Entity Linking für Softwarearchitektur Dokumentation

Bachelorarbeit von

Philipp Domenik Klaus

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

Erstgutachter:	Prof. Dr. Anne Koziolek
Zweitgutachter:	Prof. Dr. Ralf Reussner
Betreuender Mitarbeiter:	M.Sc. Jan Keim
Zweiter betreuender Mitarbeiter:	M.Sc. Dominik Fuchß

10. Juni 2022 – 10. Oktober 2022

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Zusammenfassung

Softwarearchitekturdokumentationen enthalten Fachbegriffe aus der Domäne der Softwareentwicklung. Wenn man diese Begriffe findet und zu den passenden Begriffen einer Datenbank verknüpft, können Menschen und Textverarbeitungssysteme diese Informationen verwenden, um die Dokumentation besser zu verstehen. Die Fachbegriffe in Dokumentationen entsprechen dabei Entitätserwähnungen im Text.

In dieser Ausarbeitung stellen wir unser domänenspezifisches Entity-Linking-System vor. Das System verknüpft Entitätserwähnungen innerhalb von Softwarearchitekturdokumentationen zu den zugehörigen Entitäten innerhalb einer Wissensbasis. Der von uns entwickelte Ansatz ist modular aufgebaut. Das System enthält eine domänenspezifische Wissensbasis, ein Modul zur Vorverarbeitung und ein Entity-Linking-System. Die domänenspezifische Wissensbasis haben wir im Rahmen dieser Ausarbeitung selbst erstellt. Die Wissensbasis enthält Entitäten aus der Domäne der Softwareentwicklung. Die Vorverarbeitung untergliedert sich in die *Named-Entity-Recognition* zum Finden von Entitätserwähnungen und eine Abkürzungsverarbeitung zur Auflösung selbstdefinierter Abkürzungen. Das Entity-Linking-System untergliedert sich in weitere Module und verbindet identifizierte Entitätserwähnungen zu den passenden Entitäten unserer Wissensbasis.

Für die Evaluation unseres Systems werden Architekturdokumentationen verwendet. Wir haben für eine Auswahl von vier Fallstudien zugehörige Gold-Standards erstellt. Mit Hilfe der Gold-Standards haben wir das gesamte Entity-Linking-System mit den Metriken Präzision, Ausbeute und F_1-Wert evaluiert. Unser Ansatz erreicht einen durchschnittlich gewichteten F_1-Wert von 0.7271.

Inhaltsverzeichnis

1 Einleitung

Die Kommunikation zwischen Personen derselben Domäne ist durch domänenspezifisches Vokabular geprägt. In wissenschaftlichen, wie auch in vielen allgemeinen Domänen, ist bereits ein Vokabular aus Fachbegriffen, Abkürzungen und anderen domänenspezifischen Begriffen etabliert. Mit diesem Vokabular können Personen derselben Domäne präzise miteinander kommunizieren. Die Arbeit von Chung und Nation [6] zeigt, dass Fachbegriffe einen großen Anteil in Fachtexten ausmachen. In ihrer Studie wurde ca. jedes dritte Wort eines Fachtextes über Anatomie als Fachbegriff identifiziert. Wenn Personen das Wissen über Vokabular aus einer Domäne fehlt, können sie Texte und Gespräche dieser Domäne schlechter verstehen, da ihnen das konzeptionelle Wissen in der Disziplin fehlt. Die Arbeit von Nagy und Townsend [22] behauptet, dass fachspezifische Wörter und ihr Verständnis wesentlich sind, um konzeptionelles Wissen in den Disziplinen aufzubauen, in denen sie verwendet werden.

Folglich ist ein erster Schritt, um ein besseren Verständnis eines Textes einer fremden Domäne zu bekommen, die Identifikation der verwendeten Fachbegriffe. Ein menschlicher Leser kann unbekannte Begriffe nachschlagen und mithilfe der Satzstruktur und des Kontexts erkennen, welche Wörter Fachbegriffe sind. Menschliche Leser sind so in der Lage, schrittweise einen Text zu lesen und inhaltlich zu verstehen. Texte sollen jedoch nicht nur von Menschen gelesen, verstanden und bearbeitet werden. Auch Maschinen arbeiten mit Text und müssen diesen verarbeiten können. Das Internet bietet einen direkten Zugang zu einer stetig steigenden Anzahl an Texten und Literatur. Darum wird es immer wichtiger, auch automatisierte Textverarbeitung zu verwenden. Automatisierte Textverarbeitung ist, ähnlich wie der Mensch, auf Kontextinformationen und Kenntnisse über die Domäne angewiesen.

Eine Möglichkeit, Domänenwissen in automatisierte Textverarbeitung zu integrieren, ist die Verwendung einer zusätzlichen Datenbank mit domänenspezifischem Wissen. Beispiele für solche Datenbanken sind Wissensbasen mit umfangreichen Informationen über die Entitäten der Welt, ihren semantischen Klassen und ihren gegenseitigen Beziehungen. Bekannte domänenübergreifende Beispiele sind die Wissensbasen DBpedia [1] und YAGO [24]. Automatisierte Textverarbeitungssysteme verwenden die Wissensbasen, um ein Verständnis über Fachbegriffe, Abkürzungen und den Kontext von Texten zu erhalten. Dazu identifizieren sie Entitätserwähnungen im unstrukturierten Text. Anschließend verknüpfen die Systeme die identifizierten Entitätserwähnung mit den Entitäten in einer Wissensbasis. Dieser Prozess heißt Entity Linking.

Bei den zu verlinkenden Entitätserwähnungen handelt es sich z.B. um Fachbegriffe, Personen oder Organisationen. Von Verlinkungen zwischen Entitätserwähnungen und Entitäten einer Wissensbasis können sowohl menschliche Leser als auch automatisierte Textverarbeitungssysteme profitieren.

Ein vermehrtes Aufkommen von Wissensbasen hat große Fortschritte am Entity-Linking-Prozess bewirkt. Mehrere Entity-Linking-Systeme wurden mit verschiedenen Wissensbasen und verschiedenen Ansätzen zur Verlinkung entwickelt [28, 19, 9, 29, 8]. Shen et al. haben dazu eine strukturierte Übersicht erstellt [27]. Für existierende Entity-Linking-Systeme sind vor allem Namensvariationen und Mehrdeutigkeiten große Herausforderungen. Das Problem der Namensvariationen beschreibt, dass eine Entität mehrere Namen hat. Ein Beispiel für eine Namensvariation ist die Stadt New York, die auch als Big Apple, NYC oder NY bekannt ist. Im Gegensatz dazu ist es auch möglich, dass eine Entitätserwähnung auf verschiedene Entitäten verweisen kann. Dieses Problem wird als Mehrdeutigkeit bezeichnet. Ein Beispiel für eine mehrdeutige Entitätserwähnung ist der Name *Michael Jordan*, der sowohl auf den berühmten Basketballspieler als auch auf einen Informatik-Professor von der University of California, Berkeley verweisen kann. Entity-Linking-Systeme werden oft zur Vorverarbeitung für eine Vielzahl an *Natural-Language-Processing-Systemen* (NLP-Systemen) verwendet. Insbesondere erleichtert ein Entity Linking Aspekte von *Information-Retrieval-Systemen* [5], Fragebeantwortungssystemen [35] oder der Auffüllung von Wissensbasen (*Knowledge Base Population*).

Wir betrachten in dieser Arbeit die Domäne der Softwareentwicklung. Speziell betrachten wir die Softwarearchitekturdokumentation von Softwareprojekten. Bei Entitätserwähnungen in der Softwareentwicklung gibt es ebenfalls die Herausforderung der Namensvariationen und Mehrdeutigkeiten. Ansätze zur Interpretation der Dokumentation benötigen zusätzliche Informationen, um den Text zu verarbeiten. Das Forschungsprojekt ArDoCo [15] zielt darauf ab, Konsistenzanalysen zwischen verschiedenen Arten von Dokumentationen bereitzustellen. Das ArDoCo-System identifiziert automatisch Inkonsistenzen zwischen formalen Modellen und informeller textueller Dokumentation. Unser Entity-Linking-System kann als Vorverarbeitungsschritt für das ArDoCo-System verwendet werden, um Zusatzinformationen aus der textuellen Dokumentation zu extrahieren und bereitzustellen. In der Softwarearchitekturdokumentation identifizieren wir Entitätserwähnungen und verknüpfen diese anschließend mit Entitäten in der Wissensbasis.

Die Arbeit untergliedert sich in zwei Teilaspekte. Der erste Teil umfasst die Erstellung einer domänenspezifischen Wissensbasis. Der zweite Teil umfasst die Entwicklung eines Entity-Linking-Systems, das die erstellte Wissensbasis verwendet.

Der erste Aspekt umfasst die Entwicklung und Population einer Wissensbasis für die Domäne der Softwareentwicklung. Wissensbasen wie DBpedia [1] und Yago [24] sind nicht domänenspezifisch. Für unser System, bei dem die Domäne festgelegt ist, verwenden wir keine allgemeine Wissensbasis. Unser System arbeitet ausschließlich mit Softwarearchitekturdokumentationen und soll auf Entitäten dieser Domäne spezialisiert sein. Für die Domäne der Softwareentwicklung gibt es aktuell keine Wissensbasis, die unseren Anforderungen genügt. Die Anforderungen und weitere Details zur Wissensbasis befinden sich in Kapitel 5.

