme pro Wort und den Verfahren zur Bestimmung der Ähnlichkeit zwischen den Bezeichnern der Textelemente und den Namen der Modellelemente. Mit diesem Synonym-basierten Vergleichsansatz kann eine Verbesserung beim Erstellen von Verknüpfungen zwischen den Textelementen der SAD und den Modellelementen des SAM erzielt werden gegenüber einem Vergleichsansatz ohne Einbeziehung von Synonymen.

7.2. Annahmen und Einschränkungen

Die Erstellung von Verknüpfungen zwischen Textelementen der SAD und Modellelementen des SAM mit dem Synonym-basierten Vergleichsansatz unterliegt der Annahme, dass die Bedeutung eines Modellnamens nach einer Umbenennung unverändert bleibt. Zwischen *StorageAccess* und *MemoryAccess* kann mit diesem Synonym-basierten Vergleichsansatz abhängig des Kontextes eine Verknüpfung erstellt werden. Eine Umbenennung von *Web GUI* zu *Facade* kann mit dem Vergleichsansatz hingegen nicht aufgelöst werden.

Darüber hinaus wird bei Anwendung des Synonym-basierten Vergleichsansatzes versucht die Ausbeute zu maximieren, indem Textelemente Modellelementen zugeordnet werden, wenn nur ein Teilwort des Textelements mit dem Namen des Modellelements übereinstimmt. Beispielsweise wird das Textelement *UserDBAdapter* dem Modellelement *DB* zugeordnet, wenn kein anderes Modellelement für eine Verknüpfung mit dem Textelement existiert. Diese Verknüpfung kann unter Umständen falsch sein und zu Problemen beim Auffinden von Inkonsistenzen zwischen SAD und SAM führen. Das mögliche Fehlen des Modellelements *UserDBAdapter* wird nicht erkannt, wenn eine Verknüpfung mit dem Modellelement *DB* erstellt wird. Dieses Problem könnte durch einen weiteren Schwellen-

wert gelöst werden, bei dem Teilwörter der Synonyme oder des Bezeichners zu mindestens 50% mit dem Namen des Modellelements übereinstimmen müssen. Damit würde eine Verknüpfung zwischen *DBAdapter* und *UserDBAdapter* erstellt werden, aber keine zwischen *DB* und *UserDBAdapter*.

Auch das Vorgehen, erst die einzelnen Wörter einer Phrase und danach die gesamte Phrase für eine Verknüpfung zu betrachten, kann zu ungewollten Fehlern führen. Bei dieser Reihenfolge wird bei Betrachtung der Phrase *user database adapter* eine Verknüpfung für das Wort *database* mit dem Modellelement *DB* erstellt. Anstatt einer Verknüpfung zwischen der gesamten Phrase und dem Modellelement *UserDBAdapter*. Darüber hinaus hat die Evaluation für den Synonym-basierten Vergleichsansatz ergeben, dass für zwei bis drei Synonyme pro Wort dieses Vorgehen unnötig ist. Zwei bis drei Synonyme pro Wort erzielen durchschnittlich die besten Ergebnisse und ziehen gleichzeitig nur geringe Laufzeiterhöhungen nach sich. An dieser Stelle scheint es deshalb sinnvoller, die Unterscheidung der zwei Stufen zu entfernen und direkt die gesamte Phrase für eine Verknüpfung zu betrachten. Die Laufzeit des Synonym-basierten Vergleichsansatz kann auch durch Parallelisierung der Verarbeitungsschritte verbessert werden. Die Annotation der Bedeutungen, als auch die Suche nach Verknüpfungen erfolgt für die einzelnen Sätze unabhängig voneinander und kann parallel ausgeführt werden.