Der zweite Aspekt umfasst die Entwicklung eines Entity-Linking-Systems. Dazu sucht das System Entitätserwähnungen in der Architekturdokumentation und verlinkt diese anschließend mit den Entitäten einer Wissensbasis. Es existiert bereits eine große Auswahl an Entity-Linking-Systemen. Unser System unterscheidet sich von existierenden Systemen durch die festgelegte Domäne und die Verwendung einer domänenspezifischen Wissensbasis. Entity-Linking-Systeme können auf verschiedene Anwendungsfälle optimiert werden.

Ein Beispielsystem eines spezifischen Anwendungsfalls ist das Entity-Linking-System für *micro-blogging-services* [9] zur Identifizierung von Entitäten in Tweets. Für den Anwendungsfall der Softwarearchitekturdokumentation von Softwareprojekten gibt es aktuell noch kein Entity-Linking-System.

Die Arbeit beschäftigt sich mit der Frage, welcher Anteil der Entitätserwähnungen von Softwarearchitekturdokumentationen sich mit einer domänenspezifischen Wissensbasis in Verbindung bringen lässt. Außerdem wird untersucht, wie präzise die Entitätserwähnungen automatisiert zu Entitäten der Wissensbasis zugeordnet werden können.

In dieser Arbeit sind die Grundlagen in Kapitel 2 enthalten. Kapitel 3 enthält verwandte Arbeiten und Kapitel 4 beschreibt den Ansatz unserer Arbeit. Die Eigenschaften und der Aufbau der Wissensbasis befinden sich in Kapitel 5. Details bezüglich der *Named-Entity-Recognition* befinden sich in Kapitel 6 und die Abkürzungsverarbeitung ist in Kapitel 7 zu finden. Die Ausarbeitung bezüglich der Module des Entity-Linking-Systems befinden sich in den Kapiteln 8, 9 und 10. Die *Candidate-Entity-Generation* ist in Kapitel 8, das *Candidate-Entity-Ranking* in Kapitel 9 und die *Unlinkable-Mention-Prediction* in Kapitel 10 enthalten. Anschließend sind die Gefährdungen der Validität in Kapitel 12 zu finden. Die Schlussfolgerung mit zukünftigen Arbeiten befindet sich in Kapitel 13.

2 Grundlagen

Dieses Kapitel beschäftigt sich mit den Grundlagen von Entity-Linking-Systemen. Dazu wird zunächst das verwandte Problem *Entity-Coreference-Resolution* in Sektion 2.1 ausgeführt. Anschließend wird die Vorverarbeitung von Entity-Linking-Systemen in Sektion 2.2 thematisiert. Die Vorverarbeitungsschritte untergliedern sich in die Abkürzungsverarbeitung und die *Named-Entity-Recognition* [21]. Nach den Vorverarbeitungsmodulen folgt der Aufbau des Entity-Linking-Systems in Sektion 2.3. Dabei werden Entity-Linking-Systeme in drei Module unterteilt, die separat behandelt werden.

2.1 Entity-Coreference-Resolution

Das Entity-Linking-System ist auf eine Wissensbasis angewiesen. Falls keine Wissensbasis vorhanden ist, reduziert sich das Problem auf das *Entity-Coreference-Resolution-Problem* [30, 2, 12]. Im Rahmen dieser Problemstellung erfolgt die Entitätsidentifikation innerhalb eines oder mehrerer Dokumente. Die Entitäten im Text werden in verschiedene Cluster gruppiert, von denen jedes Cluster eine bestimmte Entität darstellt. Fallen mehrere Entitäten im Text in denselben Cluster, teilen sie sich die inhaltliche Bedeutung und unterscheiden sich nur in ihrer Oberflächenform. *Entity-Coreference-Resolution-Systeme* arbeiten mit dem Kontext der Entitätserwähnung, Dokumentstatistiken und der Entität selbst.

Alle Informationen, die einem *Entity-Coreference-Resolution-System* zur Verfügung stehen, stehen ebenfalls einem Entity-Linking-System zur Verfügung. Entity-Linking-Systeme können jedoch zusätzliche Informationen aus der Wissensbasis verwenden und diese in der Verarbeitung miteinbeziehen.

2.2 Vorverarbeitungsmodule

In dieser Sektion werden die notwendigen Vorverarbeitungsschritte und -module eines Entity-Linking-Systems definiert. Die Vorverarbeitung umfasst die Module Abkürzungsverarbeitung und *Named-Entity-Recognition*. Beide Module arbeiten mit der Architekturdokumentation und liefern dem anschließenden Entity-Linking-System wichtige Zusatzinformationen.

2.2.1 Abkürzungsverarbeitung

Die Abkürzungsverarbeitung beschreibt die automatische Erkennung und Auflösung selbstdefinierter Abkürzungen im Text. Offizielle Abkürzungen können von einem Entity-Linking-System mit Hilfe der Wissensbasis aufgelöst und verlinkt werden. Selbstdefinierte

Abkürzungen müssen vor dem Entity Linking im Text erkannt werden. Das erste Vorkommen einer selbstdefinierten Abkürzung muss an einer Langform ausgerichtet werden. Zum Beispiel entspricht die Definition *The **b**ody **o**f **k**nowldge (BOK)* einer selbstdefinierten Abkürzung. Die Großbuchstaben *BOK* entsprechen der Abkürzung und ***b**ody **o**f **k**nowldge* entspricht der identifizierten Langform, an der die Abkürzung ausgerichtet wird. Anschließend müssen weiteren Verwendungen der selbstdefinierten Abkürzungen im Text gefunden und aufgelöst werden.

Im Folgenden ist eine Auswahl von Herausforderungen und Fehlerquellen der Abkürzungserkennung und Auflösung gegeben. Die Herausforderungen und Fehlerquellen entstammen aus der Ausarbeitung von Vanopstal et al. [32].

Herausforderungen

1. Die Unterscheidung zwischen einem gewöhnlichen Klammermuster und Klammern, die eine Abkürzung enthalten.
2. Viele Abkürzungen lassen sich nicht den Anfangsbuchstaben der Definition zuordnen.
3. Nicht alle Wörter sind in der Abkürzung enthalten. Oftmals werden Funktionswörter wie *of, in, for* weggelassen.
4. Je länger die Abkürzung, desto eindeutiger ist sie. Dies ist herausfordernd für die Auflösung von besonders kurzen Abkürzungen.

Fehlerquellen in der Abkürzungserkennung für *false positives*

1. Titel, die in Großbuchstaben gedruckt werden.
2. Initialen des Vornamens, die fälschlicherweise als Abkürzung erkannt werden.
3. Römische Ziffern, die mit Großbuchstaben I, V oder X verwechselt werden.
4. Einzelne Buchstaben, die keine Abkürzungen sind.
5. Abkürzungen, die aus einer Verbindung aus Zahlen und Buchstaben bestehen.

Fehlerquellen in der Abkürzungserkennung für *false negatives*

1. Großbuchstaben innerhalb von Wörtern.
2. Die Verwendung von Ziffern und Bindestrichen.
3. Abkürzungen aus Kleinbuchstaben.

Fehlerquellen in der Ausrichtung für *false positives*

1. Verknüpfung des ersten Buchstabens der Abkürzung mit einem anderen Wort im Satz, das mit demselben Buchstaben beginnt.
2. Funktionswörter können als erstes Wort einer Definition erkannt werden.
3. Eingeklammerte Muster, die eine Abkürzung enthalten, die nicht im umliegenden Text definiert ist.

Fehlerquellen in der Ausrichtung für *false negatives*

1. Verwendung von Funktionswörtern, die nicht mit den Buchstaben in der Abkürzung verknüpft werden können.
2. Definitionen, die nur das Konzept der Abkürzung beschreiben.

2.2.2 Named-Entity-Recognition

Typischerweise erfolgt vor dem Entity Linking eine Erkennung und Kategorisierung benannter Entitäten, die *Named-Entity-Recognition (NER)* [21]. Der Vorverarbeitungsschritt identifiziert die Grenzen von benannten Entitäten. Das System lokalisiert und kategorisiert wichtige Substantive und Eigennamen im Text. NER ist ein Teilproblem der Informationsextraktion und bildet den Kern von NLP-Systemen zur Verarbeitung von natürlicher Sprache. NER umfasst zwei Aufgaben:

1. Identifizierung von Eigennamen im Text.
2. Klassifizierung von Substantiven und Eigennamen in eine Reihe vordefinierter Interessenskategorien.

Für weitere Informationen zu NLP-Anwendungen und einem konkreten NER-Tool siehe Stanford-NER [20].