Weitere Einschränkungen bei der Verarbeitung des Ansatzes ergeben sich dadurch, dass in der SAD keine neuen Modellelemente erkannt werden können, die nicht im SAM definiert sind. Es kann überprüft werden, ob in der SAD Änderungen der Softwarearchitektur dokumentiert werden, die durch neue Implementierungen entstanden sind. Es kann nicht überprüft werden, ob in der Software implementiert wird, was im SAM definiert ist. Im Allgemeinen wird zur Konsistenzprüfung zwischen Softwarearchitektur und informeller Dokumentation der Teil der Dokumentation betrachtet, welcher sich mit der textuellen Beschreibung der Softwarearchitektur auseinandersetzt (SAD). Andere Teile der Dokumentation, wie Benutzerszenarien, Anforderungen oder Risikomanagement, werden nicht betrachtet. Die SAD ist dabei auf den englischen Sprachraum und die Verarbeitung des SAM in Form einer Ontologie im OWL-Format begrenzt.

7.3. Ausblick

In Bezug auf die Bestimmung der Ähnlichkeit zwischen Zeichenketten können statt der Levenshtein- und Jaro-Winkler-Distanz andere Verfahren eingesetzt werden, wie beispielsweise *Path-Similarity*-Algorithmen. Diese bestimmen die semantische Ähnlichkeit von zwei Synsets auf Basis der lexikalischen Ressource WordNet [20]. Dabei wird das Synset des Bezeichners eines Textelements im Kontext der SAD mit dem Synset des Namens eines Modellelements im Kontext des SAM verglichen. Abhängig dieser Ähnlichkeit werden die Verknüpfungen zwischen den Text- und Modellelementen bestimmt. Ein Vergleich der einzelnen Wörter der Synsets mit den Namen der Modellelemente wäre mit einem *Path-Similarity*-Algorithmus nicht nötig. Stattdessen wird bestimmt, wie ähnlich die Bedeutungen der Synsets sind und ob diese Ähnlichkeit für eine Verknüpfung ausreicht.

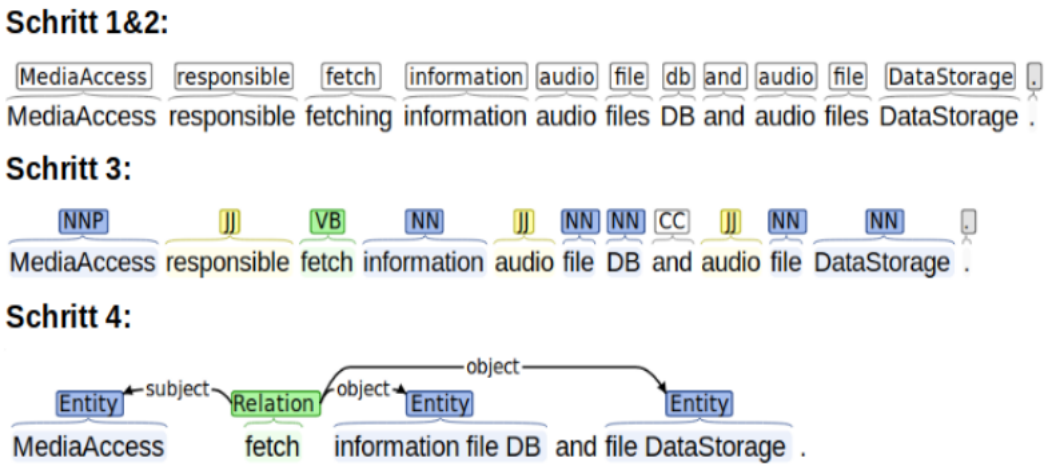


Abbildung 7.1.: Anwendung der Schritte zur Extraktion von Relationen zwischen Textelementen der SAD