2.3 Entity-Linking-System

Nachdem die Vorverarbeitungsschritte abgeschlossen sind, folgt das Entity Linking. In dieser Sektionen wird der modulare Aufbau klassischer Entity-Linking-Systeme aufgezeigt. Die Sektion spezifiziert die Aufgaben und Funktionsweisen der Module *Candidate-Entity-Generation, Candidate-Entity-Ranking und Unlinkable-Mention-Prediction.*

2.3.1 Modularer Aufbau von Entity-Linking-Systemen

Aufbauend auf der *Named-Entity-Recognition* folgt das Entity Linking. Das Entity-Linking-Problem lässt sich in drei Teilprobleme untergliedern. Jedes Teilproblem entspricht im System einem eigenen Modul. Das erste Modul ist die *Candidate-Entity-Generation*, das zum Herausfiltern irrelevanter Entitäten in der Wissensbasis dient. Das Modul erzeugt für jede Entitätserwähnung ein Kandidatenentitäten-Set E_m. Dieses Set enthält alle möglichen Entitäten der Wissensbasis, auf die sich die Entitätserwähnung möglicherweise bezieht. Im nächsten Schritt muss die Entität aus dem Set E_m identifiziert werden, die am wahrscheinlichsten der Erwähnung im Text entspricht. Dafür berechnet das *Candidate-Entity-Ranking* Modul für die Entitäten im Set E_m ein Ranking und identifiziert die Entität mit dem höchsten Rankingwert. In den meisten Fällen ist das Kandidatenentitäten-Set E_m nicht leer und eine Entität wird identifiziert. Daraus folgt jedoch nicht direkt, dass die Entitätserwähnung und die identifizierte Entität aus dem Set E_m übereinstimmen. Für diese Überprüfung ist das dritte Modul zur *Unlinkable-Mention-Prediction* zuständig. Das Modul validiert, ob für die Entitätserwähnung ein Link zur identifizierten Entität des *Candidate-Entity-Ranking* erstellt werden soll. In den nächsten Sektionen 2.3.2, 2.3.3 und 2.3.4 folgt eine detailliertere Erläuterung der einzelnen Module von Entity-Linking-Systemen.

2.3.2 Candidate-Entity-Generation

Das Modul für die *Candidate-Entity-Generation* filtert für eine Entitätserwähnung im Text alle irrelevanten Entitäten in der Wissensbasis heraus. Für jede Entitätserwähnung wird ein Kandidatenentitäten-Set erstellt. Dieses Set enthält alle Entitäten der Wissensbasis, auf die sich die Entitätserwähnung möglicherweise bezieht. Formal bezeichnen wir die Entitätserwähnung als m (abgeleitet aus dem englischen Wort „mention“) und das Kandidatenentitäten-Set als E_m (abgeleitet aus den englischen Wörtern „entity“ und „mention“). Das Modul berechnet die Funktion $f : M \rightarrow EM$, $m \rightarrow \mathrm{E}_m$ wobei M die Menge aller Entitätserwähnung darstellt und EM die Potenzmenge aller Entitäten in der Wissensbasis.

2.3.3 Candidate-Entity-Ranking

Das Modul für das *Candidate-Entity-Ranking* identifiziert eine Entität aus dem Set E_m, die am wahrscheinlichsten der Erwähnung im Text entspricht. Dazu berechnet das Modul für die Entitäten im Set E_m ein Ranking und identifiziert die Entität mit dem höchsten Wert. Das Modul berechnet die Rankingfunktion $rk : \mathrm{E}_m \rightarrow \mathbb{R}$, $e \rightarrow [0, \infty)$ bezüglich ausgewählter Merkmale, wobei $e \in \mathrm{E}_m$ eine Entität aus dem Kandidatenentitäten-Set ist.

In den meisten Fällen ist die Größe des Kandidatenentitäten-Sets E_m größer als eins. Das Ranking bewertet die Entitäten im Set E_m bezüglich einer Auswahl verschiedener Merkmale. Die Herausforderung in der Entwicklung des *Candidate-Entity-Ranking-Moduls* ist die Auswahl und Gewichtung der Merkmale für die Rankingfunktion rk. In der folgenden Übersicht wird eine Kategorisierung verschiedener Rankingmethoden vorgestellt.

Überwachte und nicht überwachte Rankingmethoden

Überwachte Rankingmethoden basieren auf annotierten Trainingsdaten. Das Modul lernt mit Hilfe der Trainingsdaten, wie die Kandidatenentitäten im Set E_m eingestuft werden sollen. Die verschiedenen Ansätze verwenden unter anderem binäre Klassifikationsverfahren (vrgl. [38]), Learning-to-Rank-Verfahren (vrgl. [28]), probabilistische Verfahren (vrgl. [17]) oder Graph-Verfahren (vrgl. [11]).

Systeme mit nicht überwachten Rankingmethoden [13, 7] sind unabhängig von annotierten Trainingsdaten.

Unabhängige und kollektive Rankingmethoden

Unabhängige Rankingmethoden, wie in den Systemen von Lin et al. [19] und von Zhang et al. [38] basieren auf der Annahme, dass Entitätserwähnungen unabhängig sind. Das System identifiziert keine Beziehungen zwischen den Entitätserwähnungen in einem Dokument, um das Ranking der Kandidatenentitäten zu unterstützen. Unabhängige Rankingmethoden berechnen die Ähnlichkeit zwischen dem Kontext der Entitätserwähnung und den zusätzlichen Informationen der Kandidatenentität.

Kollektive Rankingmethoden, wie in den Systemen von Guo et al. [9] und von Cucerzan [7] arbeiten mit der Annahme, dass ein Dokument auf zusammenhängende Entitäten aus wenigen und verwandten Themen verweist. Die Verlinkungen früherer Entitätserwähnungen werden berücksichtigt und beeinflussen das Ranking der Kandidatenentitäten nachfolgender Entitätserwähnungen.

2.3.4 Unlinkable-Mention-Prediction

Nicht jede Entitätserwähnung entspricht einer Entität in der Wissensbasis. Trotzdem ist in den meisten Fällen das Kandidatenentitäten-Set E_m für eine Entitätserwähnung m nicht leer. Eine durch das *Candidate-Entity-Ranking* identifizierte Entität muss demnach nicht der zugehörigen Entitätserwähnung entsprechen. Für diesen Fall ist das dritte Modul zur Vorhersage nicht verknüpfbarer Entitätserwähnungen (*Unlinkable-Mention-Prediction*) zuständig. Das Modul validiert, ob für die Entitätserwähnung ein Link zur identifizierten Entität des *Candidate-Entity-Ranking* erstellt werden soll. Für die Vorhersage gibt es verschiedene Ansätze. Die einfachsten Ansätze basieren auf heuristischen Methoden. Beispielsweise kann eine Entitätserwähnung m als unverknüpfbar identifiziert werden, wenn das Kandidatenentitäten-Set E_m leer ist. Dieser Ansatz wird in den Systemen von Chen et al. [4] und Varma et al. [33] implementiert. Schwellenwertverfahren sind eine weitere Möglichkeit, nicht verknüpfbare Entitätserwähnung vorherzusagen.

Die Vorhersage nicht verknüpfbarer Entitätserwähnungen muss nicht zwingend ein separates Modul sein. In vielen Systemen ist der Vorhersageprozess in den *Candidate-Entity-Ranking-Prozess* integriert [9, 29]. Im Anschluss an die Vorhersage muss für ausgewählte Entitätserwähnungen ein Link zu einer Entität in der Wissensbasis erstellt werden. Die Erstellung eines Links kann in diesem Modul integriert sein oder separat erfolgen.

3 Verwandte Arbeiten

In Kapitel 2 wurden bereits verschiedene Module und Ansätze einzelner Entity-Linking-Systeme erläutert. Dieses Kapitel gibt nun einen Überblick über verschiedene Systeme und Arbeiten im Forschungsbereich des Entity Linking.

Die existierenden Systeme könne in folgende Kategorien unterteilt werden:

General-Purpose-Systeme

General-Purpose-Systeme haben den Anspruch, Entitäten in beliebigen Textdokumenten zu identifizieren und zu einer Wissensbasis zu verlinken.

Das Linden-System [28] verknüpft benannte Entitäten im Text mit der Wissensbasis Yago [24]. Das System von Lin et al. [19] untersucht Entitätsverknüpfung von 500 Millionen Webdokumenten für die Erstellung einer allgemeinen Wissensbasis. Das NEREL-System [29] verbindet die Aufgaben NER und Entity Linking in einem Modul.

Domänenspezifische Systeme

Domänenspezifische Systeme sind entweder auf eine inhaltliche Domäne spezialisiert oder auf Textdokumente bestimmter Formaten in einem bestimmtem Kontext.

Das System von Guo et al. [9] ist auf Entity Linking für Tweets spezialisiert. Das System von Zhang et al. [37] ist auf domänenspezifisches Entity Linking spezialisiert. Das System modelliert die *Named-Entity-Detection (NED)* und das Entity Linking gemeinsam für eine beliebige domänenspezifische Wissensbasis. Dabei identifiziert das System Entitätserwähnungen, die nicht aus der festgelegten Domäne sind *(Fake-Named-Entities)*. Im Gegensatz zu den Entitätserwähnungen der festgelegten Domäne können *Fake-Named-Entities* anschließend ignoriert oder separat verarbeitet werden. Die Aufgaben NED und Entity Linking sind iterativ verknüpft und verbessern damit iterativ die Konfidenz der NED und die Gewissheit der Entitätsverknüpfungen.

3.1 Vergleichbarkeit verschiedener Entity-Linking-Systeme

Die existierenden Entity-Linking-Systeme sind nur schwer vergleichbar. Die Verwendung verschiedener Datensätze verhindert eine direkte Vergleichbarkeit zwischen zwei Systemen. Es existiert kein allgemein akzeptierter Gold-Standard, da die Systemanforderungen und die Eigenschaften der Datensätze sehr unterschiedlich sind. Die Systeme unterscheiden sich in mehreren Dimensionen, sodass einzelne Entwurfsentscheidungen der Module schwer isoliert evaluiert werden können. Verschiedene Ansätze für die einzelnen Module wurden bereits in Kapitel 2 vorgestellt. Die entwickelten Entity-Linking-Systeme verhalten sich sehr unterschiedlich für verschiedene Datensätze und Domänen. In der wissenschaftlichen

Veröffentlichung von Shen et al. [27] wird außerdem angemerkt, dass es unwahrscheinlich ist, dass eine Technik alle anderen über alle Datensätze hinweg dominiert.