Nachdem die Verknüpfungen zwischen Text- und Modellelementen bestimmt sind, wäre der nächste Schritt herauszufinden, welche Teile der im Text beschriebenen Relationen mit den im SAM definierten Relationen übereinstimmen. Arora et al. stellen in [3] einen Ansatz zur Extraktion von Modellen aus den Anforderungen eines Softwaredokuments vor. Relationen zwischen den Textelementen, sowie deren Attribute werden aus der Konstruktion der betrachteten Sätze extrahiert. Aus den gefundenen Relationen wird ein UML-Diagramm aufgebaut. Die Extraktion von Relationen ist bei diesem Ansatz auf kurze unabhängige Textabschnitte mit wenigen Sätzen beschränkt und nicht für ein gesamtes Textdokument ausgelegt [3]. Die grundlegende Vorgehensweise kann jedoch auf die einzelnen Sätze der SAD übertragen werden. Über die in der SAD verwendeten Verben können die Relationen zwischen den Textelementen extrahiert werden. Zur Identifikation der Relationen ist in Abbildung 7.1 eine mögliche Vorgehensweise für einen Beispielsatz in vier Schritten angegeben. Zuerst wird die Entfernung der Stoppwörter durchgeführt, welche in der Regel keinen Einfluss auf den Inhalt eines Textes und damit auf die Relationen zwischen den Textelementen haben [3]. Aus diesem Grund werden Stoppwörter für die Extraktion der Relationen nicht betrachtet. Im zweiten Schritt wird Lemmatisierung durchgeführt, welche alle Wörter in deren Grundform umformt. Zur Extraktion der Relationen stehen Nomen und Verben im Vordergrund. Diese werden im dritten Schritt mit POS-Tagging identifiziert. Nach Reduktion des Satzes auf Nomen, Verben und koordinierende Konjunktionen wird im vierten Schritt*Open IE* zur Extraktion der Relationen zwischen den Textelementen angewandt. Objekt und Subjekt einer Relation entsprechen den Bezeichnern der Textelemente und werden gegebenenfalls auf diese reduziert. Zur Extraktion der Relationen werden diejenigen Sätze der SAD betrachtet, für die eine Verknüpfung zwischen einem Text- und einem Modellelement hergestellt werden kann. Die im Text definierten Relationen können daraufhin mit den im SAM definierten Relationen verglichen werden. Die extrahierten Relationen können unter Umständen dazu beitragen, weitere Verknüpfungen zwischen Text- und Modellelementen zu erstellen.

Wörter- und Abkürzungsverzeichnis

Glossar

Ausbeute Die Ausbeute kann einen Wert zwischen 0 und 1 annehmen. Ein Ausbeute-Wert von 1 bedeutet, dass alle korrekten Verknüpfungen gefunden wurden (True Positive, TP). Unter den gefunden Verknüpfungen können aber auch Verknüpfungen sein, die nicht korrekt sind (False Positive, FP). 13, 14, 29, 33, 35, 37–39, 41–52, 54, 63–66

Babelfy Babelfy ist ein Knowledge Based WSD-Algorithmus zur Bestimmung der Bedeutung von Wörtern abhängig des Kontextes in dem sie stehen. vii, 5–7, 14, 21, 23, 25–28, 30, 32, 37, 38, 51–53, 63

BabelNet BabelNet bildet eine große lexikalische Datenbank verschiedener Sprachen ab. Es basiert auf Synsets und bildet somit ein Netwerk aus Wörtern und deren Relationen zueinander ab. 6–8, 14, 20, 21, 25, 26, 29, 30, 51

F_1-Maß Das F_1-Maß bildet das harmonische Mittel aus Ausbeute und Präzision. vii, 6, 14, 15, 25, 37–49, 52, 54, 63–66

$F_{\sqrt{2}}$-Maß Das $F_{\sqrt{2}}$-Maß bildet das Mittel aus Ausbeute und Präzision, wobei die Ausbeute zweimal so hoch gewichtet wird wie die Präzision. 38–41, 43, 45, 46, 48, 49, 63–66

Information Retrieval Computergestütztes Auffinden und Abrufen von Informationen aus Text- und Bilddaten. 13, 38

Korreferenzen Für ein Wort können innerhalb eines Satzes oder zwischen mehreren Sätzen verschiedene sprachliche Ausdrücker verwendet werden, um eine explizite Wiederholung des Wortes zu vermeiden (beispielsweise dieses, es, er usw.). 5

Lemmatisierung Ist Teil der syntaktischen Analyse, wobei man darunter die Umformung eines Wortes in dessen Grundform versteht. 4

Machine Learning Mustererkennung in Daten durch vorhergehendes Lernen von Trainingsdaten. 5, 14, 15

Präzision Die Präzision kann einen Wert zwischen 0 und 1 annehmen. Ein Präzisions-Wert von 1 bedeutet, dass alle gefundenen Verknüpfungen korrekt (TP) sind. Es kann jedoch sein, dass nicht alle korrekten Verknüpfungen gefunden wurden. 13, 14, 33, 35, 37–39, 41–50, 54, 63–66

Synsets Ein Synset ist ein Set aus synonymen Wörtern. Diese Wörter stehen in Relation mit anderen Wörtern und deren Synsets. Synsets bilden ein Netzwerk aus den Relationen von Wortbedeutungen. 2, 5, 7–9, 15, 19, 21, 23, 25, 26, 37, 53, 55

Thesaurus Referenzwerk (beispielsweise WordNet), welches bedeutungsverwandte Wörter auflistet (Im Gegensatz zu einem Wörterbuch, welches die Definitionen von Wörtern auflistet). 5

Traceability Rückverfolgbarkeit von Daten, die miteinander in Beziehung stehen. 13

Traceability Links Explizite Verbindungen von einem Quell- zu einem Zielartefakt, die eine Rückverfolgbarkeit gewährleisten. Diese Verbindungen und die Menge aller Artefakte bilden die Kanten und Knoten eines Graphen, mit dem zurückverfolgt werden kann, welche Artefakte miteinander in Beziehung stehen.. 13, 14

WordNet WordNet ist eine große lexikalische Datenbank der englischen Sprache. Es basiert auf Synsets und bildet somit ein Netzwerk aus allen Wörtern der englischen Sprache und wie diese in Realtion zueinander stehen. 5–7, 15, 51, 55

Akronyme

API Programmierschnittstelle (Application Programming Interface). 6–8, 21, 23, 25

EL Entitätsverknüpfungen (Entity Linking). 4, 6, 7, 14, 25

MCS Most Common Sense. 6, 7, 38

NLP Verarbeitung natürlicher Sprache (Natural Language Processing). 4, 5, 11, 13, 17, 18

NLU Verstehen natürlicher Sprache (Natural Language Understanding). 4, 11, 17–19, 21

OWL Web-Ontologie-Sprache (Web Ontology Language). 11, 19, 23, 55

POS-Tagging Zuordnung von Wörtern zu Wortarten (Part-Of-Speech-Tagging). 4, 26, 27, 56

SAD Softwarearchitektur-Dokumentation (Software Architecture Documentation). 1–4, 11, 13–15, 17–21, 23–28, 35, 37–42, 44, 45, 47–56

SAM Softwarearchitektur-Modell (Software Architecture Model). 1–3, 11, 13, 14, 17–20, 23, 33, 35, 37–41, 47, 48, 53–56

WSD Auflösung sprachlicher Mehrdeutigkeiten (Word Sense Disambiguation). vii, 2, 5–7, 13–15, 19–21, 23, 25, 37, 38, 40, 53, 54

Literatur

[1] 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: *Computing Research Repository* (2018).

[2] Guliano Antoniol u. a. „Recovering Traceability Links between Code and Documentation“. In: *IEEE Transactions on Software Engineering* 28.10 (Okt. 2002), S. 970–983. ISSN: 0098-5589. DOI: 10.1109/TSE.2002.1041053.

[3] Chetan Arora u. a. „Extracting Domain Models from Natural-language Requirements: Approach and Industrial Evaluation“. In: *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*. Saint-malo, France: ACM, 2016, S. 250–260. ISBN: 978-1-4503-4321-3. DOI: 10.1145/2976767.2976769.

[4] Devendra Singh Chaplot und Ruslan Salakhutdinov. „Knowledge-based Word Sense Disambiguation using Topic Models“. In: *The Thrity-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. Hrsg. von Machine Learning Department School of Computer Science Carnegie Mellon University. 2018, S. 5062–5069.

[5] Gobinda G. Chowdhury. „Natural Language Processing“. In: *Annual Review of Information Science and Technology* 37.1 (2003), S. 51–89.