3.2 Word-Sense-Disambiguation

Die *Word-Sense-Disambiguation (WSD)* beschreibt die Identifikation des gewollten Sinns eines Wortes in seinem Kontext. Das WSD-Problem resultiert aus der Existenz von Homonymen in der englischen Sprache. Ein Homonym beschreibt ein Wort, das für verschiedene Begriffe steht. Beispielsweise kann das Wort *bank* ein Kreditinstitut referenzieren oder eine Parkbank. Das Kreditinstitut und die Parkbank teilen sich dieselbe Form, aber nicht die Bedeutung. Da ein Homonym mehrere Bedeutungen haben kann, muss die gewollte Bedeutung identifiziert werden. Bei der menschlichen Kommunikation passiert die Disambiguierung in der Regel automatisch und unbewusst. In der Computerlinguistik ist eine gezielte Disambiguierung schwierig und erfordert zusätzliches Wissen über den Kontext der Worterwähnung und die verschiedenen Bedeutungen des Wortes.

In der Domäne der Softwareentwicklung sind Homonyme enthalten. Beispielsweise kann das englische Wort *model* auf das Modell eines trainierten *Machine Learning* Algorithmus verweisen oder auf die Darstellung eines realen Prozesses oder Konzepts. Somit können auch die Entitäten einer Architekturdokumentation Homonyme sein. Aus diesem Grund ist es notwendig, die WSD im Entity-Linking-Prozess zu berücksichtigen. Das System von Bevilacqua et al. [3] verwendet Informationen eines Wissensgraphen für das WSD-Problem. Die wissenschaftliche Arbeit von Navigli [23] gibt einen Überblick über überwachte, unbeaufsichtigte und wissensbasierte Ansätze im Forschungsbereich der WSD.

4 Ansatz

Das Ziel dieser Ausarbeitung ist die Entwicklung eines Entity-Linking-Systems für Softwarearchitekturdokumentationen. Voraussetzungen für das Entity-Linking-System sind eine Wissensbasis für die Domäne der Softwareentwicklung und eine Vorverarbeitung des zu analysierenden Textes. In diesem Kapitel werden die Aufgaben des Systems in die verschiedene Module untergliedert. Die Aufteilung und Ausführungsreihenfolge der Module sind in Abbildung 4.1 graphisch illustriert.

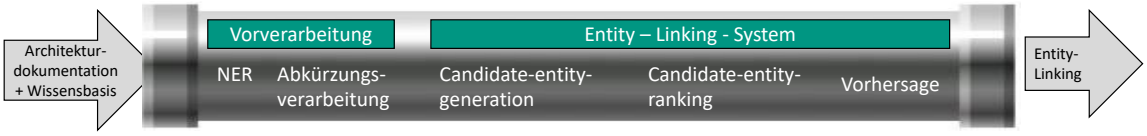

Abbildung 4.1: Ausführungsreihenfolge der Module

4.1 Module

Modul Wissensbasis

Im Entity-Linking-Prozess sollen Entitätserwähnungen im Text mit den Entitäten in einer Wissensbasis verlinkt werden. Für die Domäne der Softwareentwicklung gibt es noch keine Wissensbasis, die über Programmcode angefragt werden kann und eine große Abdeckung von Begriffen und Abkürzungen mit Beziehungen zwischen ihnen bereitstellt. Darum ist der erste Schritt dieser Arbeit, die Erstellung einer Wissensbasis bezüglich der Domäne der Softwareentwicklung. Für den Aufbau der Wissensbasis mit Fachbegriffen und Abkürzungen werden Quellen benötigt. Aus diesen Quellen werden die Daten extrahiert und aufbereitet. Anschließend wird die Wissensbasis erstellt und mit den aufbereiteten Daten popularisiert. Details zum Vorgehen bei der Erstellung der Wissensbasis befinden sich in Kapitel 5.

Modul Vorverarbeitung mit Named-Entity-Recognition und Abkürzungsverarbeitung

Die Vorverarbeitung wird ausgeführt bevor der Entity-Linking-Prozess beginnt. Das Entity-Linking-System ist auf die Ergebnisse der NER angewiesen. Die Abkürzungsverarbeitung ergänzt die NER in der Vorverarbeitung. Das Entity-Linking-System beginnt, sobald die Vorverarbeitung beendet ist. Die Vorverarbeitung untergliedert sich in zwei verschiedene Aufgaben.

Der Vorverarbeitungsschritt zur *Named-Entity-Recognition* ist die Hauptaufgabe des Vorverarbeitungsmoduls. Das NER-System verarbeitet natürlichsprachlichen Text und identifiziert die Grenzen von benannten Entitäten. Für die Lokalisation der Entitäten werden wichtige Substantive, Eigennamen und Fachbegriffe im Text erkannt und kategorisiert. Details bezüglich der *Named-Entity-Recognition* befinden sich in Kapitel 6.

Die Abkürzungsverarbeitung ist ein Vorverarbeitungsschritt zur Erkennung und Auflösung von selbstdefinierten Abkürzungen. Eine selbstdefinierten Abkürzung im Text kann zum Beispiel wie folgt definiert werden: *The Access Control List (ACL)*. Für die Abkürzungsverarbeitung müssen Abkürzungen zunächst erkannt werden und anschließend an den Präfixen der Definition ausgerichtet werden. Falls es mehrere mögliche Definitionen der Abkürzung gibt, muss die beste ausgewählt und gespeichert werden. Weitere Vorkommen derselben Abkürzung müssen im weiteren Textverlauf erkannt und aufgelöst werden. Details bezüglich der Abkürzungsverarbeitung sind in Kapitel 7 zu finden.

Modul *Candidate-Entity-Generation*

Das Entity-Linking-System untergliedert sich in drei Module. Das erste Modul ist die *Candidate-Entity-Generation* zum Finden passender Entitäten in der Wissensbasis. Dafür verwendet das System *Name-Dictionary-Based-Techniques*, um für jede Entitätserwähnung ein Kandidatenentitäten-Set zu generieren. Details bezüglich der Generierung der Kandidatenentitäten-Sets befinden sich in Kapitel 8.

Modul *Candidate-Entity-Ranking*

Anschließend werden die Entitäten in dem Kandidatenentitäten-Set mit einer Rankingfunktion bewertet. Die Rankingfunktion basiert auf einer Kombination aus kontextunabhängigen und kontextabhängigen Merkmalen. Die Merkmale werden so ausgewählt und gewichtet, dass der Rangkingwert die Ähnlichkeit zwischen Entitätserwähnung und Kandidatenentität repräsentiert. Weitere Details zum Vorgehen und den spezifischen Merkmalen der Rankingfunktion befinden sich in Kapitel 9.

Modul *Unlinkable-Mention-Prediction*

Im abschließenden Modul werden die Entitätserwähnungen ausgewählt, die zu einer Entität in der Wissensbasis verlinkt werden sollen. Dafür wird eine Prioritätsliste für jeden Satz erstellt, die alle Entitätserwähnungen enthält, die zu einer Entität verbunden werden sollen. Für die Entitätserwähnungen in der Prioritätsliste wird ein Schwellenwertverfahren verwendet, um die Verbindung zu verifizieren. Weitere Details zur *Unlinkable-Mention-Prediction* befinden sich in Kapitel 10.

4.2 Laufendes Beispiel

Für einen Beispielablauf des Entity-Linking-Systems besteht die Architekturdokumentation nur aus einem Satz: *The service component communicates with the server.* Numerische Werte und Mengen sind im kommenden Beispiel an echten Daten orientiert, entsprechen jedoch

aus Gründen der Übersicht, nicht den exakten Ergebnissen des Systems. Das Beispiel wird in der gesamten Arbeit an verschiedenen Stellen wiederverwendet.

Die Architekturdokumentation wird dem System übergeben und die Vorverarbeitung wird ausgeführt. In der Abkürzungsverarbeitung wird keine Abkürzung im Satz erkannt. Das Ergebnis der *Named-Entity-Recognition* ist in Abbildung 4.2 und Abbildung 4.3 illustriert.