[6] Paul Clements u. a. *Documenting Software Architectures : Views and Beyond*. Hrsg. von Paul Clements. 2. Aufl. SEI Series in Software Engineering. Boston, MA: Addison-Wesley, Pearson Education, 2011. ISBN: 0321552687.

[7] Barthélémy Dagenais und Martin P. Robillard. „Creating and Evolving Developer Documentation: Understanding the Decisions of Open Source Contributors“. In: *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE '10. Santa Fe, New Mexico, USA: ACM, 2010, S. 127–136. ISBN: 978-1-60558-791-2. DOI: 10.1145/1882291.1882312.

[8] Orlena Gotel u. a. „Traceability Fundamentals“. In: *Software and Systems Traceability*. Hrsg. von Jane Cleland-Huang, Orlena Gotel und Andrea Zisman. London: Springer London, 2012, S. 3–22. ISBN: 978-1-4471-2239-5. DOI: 10.1007/978-1-4471-2239-5_1.

[9] Thomas R. Gruber. „A translation approach to portable ontology specifications“. In: *Knowledge Acquisition* 5.2 (1993), S. 199–220. ISSN: 1042-8143. DOI: https://doi.org/10.1006/knac.1993.1008.

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

[11] Dan Jurafsky und James H. Martin. *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Hrsg. von Marcia J. Horton, Tracy Dunkelberger und Melinda Haggerty. 2. Aufl. Prentice Hall series in artificial intelligence. Upper Saddle River, NJ: Prentice Hall, Pearson Education International, 2009. ISBN: 0135041961.

[12] Jan Keim und Anne Koziolek. „Towards Consistency Checking between Software Architecture and Informal Documentation". In: *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. 2019.

[13] Joachim Kleb. „Ontologie-baiserte Monoseminierung: Bestimmung von Referenzen im Semantic Web". Diss. Fakultät für Wirtschaftswissenschaften am Karlsruher Institut für Technologie, Dez. 2012.

[14] Brinardi Leonardo und Seng Hansun. „Text Documents Plagiarism Detection using Rabin-Karp and Jaro-Winkler Distance Algorithms". In: *Indonesian Journal of Electrical Engineering and Computer Science* 5.2 (2017), S. 462–471.

[15] Paul W. McBurney und Collin McMillan. „Automatic Documentation Generation via Source Code Summarization of Method Context". In: *Proceedings of the 22Nd International Conference on Program Comprehension*. ICPC 2014. Hyderabad, India: ACM, 2014, S. 279–290. ISBN: 978-1-4503-2879-1. DOI: 10.1145/2597008.2597149.

[16] Tristan Miller u. a. „DKPro WSD: A Generalized UIMA-based Framework for Word Sense Disambiguation". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (System Demonstrations) (ACL 2013)*. Aug. 2013, S. 37–42.

[17] Andrea Moro, Francesco Cecconi und Roberto Navigli. „Multilingual Word Sense Disambiguation and Entity Linking for Everybody". In: *Proceedings of the 13th International Semantic Web Conference, Posters and Demonstrations (ISWC 2014)*. Riva del Garda, Italy, 2014, S. 25–28.

[18] Andrea Moro und Robert Navigli. *Babelfy*. `http://babelfy.org/`. Besucht am 03.06.19.

[19] 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 (2014), S. 231–244.

[20] Ted Pedersen, Siddharth Patwardhan und Jason Michelizzi. „WordNet::Similarity: Measuring the Relatedness of Concepts". In: *Demonstration Papers at HLT-NAACL 2004*. HLT-NAACL–Demonstrations '04. Boston, Massachusetts: Association for Computational Linguistics, 2004, S. 38–41.

[21] 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*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, S. 99–110.

[22] Sebastian Ruder. *NLP Progress: Word Sense Disambiguation*. http://nlpprogress.com/english/word_sense_disambiguation.html. (Besucht am 17.05.19). Dez. 2018.