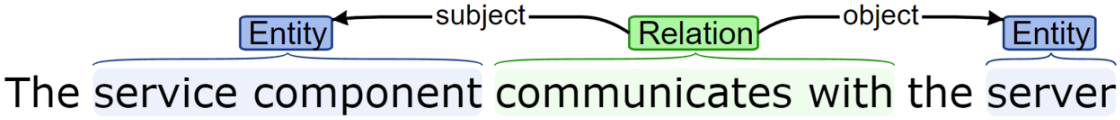

Abbildung 4.2: NER-OpenIE-Analyse

Abbildung 4.3: NER-POS-Analyse

Aus dem Ergebnis der NER werden die Entitätserwähnungen extrahiert. Für den Beispielsatz ergeben sich folgende Entitätserwähnungen:

$$\{\text{service component, service, component, server}\}. \tag{4.1}$$

Das Ergebnis der NER wird an die Module des Entity-Linking-Systems weitergegeben. Im Entity-Linking-System erfolgt für jede Entitätserwähnungen die Generierung des Kandidatenentitäten-Sets E_m. Für die Erstellung der Kandidatenentitäten-Sets wird die Wissensbasis verwendet. Die Entitäten der Wissensbasis sind in Abbildung 4.4 dargestellt. Mit den Entitäten der Wissensbasis werden folgende Kandidatenentitäten-Sets generiert:

$$\begin{aligned} &E_{\text{component}} = \{\text{component}\}, \quad E_{\text{service component}} = \{\text{service component}\}, \\ &E_{\text{server}} = \{\text{server}_1, \text{server}_2\}, \quad E_{\text{service}} = \{\text{service}\} \end{aligned} \tag{4.2}$$

Die Kandidatenentitäten-Sets werden an das Modul des *Candiate-Entity-Ranking* übergeben. Das Modul berechnet für jede Kandidatenentität im Kandidatenentitäten-Set ein Rankingwert.

$$\begin{aligned} &\text{Entitätserwähnung } \textit{service component}\text{: } rk(\text{service component}) = 2.1 \\ &\text{Entitätserwähnung } \textit{component}\text{: } rk(\text{component}) = 2.3 \\ &\text{Entitätserwähnung } \textit{service}\text{: } rk(\text{service}) = 2.2 \\ &\text{Entitätserwähnung } \textit{server}\text{: } rk(\text{server}_1) = 1.60; \quad rk(\text{server}_2) = 2.4 \end{aligned} \tag{4.3}$$

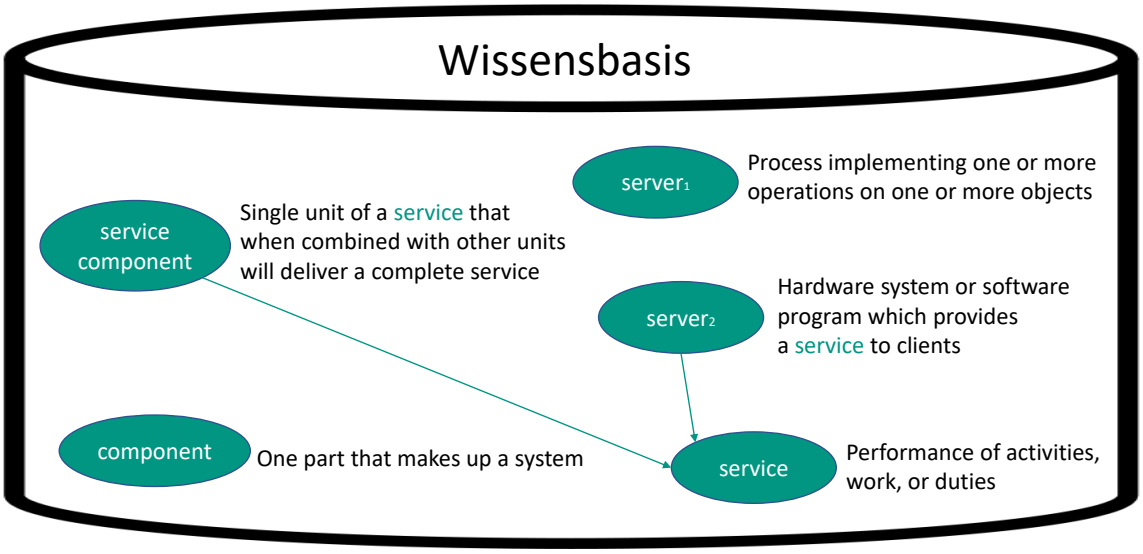

Abbildung 4.4: Beispiel-Wissensbasis

In diesem Satz hat die Kandidatenentität $server_2$ einen höheren Rankingwert für die Entitätserwähnung *server* als die Kandidatenentität $server_1$. Aus diesem Grund wird die Kandidatenentität $server_2$ für die Entitätserwähnung *server* ausgewählt.

Abschließend wird in der *Unlinkable-Mention-Prediction* ausgewählt, welche Entitätserwähnungen zu einer Entität in der Wissensbasis verlinkt werden sollen. Die ausgewählten Entitätserwähnungen werden in eine Prioritätsliste hinzugefügt. In dem vorliegenden Beispielsatz wird die folgende Prioritätsliste erstellt:

$$\text{Prioritätsliste} = \{\text{service component, server}\} \tag{4.4}$$

Dabei fällt auf, dass die Entitätserwähnung *service component* die Entitätserwähnungen *service* und *component* überdeckt.

5 Aufbau der Wissensbasis

Im Entity-Linking-Prozess sollen Entitätserwähnungen im Text mit den Entitäten in einer Wissensbasis verlinkt werden. Die Wissensbasis muss Begriffe, Abkürzungen und Beziehungen zwischen ihnen enthalten. Für die Domäne der Softwareentwicklung gibt es noch keine Wissensbasis mit ausreichend großer Abdeckung, die den Ansprüchen dieser Arbeit entspricht.

Im folgenden Kapitel wird die Erstellung und die anschließende Verwendung der Wissensbasis im Rahmen dieser Arbeit erläutert. Dazu werden in Sektion 5.1 die Eigenschaften der erstellten Wissensbasis aufgezeigt. Darauf folgt in den Sektionen 5.2, 5.3 und 5.4 die Vorgehensweise zur Erstellung der Wissensbasis. In Sektion 5.2 wird dabei zunächst auf die möglichen Quellen für die Wissensbasis eingegangen. In Sektion 5.3 folgt eine Analyse der ausgewählten Quelle. Die Datenaufbereitung und Extraktion wird in Sektion 5.4 erläutert. Abschließend enthält das Kapitel Informationen zur Population und dem Zugriff auf die Wissensbasis in Sektion 5.5.

5.1 Eigenschaften

Die erstellte Wissensbasis wird von dem Entity-Linking-System verwendet. Dementsprechend sind die Eigenschaften der Wissensbasis so ausgelegt, dass die Anfragen des Systems bestmöglich beantwortet werden können. Die Eigenschaften der erstellten Wissensbasis werden im Folgenden näher erläutert.

Datenbanktyp

Die Daten, die unsere Wissensbasis enthalten soll, sind stark vernetzt. Daher passt eine Datenspeicherung mit Graph-basierten Ansatz gut zu unserer Datenstruktur. Aus diesem Grund haben wir uns für die native Graph-basierte Datenbank Neo4j entschieden. Neo4j wird auf der eigenen Website als eine *open-soure* JVM-basierte NOSQL-Datenbank beschrieben. Bei stark vernetzten Daten, wie bei den Entitäten, ist Neo4j schneller und intuitiver als relationale Datenbanken. Die Performance von Neo4j wurde in dem Guide *Neo4j in action* von Vukotic et al. [34] getestet. Neo4j erreicht bei Durchläufen über Diagrammdaten eines sozialen Netzwerks bei einer Anfragetiefe von drei Schritten die vierfache Geschwindigkeit im Gegensatz zur MySQL Anfrage. Je mehr Schritte eine Anfrage zulässt bzw. je tiefer eine Anfrage die Diagrammdaten des sozialen Netzwerks durchlaufen soll, desto besser ist die Performance von Neo4j im Vergleich zu MySQL Anfragen. Die Vernetzung der Daten muss nicht mithilfe von Fremdschlüsseln oder *Out-of-Band*-Verarbeitung wie *MapReduce* abgeleitet werden, sondern kann direkt als Graph repräsentiert und gespeichert werden.

Relationale Datenbanken können ebenfalls Beziehungen speichern, dazu werden teure *JOIN-Operationen* oder *Cross-Lookups* verwendet, die oft an ein starres Schema gebunden sind. In einer Graphdatenbank gibt es keine *JOINs* oder *Lookups*. Beziehungen werden nativ neben den Datenelementen in einem flexiblen Format gespeichert. Graphdatenbanken sind somit besser geeignet, um die Beziehungen der Datenelemente zu speichern und zu verarbeiten. Weitere Details bezüglich Abfragen der Struktur von Graphdatenbanken und konkrete Werte der Performance befinden sich im Neo4j Guide [34]. Beziehungsabfragen haben für unsere Daten somit eine intuitive Struktur bei der Verwendung von Graphdatenbanken und können einfach im Programmcode formuliert werden. Durch den nativen Graphspeicher hat Neo4j eine skalierbare und geschwindigkeitsoptimierte Architektur mit *ACID (atomicity, consistency, isolation, durability)* Konformität. Ein weiterer Vorteil des Graph-basierten Ansatzes ist die intuitive Visualisierung der Daten als Graph. Zur tabellarischen oder graphischen Ausgabe der Dateneinträge eignet sich der integrierte Neo4j-Brower.

Datenbankschema

In der Wissensbasis gibt es Knoten und Beziehungen. Die Knoten werden in Entitäten und Abkürzungen unterteilt. Es gibt zudem drei verschiedene Beziehungtypen.

1. **Knoten:** Entität (entity)
   Attribute: Identifikator (id), Name (name), Beschreibung (description)
2. **Knoten:** Abkürzung (abbreviation)
   Attribute: Identifikator (id), Name (name), Beschreibung (description)
3. **Beziehung:** ICT (*Is Connected To*)
   Entität → Entität, e1 —[ICT]→ e2.
   Verbindet die Entität e1 zur Entität e2, wenn die Beschreibung von e1 (e1.description) auf den Entitätsnamen von e2 (e2.name) verweißt.
4. **Beziehung:** SYN (*Synonym*)
   Entität → Entität, e1 —[SYN]→ e2.
   Verbindet die Entität e1 mit der Entität e2, wenn e2 ein Synonym von e1 ist.
5. **Beziehung:** ISF (*Is Short For*)
   Abkürzung → Entität, a —[ISF]→ e.
   Verbindet die Abkürzung a zur Entität e, wenn a eine Abkürzung für e ist. Dies kann entweder durch die Definition innerhalb des Dokuments hervor gehen oder anhand der Beschreibung von Abkürzung a (a.description).

Die Beziehungen in der Wissensbasis beschreiben inhaltliche Zusammenhänge zwischen Entitäten und Abkürzungen. Dazu betrachten wir die Beispielentität $server_2$ aus dem laufenden Beispiel. Für die Entität $server_2$ soll in der Wissensbasis eine Verknüpfung zur Entität *service* enthalten sein ($server_2$ —[ICT]→ *service*). Die Entitäten und Verknüpfungen der Wissensbasis sind in Abbildung 5.1 repräsentiert.

Domänenspezifität

Die Wissensbasis enthält nur Entitäten aus der Domäne der Softwareentwicklung oder inhaltlich angrenzenden Domänen wie z.B. die Informationstechnik oder Informatik. Als Beispiel wird der Satz des laufenden Beispiels mit der Entität *server*

betrachtet. Der Satz, die Entitäten des Satzes und ein Ausschnitt der Wissensbasis sind in Abbildung 5.1 aufgezeigt.

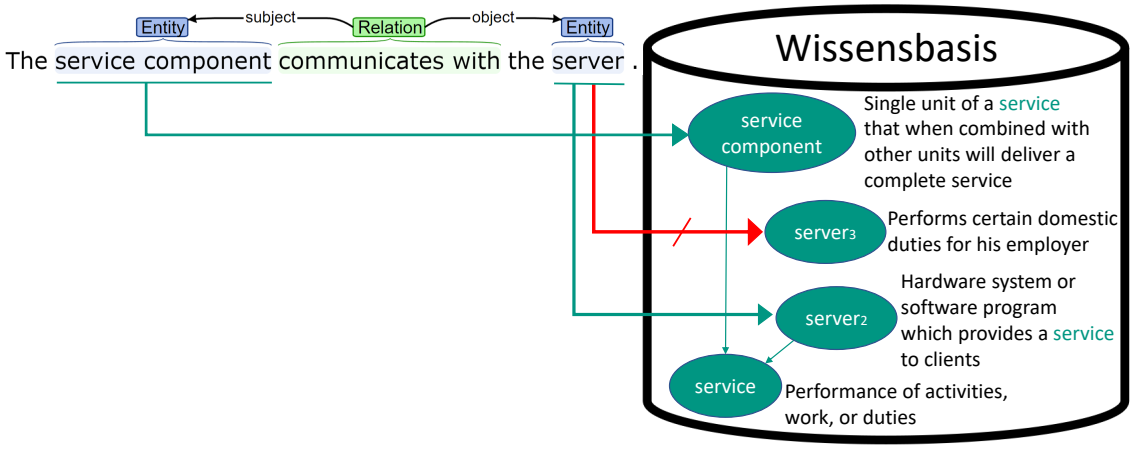

Abbildung 5.1: Beispielsatz mit der Entität *server*.

Die Entität $server_2$ hat für die Domäne der Softwareentwicklung den folgenden Beschreibungstext: *Hardware system or software program which provides a service to clients*. Eine andere Beschreibung eines *servers* ist die Definition der Entität $server_3$: *performs certain domestic duties for his employer*. Diese Definition beschreibt einen *server* als Diener bzw. Bediensteter. In dem Beispielsatz der Softwarearchitekturdokumentation passt die Definition der Entität $server_2$ besser zu der Entitätserwähnung *server*. In der Wissensbasis sind in diesem Beispiel jedoch beide Entitäten inklusive der Beschreibungstexte für die Oberflächenform *server* enthalten. Um die Entitätserwähnung *server* zu einer der *server* Entitäten zuordnen zu können, muss diese von der anderen *server* Entitäten unterschieden und ausgewählt werden. Um eine falsche Auswahl und zusätzlichen Arbeitsaufwand zu vermeiden, ist es sinnvoll, nur potenzielle Entitätskandidaten im Rahmen der Wissensbasis zu berücksichtigen. Für die Domäne der Softwareentwicklung ist die Entität $server_2$ sinnvoll, die Entität $server_3$ sollte jedoch nicht in der Wissensbasis enthalten sein. Insgesamt soll die Wissensbasis also domänenspeziefisch sein und nur Entitäten der festgelegten Domäne enthalten.

5.2 Verfügbare Quellen

Das Entity-Linking-System analysiert natürlichsprachliche Texte zur Dokumentation der Softwarearchitektur. Die zugehörige Domäne ist die Softwareentwicklung. Es gibt viele angrenzende und überlappende Domänen wie z.B. die Informationstechnik oder die Informatik. Die Zuteilung einzelner Entitäten zu einer Domäne ist oft nicht eindeutig, weshalb auch nicht eindeutig ist, welche Entitäten die Wissensbasis bezüglich der Domäne enthalten soll. Im Folgenden sind drei Listen an Begriffen gegeben, die bezüglich der

Domäne der Softwareentwicklung und inhaltlich angrenzenden Domänen Vokabular auflisten und beschreiben.

1. Glossary of computer science - Wikipedia
2. List of computing and IT abbreviations - Wikipedia
3. ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary [14]

Diese Listen wurden ausgewählt, da sie zu jeder Entität eine Beschreibung enthalten und zu Abkürzungen die zugehörige Langform. Außerdem sind die Quellen umfangreich und decken ein breites Spektrum der Begriffe in der Domäne der Softwareentwicklung ab. Die Quellen haben außerdem jeweils ein genormtes Format, das eine automatische Informationsextraktrion erleichtert. Die Quellen 1 und 2 sind beide von Wikipedia und können in Kombination verwendet werden. Quelle 1 enthält Fachbegriffe aus der Informatik inklusive der Begriffsbeschreibungen. Quelle 2 enthält ergänzend die Abkürzungen und zugehörigen Langformen der Fachbegriffe. Quelle 3 kombiniert die Vorzüge der ersten beiden Quellen indem sowohl Fachbegriffe mit Begriffsbeschreibungen als auch Abkürzungen mit der zugehörigen Langform in einem Dokument vereint sind. Außerdem enthält Quelle 3 bereits Listen von Synonymen und Listen verwandter Entitäten, die einfach extrahiert und in die Wissensbasis übernommen werden können. Aufgrund der genannten Vorteile gegenüber den Informationsquellen von Wikipedia haben wir uns in dieser Arbeit für die Zusammenstellung in Quelle 3 entschieden [14]. Extrahierte Informationen aus der Quelle werden in einer Datenbank gespeichert und anschließend vom Entity-Linking-System verwendet.

Die ausgewählte Quelle enthält für jeden Fachbegriff mindestens eine Begriffsbeschreibung. Durch die Beschreibungen ist es möglich, Verbindungen zwischen Begriffen zu identifizieren und damit eine Wissensbasis aufzubauen. Dazu müssen die Beschreibungen analysiert und auf Entitätserwähnungen überprüft werden. Auch Quellen ohne Begriffsbeschreibung können verwendet werden, erfordern jedoch eine händische Erstellung der fehlenden Informationen.

Ein alternativer Ansatz wäre die Verwendung bereits existierender Wissensbasen. Die klassischen Wissensbasen DBpedia [1] und YAGO [24] sind nicht domänenspezifisch. Es ist möglich diese Wissensbasen zu verwenden und auf die Domäne Softwareentwicklung zuzuschneiden. Dieses Vorgehen ist weniger aufwändig, hat aber einen entscheidenden Nachteil: Die Begriffsbeschreibungen und Daten sind nicht speziell auf den Kontext unsere Domäne ausgelegt. Es gibt in Yago abgeleitete Kategorien aus WordNet wie z.B. „Musik“, „Wissenschaft“ oder andere Domänen. Diese Kategorisierung ist aber nicht vergleichbar zu einer händisch evaluierten Zusammenstellung aus Entitäten und Begriffsdefinitionen wie im IEEE 24765:2017 [14]. Aus diesem Grund bringen existierende Wissensbasen keine Verbesserung und werden nicht im Rahmen dieser Arbeit verwendet.

5.3 Eigenschaften der Quelle

Für die Erstellung der Wissensbasis werden Daten benötigt. Eine umfangreiche Zusammenstellung aus Begriffen mit Begriffsbeschreibungen und einer einheitlichen Struktur bietet

das ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary [14]. Das Dokument wurde erstellt, um Terminologie zu sammeln und zu standardisieren. Somit ist das Dokument eine Zusammenstellung der verwendeten Begriffe und Standarddefinitionen aus dem Bereich System- und Softwareentwicklung. Es soll als nützliches Nachschlagewerk für Fachleute auf dem Gebiet der Informationstechnologie dienen und die Verwendung von System- und Software-Entwicklung-Standards fördern.

Die Quelle bietet Definitionen, die streng, unkompliziert und möglichst verständlich sind. Das Dokument ermöglicht nach verwandten Konzepten zu suchen und zu überprüfen, wie ein Begriff in den Definitionen anderer Begriffe verwendet wird.