[23] Misha Strittmatter und Amine Kechaou. *The Media Store 3 Case Study System*. Software 1. Karlsruher Institut für Technologie, Fakultät für Informatik, 2016.

[24] Princeton University. *About WordNet*. https://wordnet.princeton.edu/. (Besucht am 16.05.19). 2010.

[25] Princeton University. *WordNet Search - 3.1*. http://wordnetweb.princeton.edu/perl/webwn. (Besucht am 22.05.19). 2010.

[26] S. Weigelt und W.F. Tichy. „Poster: ProNat: An Agent-Based System Design for Programming in Spoken Natural Language". In: *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*. Bd. 2. Mai 2015, S. 819–820. DOI: 10.1109/ICSE.2015.264.

[27] Rebekka Wohlrab u. a. „Improving the Consistency and Usefulness of Architecture Descriptions: Guidelines for Architects". In: *2019 IEEE International Conference on Software Architecture (ICSA)*. IEEE. 2019, S. 151–160.

[28] Li Yujian und Liu Bo. „A Normalized Levenshtein Distance Metric". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (Juni 2007), S. 1091–1095. DOI: 10.1109/TPAMI.2007.1078.

A. Anhang

A.1. Evaluationsergebnisse von Babelfy

Annotiert	Wörter	TP	TN	FP	FN	Ausbeute	Präzision	F_1-Maß
306	571	126	351	180	0	100%	41,2%	58,3%

Tabelle A.1.: Babelfy-Ergebnisse im Kontext der Softwarearchitektur-Dokumentation von Version 1 des Media Stores

A.2. Evaluationsergebnisse des Ansatzes

A.2.1. Evaluation des Ansatzes mit der Levenshtein-Distanz

Synonyme	Laufzeit (sec.)	TP	FP	FN	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
10	164	36	11	16	69,2	76,6	72,7	71,5
9	119	36	11	16	69,2	76,6	72,7	71,5
8	101	36	11	16	69,2	76,6	72,7	71,5
7	70	36	11	16	69,2	76,6	72,7	71,5
6	67	36	10	16	69,2	78,3	73,5	72,0
5	62	36	10	16	69,2	78,3	73,5	72,0
4	60	36	10	16	69,2	78,3	73,5	72,0
3	56	36	5	16	69,2	87,8	77,4	74,5
2	53	36	2	16	69,2	94,7	80,0	76,1
1	53	34	1	18	65,4	97,1	78,1	73,4
0	51	34	0	18	65,4	100	79,1	73,9

Tabelle A.2.: Evaluationsergebnisse des Synonym-basierten Vergleichsansatzes für Version 1 des Media Stores mit der Levenshtein-Distanz und einem Schwellenwert von 3 (für 52 mögliche Verknüpfungen)

Synonyme	Laufzeit (sec.)	TP	FP	FN	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
10	714	24	30	16	60,0	44,4	51,1	53,7
9	529	24	27	16	60,0	47,1	52,3	55,0
8	233	24	27	16	60,0	47,1	52,3	55,0
7	167	24	23	16	60,0	51,1	55,2	56,7
6	126	24	20	16	60,0	54,6	57,1	58,1
5	93	24	18	16	60,0	57,1	58,5	59,0
4	81	24	18	16	60,0	57,1	58,5	59,0
3	76	25	9	14	64,1	73,5	68,5	67,0
2	73	24	10	16	60,0	70,6	64,9	63,2
1	71	18	7	22	45,0	72,0	55,4	51,4
0	72	18	5	23	43,9	78,2	56,2	51,4

Tabelle A.3.: Evaluationsergebnisse des Synonym-basierten Vergleichsansatzes für Version 2 des Media Stores mit der Levenshtein-Distanz und einem Schwellenwert von 3 (für 41 mögliche Verknüpfungen)

Synonyme	Laufzeit (sec.)	TP	FP	FN	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
10	178	22	10	27	44,9	68,8	54,3	50,8
9	132	22	10	27	44,9	68,8	54,3	50,8
8	124	22	10	27	44,9	68,8	54,3	50,8
7	107	21	9	28	42,9	70,0	53,2	49,2
6	103	18	8	31	36,7	69,2	48,0	43,6
5	97	18	8	31	36,7	69,2	48,0	43,6
4	94	19	8	30	38,8	70,4	50,0	45,6
3	85	19	5	30	38,8	79,2	52,1	46,7
2	58	19	1	30	38,8	95,0	55,1	48,3
1	56	15	1	34	30,6	93,8	46,2	39,5
0	55	13	0	36	26,5	100	41,9	35,1

Tabelle A.4.: Evaluationsergebnisse des Synonym-basierten Vergleichsansatzes für Version 3 des Media Stores mit der Levenshtein-Distanz und einem Schwellenwert von 3 (für 49 mögliche Verknüpfungen)

A.2.2. Evaluation des Ansatzes mit der Jaro-Winkler-Distanz

Synonyme	Laufzeit (sec.)	TP	FP	FN	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
10	229	31	4	21	59,6	88,6	71,3	66,9
9	149	31	4	21	59,6	88,6	71,3	66,9
8	126	31	4	21	59,6	88,6	71,3	66,9
7	93	31	4	21	59,6	88,6	71,3	66,9
6	79	31	4	21	59,6	88,6	71,3	66,9
5	66	31	4	21	59,6	88,6	71,3	66,9
4	58	31	4	21	59,6	88,6	71,3	66,9
3	54	31	4	21	59,6	88,6	71,3	66,9
2	53	31	0	21	59,6	100	74,7	68,9
1	52	29	0	23	55,8	100	71,6	65,4
0	52	28	0	24	53,9	100	70,0	63,6

Tabelle A.5.: Evaluationsergebnisse des Synonym-basierten Vergleichsansatzes für Version 1 des Media Stores mit der Jaro-Winkler-Distanz und einem Schwellenwert von 0,09 (für 52 mögliche Verknüpfungen)

Synonyme	Laufzeit (sec.)	TP	FP	FN	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
10	3072	24	10	16	60,0	70,6	64,9	63,2
9	1933	24	10	16	60,0	70,6	64,9	63,2
8	362	24	10	16	60,0	70,6	64,9	63,2
7	362	24	10	16	60,0	70,6	64,9	63,2
6	192	24	10	16	60,0	70,6	64,9	63,2
5	102	24	10	16	60,0	70,6	64,9	63,2
4	84	24	10	16	60,0	70,6	64,9	63,2
3	78	23	9	17	57,5	71,9	63,9	61,6
2	73	24	4	16	60,0	85,7	70,6	66,7
1	72	18	1	23	43,9	94,7	60,0	53,5
0	72	17	1	24	41,5	94,4	57,6	51,0

Tabelle A.6.: Evaluationsergebnisse des Synonym-basierten Vergleichsansatzes für Version 2 des Media Stores mit der Jaro-Winkler-Distanz und einem Schwellenwert von 0,09 (für 41 mögliche Verknüpfungen)

Synonyme	Laufzeit (sec.)	TP	FP	FN	Ausbeute	Präzision	F_1-Maß	$F_{\sqrt{2}}$-Maß
10	210	19	5	30	38,8	79,2	52,1	46,7
9	130	18	5	31	36,7	78,3	50,0	44,6
8	117	18	5	31	36,7	78,3	50,0	44,6
7	98	17	4	32	34,7	80,1	48,6	42,9
6	82	15	4	34	30,6	79,0	44,1	38,5
5	71	15	4	34	30,6	79,0	44,1	38,5
4	61	16	4	33	32,7	80,0	46,4	40,7
3	58	16	4	33	32,7	80,0	46,4	40,7
2	58	15	0	34	30,6	100	46,9	39,8
1	56	11	0	38	22,5	100	36,7	30,3
0	55	8	0	41	16,3	100	28,1	22,6

Tabelle A.7.: Evaluationsergebnisse des Synonym-basierten Vergleichsansatzes für Version 3 des Media Stores mit der Jaro-Winkler-Distanz und einem Schwellenwert von 0,09 (für 49 mögliche Verknüpfungen)