Das Vokabular wird sowohl in gedruckter als auch in einer über das Internet zugänglichen Version angeboten. Im Rahmen dieser Arbeit wurde die online verfügbare Version des ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary [14] verwendet, um die Informationsextraktion automatisiert vornehmen zu können.

Das Dokument listet nach jedem Begriff nummerierte Definitionen auf. Die Definitionen sind in der Reihenfolge ihrer Präferenz aufgelistet und enthalten allgemeinere Definitionen vor spezifischeren Definitionen. Abkürzungen können sowohl separat als auch in Klammern hinter dem Quellbegriff aufgeführt werden. Außerdem können die verschiedenen Definitionen die Verwendung eines Begriffs als Substantiv, Verb oder Adjektiv zeigen.

5.4 Datenaufbereitung und Informationsextraktion

Im nächsten Schritt fand die Datenaufbereitung statt. Dafür wurden die benötigten Daten für die Wissensbasis zunächst in einer Textdatei gespeichert und mithilfe von regulären Ausdrücken entsprechend der Informationsbedürfnisse aufbereitet. Die vorhandenen Quellenangaben innerhalb des Dokuments werden im Rahmen der Wissensbasis nicht benötigt, sodass diese aus den Begriffsdefinitionen entfernt wurden. Die Begriffsdefinitionen sind in der Reihenfolge ihrer Präferenz gegeben. Die Wissensbasis ist für eine Entität auf maximal drei Begriffsdefinitionen limitiert, um nicht relevante und selten verwendete Definitionen zu vermeiden. Darum wurden alle darüber hinausgehenden Begriffsdefinitionen aus der Quelle entfernt. Zudem wurde allgemeines Rauschen wie z.B. der Dokument-Header entfernt, sodass nur die Begriffe, Abkürzungen und Synonyme inklusive der Begriffsdefintionen erhalten bleiben.

Für die Informationsextraktion wurde das überarbeitete Dokument eingelesen, die enthaltenen Daten aufbereitet und anschließend in der Wissensbasis persistiert. Dieser Vorgang kann im Falle einer Aktualisierung der Quelle wiederverwendet werden, unter der Voraussetzung, dass das neue Dokument demselben Definitionsschema folgt. Somit ist es möglich, die erstellte Wissensbasis zu aktualisieren und bei Weiterentwicklungen im Bereich der Domäne, die Wissensbasis (ohne großen Aufwand) anzupassen. Die Informationsextraktion wurde nach folgendem Schema durchgeführt:

Abkürzungen verbinden

Die Verbindung der Abkürzungen kann mithilfe von zwei Methoden umgesetzt werden. Anhand der Struktur des Dokuments ist es möglich, nach Abkürzungsvorkommen innerhalb von Entitätsnamen zu suchen, die als Klammerausdrücke

hinter dem Namen stehen z.B. *start-to-start (SS)*. Als zweite Möglichkeit kann in der Beschreibung von Abkürzungen der Name der Entität gesucht werden, auf die die Abkürzung verweist wie z.B. für die Abkürzung *BM*: *BM. (1) business manager*. Eine Abkürzungsbeschreibung enthält im Gegensatz zu Entitäten keine inhaltliche Beschreibung der Abkürzung, sondern nur die zugehörige Langform und eine Quelle. Die extrahierte Langform kann anschließend mit Entitätsnamen verglichen werden, um die zugehörige Entität zu identifizieren.

Beide Möglichkeiten enthalten viele Abkürzungszuweisungen und auch die Schnittmenge gleicher Zuweisungen zwischen den beiden Möglichkeiten ist sehr groß. Insgesamt müssen jedoch trotzdem beide Möglichkeiten verwendet werden, um möglichst viele und bestenfalls alle Abkürzungen zu ihren Entitäten zu verbinden. Die Verbindung der Abkürzungen zu ihren zugehörigen Entitäten ist sehr wichtig, denn isolierte Abkürzungen ohne verlinkte Entität, sind weniger hilfreich in der Wissensbasis und in den meisten Fällen unbrauchbar.

1. Für den ersten Ansatz müssen die Entitätsnamen nach einem Klammermuster durchsucht werden. Anschließend extrahiert unser System die Abkürzungsnamen in den Klammern.
2. Der zweite Ansatz erfordert eine Berechnung der Übereinstimmung zwischen Abkürzungbeschreibung und Entitätsnamen mit einem anschließenden *alignment*. Zunächst wird der *Longest Common Substring* mit Hilfe von dynamischer Programmierung zwischen dem Entitätsnamen und der Abkürzungsbeschreibung bestimmt. Die Abkürzungsbeschreibung sollte in diesem Schritt ausschließlich aus der Langform bestehen und keine zusätzlichen Quellen oder anderes Rauschen enthalten. Unser System berechnet mit Hilfe des *Longest Common Substrings* die folgenden Längenverhältnisse und mittelt diese zu einem Verhältniswert (*totalRatio*).

$$\begin{aligned} \text{nameRatio} &= \frac{\text{Longest Common Substring Length}}{\text{Entity Name Length}} \\ \text{descriptionRatio} &= \frac{\text{Longest Common Substring Length}}{\text{Description Length}} \\ \text{totalRatio} &= \frac{(\text{nameRatio} + \text{descriptionRatio})}{2} \end{aligned} \tag{5.1}$$

   Das Ergebnis wird anschließend mit einem Schwellenwert σ verglichen. Für den Schwellenwert gilt $\sigma \in [0, 1]$. Je niedriger der Schwellenwert, desto mehr Verhältnisse werden akzeptiert. Nach händischer Evaluation hat sich ein Schwellenwert von $\sigma = 0.8$ als sinnvoll erwiesen. In den Abkürzungsbeschreibungen ist wenig Rauschen und somit kann der Wert von *sigma* verhältnismäßig hoch gewählt werden. Wenn die Abkürzungsbeschreibung viel Rauschen enthält, muss der Schwellenwert entsprechend niedriger gewählt oder die *descriptionRatio* weggelassen werden. Wenn die Kombination aus einer Abkürzungsbeschreibung und einem Entitätsnamen überhalb des Schwellenwertes liegt, wird diese Kombination näher analysiert. Dazu wird der Name der Abkürzung an dem

Entitätsnamen ausgerichtet. Hierbei wird versucht, die einzelnen Buchstaben der Abkürzung an den Präfixen der Wörter des Entitätsnamen auszurichten. Dieser Vorgang wird im englischen als *alignment* bezeichnet und findet häufig Gebrauch in Abkürzungsverarbeitungssystemen.

Mit der zusätzlichen Einschränkung über die Ausrichtung werden *false positives* vermieden. Die Notwendigkeit einer gültigen Ausrichtung verhindert aber auch, dass Abkürzungen mit logischen Abkürzungskürzeln akzeptiert werden. Zum Beispiel wird die Abkürzung H2O für Wasser nicht akzeptiert. Dies ist jedoch kein Problem, da davon ausgegangen werden kann, dass gültige Abkürzungen ohne eine gültige Ausrichtung bereits explizit im Dokument durch ein Klammermuster definiert sind und somit eine Verlinkung im System haben. Wenn eine Ausrichtung gefunden wurde, wird die Verlinkung akzeptiert, anderenfalls wird die Verlinkung abgelehnt.

Entitäten verbinden

Sobald die Abkürzungen zu den Entitäten verlinkt worden sind, werden die Entitäten betrachtet. Die Quelle bietet für verwandte Entitäten eine Erkennung im Beschreibungtext (*See also:* <List> Entities). Diese Listen enthalten eine schriftliche Auflistung inhaltlich verwandter Entitäten. Nicht jede Entität hat eine solche Liste und in den meisten Fällen ist in der Liste nur eine Entität enthalten. Aus diesem Grund müssen noch weitere Verlinkungen zwischen Entitäten mit anderen Techniken identifiziert werden. Dafür analysieren wir die Entitätsbeschreibungen und identifizieren daraus weitere Verlinkungen. Wenn eine Entitätsbeschreibung (e_1.description) den Namen einer anderen Entität (e_2.name) enthält, wird ein Link von der Entität e_1 zur Entität e_2 erstellt($e_1 \rightarrow e_2$). In diesem Schritt verwendet unser System kein NER-Tool um Entitätserwähnungen in den Begriffsbeschreibungen zu finden. Dies hat den Grund, dass die Begriffsbeschreibungen kurz sind und es somit möglich ist, über die Entitäten zu iterieren und dabei die gesamte Begriffsbeschreibung nach dem Entitätsnamen zu durchsuchen. Außerdem muss dieser Schritt nur einmal durchgeführt werden, um die Wissensbasis zu erstellen und ist im weiteren Gebrauch mit einer bereits existierenden Wissensbasis nicht notwendig.

Synonyme verbinden

Sobald die Entitäten mit einem inhaltlichen Zusammenhang verlinkt sind, werden Synonyme zwischen Entitäten gesucht. Ähnlich wie mit verwandten Entitäten gibt es hierfür eine Erkennung im Dokument (*SYN:* List<Entities>). Das Problem an den Synonymlisten ist die Eigenschaft, dass nicht alle angegebenen Synonyme definierte Entitäten sind. Oft handelt es sich bei den Synonymen um andere Schreibweisen, Wortwahlen, Formulierungen oder grammatikalische Unterschiede. Aus diesem Grund haben wir zwei unterschiedliche Ansätze zum Speichern von Synonymen implementiert. Die erste Form der Speicherung ist die Liste der Synonyme als *String*. Der Eintrag enthält also die textuelle Form der Synonyme und insbesondere alle Synonyme, die nicht in der Wissensbasis enthalten sind und keine eigene Entität darstellen. Zusätzlich sucht das System in der Synonymliste nach Entitäten, die in der

Wissensbasis einen Eintrag haben. Gefundene Entitäten werden in der Wissensbasis über die SYN-Beziehung verbunden.

Sobald dieser Schritt beendet wurde, ist die Informationsextraktion und Datenaufbereitung abgeschlossen. Anschließend folgt die Übertragung der Daten in die Wissensbasis.

5.5 Schnittstelle

Für Anfragen an die Datenbank stellt unser System explizite Schnittstellen zur Verfügung. Die Systemmodule kommunizieren ausschließlich über die bereitgestellten Schnittstellen mit der Wissensbasis. Externe Zugriffe auf die Wissensbasis sind nicht notwendig und nicht vorgesehen.

Wir haben bereits Standardanfragen implementiert, die nur um die Attribute ergänzt werden müssen. Die Datenbank wird automatisiert über die Schnittstellen popularisiert und verwendet dabei eine externe Textdatei als Datenquelle. Die Schnittstelle zur Datenbank kann einfach um weitere Anfragen ergänzt werden und ist somit für zukünftige Verwendung einfach erweiterbar.

6 Named-Entity-Recognition

Vor dem Entity Linking erfolgt die NER [21] zur Erkennung und Kategorisierung benannter Entitäten. Der Vorverarbeitungsschritt identifiziert im Text die Grenzen von benannten Entitäten und speichert diese mit der zugehörigen Position im Text. Unser System lokalisiert und kategorisiert wichtige Substantive, Eigennamen und Fachbegriffe. In diesem Kapitel sind das Vorgehen und die Entwurfsentscheidungen der NER unseres Systems spezifiziert. Das Kapitel beschreibt die Kombination aus *POS-Analyse* und *OpenIE-Analyse*, um Entitätserwähnungen im Text zu identifizieren.

In Sektion 6.1 ist definiert, welche Wörter in einem Text als Entitätserwähnung kategorisiert werden. In Sektion 6.2 sind verschiedene Tools von Stanford-*CoreNLP* für die *Named-Entity-Recognition* aufgezeigt. Neben dem *NER-Tool* wird die Verwendung der *POS-Analyse* und die *OpenIE-Analyse* diskutiert. Abschließend wird in Sektion 6.3 das Konzept der Grundform einer Entitätserwähnungen definiert. Zusätzlich wird spezifiziert, wie das System die Grundform einer Entitätserwähnung erhält und speichert.

6.1 Definition einer Entitätserwähnung

Um Entitätserwähnungen zu identifizieren, muss zunächst definiert werden, was eine Entitätserwähnung ist. Im Kontext dieser Arbeit werden Entitätserwähnung definiert als Wörter im Text, die Entitäten aus der Domäne der Softwareentwicklung sind. Die Definition einer Entität wird aus Wikipedia übernommen: *„An entity is something that exists as itself, as a subject or as an object, actually or potentially, concretely or abstractly, physically or not. It need not be of material existence. In particular, abstractions and legal fictions are usually regarded as entities.“* (Entität - Wikipedia).

6.2 Verwendetes Tool

Das *CoreNLP-Tool* von Stanford enthält eine eigene *Named-Entity-Recognition* zum Suchen und Kategorisieren von benannten Entitäten. In der englischen Sprache werden die Entitäten in die folgenden Kategorien unterteilt:

benannt PERSON, LOCATION, ORGANIZATION, MISC

numerisch MONEY, NUMBER, ORDINAL, PERCENT

temporal DATE, TIME, DURATION, SET

In dieser Arbeit ist die Kategorisierung einzelner Entitäten jedoch nicht von Relevanz. Innerhalb der Wissensbasis ist nicht vermerkt, zu welchen Kategorien die Entitäten zugehörig sind. Dementsprechend kann nicht verglichen werden, ob die Kategorie einer Entitätserwähnung mit der Kategorie einer Entität in der Wissensbasis übereinstimmt. Aus diesem Grund hat eine Kategorisierung in der *Named-Entity-Recognition* keinen Vorteil für das anschließende Entity-Linking-System.

Ein weiteres Argument gegen die Verwendung des *Named-Entity-Recognition-Tools* von Stanford ist die Definition einer Entität. Im Stanford-Tool ist alles eine Entität, das in eine der Kategorien eingeordnet werden kann. In der offiziellen Dokumentation heißt es, dass z.B. Personen, Unternehmensnamen und weiteres als Entität zählt. Im Kontext dieser Arbeit werden Entitäten als Wörter im Text definiert, die Entitäten aus der Domäne der Softwareentwicklung sind, sodass gerade Personen oder Unternehmensnamen nicht als Entität erkannt werden sollen. Folglich sind die Auslegungen bezüglich des Entitätsbegriffs grundsätzlich verschieden und eine Verwendung des *Named-Entity-Recognition-Tools* von Stanford ist wenig wegweisend. Trotzdem wird für die *Named-Entity-Recognition* das *CoreNLP-Tool* von Stanford [20] verwendet. Anstelle des *Named-Entity-Recognition-Tools* wird für die Erkennung und Kategorisierung der Entitäten im Text eine Kombination aus *Part-Of-Speech (POS)* und *OpenIE* implementiert. Die einzelnen Funktionen des *CoreNLP-Tools* werden als Blackbox betrachtet und mit dem Post-Processing-Ansatz optimiert.

6.2.1 POS-Analyse

Die *POS-Analyse* bietet eine gute Grundlage für das Finden von Entitätserwähnungen, da sie eine Kategorisierung der einzelnen Wörter enthält. Die Kategorisierung der Wörter im Text untergliedert sich in die Wortgruppen (*Tags*) der Penn Treebank aus der Abbildung 6.1.

Nb	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or SWABI
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass

Nb	Tag	Description
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol

Nb	Tag	Description
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Abbildung 6.1: Alphabetische Liste der *part-of-speech tags*

Substantive, Eigennamen und Fachbegriffe sind Nomen und werden somit in eine der folgenden *Tags* kategorisiert:

12 NN - Noun, singular or mass

13 NNS - Noun, plural

14 NNP - Proper noun, singular

15 NNPS - Proper noun, plural

6.2.2 OpenIE-Analyse

Die *OpenIE-Analyse* erkennt zusammenhängende Entitäten aus mehreren Wörtern. Dazu identifiziert das Modul relationale Tripel aus Subjekt, Beziehung und Objekt. Substantive, Eigennamen und Fachbegriffe sind entweder als Subjekt oder Objekt im Satz enthalten. Die Beziehung beschreibt das Verhältnis von dem Subjekt zu dem Objekt und entspricht meistens einem Verb, das für die *Named-Entity-Recognition* nicht weiter relevant ist.

Die Subjekte und Objekte in der *OpenIE-Analyse* können zusammenhängende Wortgruppen umfassen. Darum ist die *OpenIE-Analyse* eine gute Ergänzung zur *POS-Analyse*, welche nur einzelne Wörter kategorisiert. Im Rahmen unserer NER werden Entitäten aus einem Wort durch die *POS-Analyse* und Entitäten aus mehreren Worten durch die *OpenIE-Analyse* identifiziert.

6.2.3 Anpassungen in der Named-Entity-Recognition

In der *NER* sucht das System speziell nach Subjekten und Objekten. Die *OpenIE-Analyse* kann jedoch auch die Beziehung zwischen dem Subjekt und Objekt identifizieren und extrahieren. Um Beziehungen zwischen Subjekt und Objekt zu inkludieren muss die *OpenIE-Analyse* nur lokal angepasst werden, wobei die restlichen Module des Systems nicht modifiziert werden müssen. Ebenfalls können alle Wortgruppen aus Abbildung 6.1 in der *POS-Analyse* hinzugefügt oder entfernt werden. Durch diese lokalen Anpassungen können wir die Entitätsdefinition und die damit verbundene NER an das jeweilige Informationsbedürfnis angepassen.

6.3 Entitätserwähnungen in Grundform

Entitätserwähnungen sind im Text nicht immer in ihrer Grundform enthalten. Als Grundform gelten Wörter im Singular, Präsens ohne grammatikalische Anpassungen und Verben im Infinitiv. Dies ist ein Problem für spätere Module im Entity-Linking-Prozess, da die Mehrheit der Entitäten in der Wissensbasis in der Grundform enthalten sind. Damit es in der *Candidate-Entity-Generation* zu einem aussagekräftigen Vergleich der Entitätserwähnung und anderen Entitätsnamen kommt, muss ebenfalls die Grundform der Entitätserwähnung berücksichtigt werden. Die Grundform eines oder mehrerer Worte erstellen wir mit Hilfe des Stanford-*CoreNLP-Tools*. Somit wird zusätzlich zur originale Entitätserwähnung, die modifizierte Entitätserwähnung in Grundform gespeichert. Modifizierte Entitätserwähnungen werden bei der Erstellung gekennzeichnet, um sie im späteren Verlauf erkennen zu können. Somit ist es möglich im späteren Verlauf zu differenzieren, welche Entitätserwähnungen direkt aus dem Text extrahiert wurden und welche bereits bearbeitet sind.

7 Abkürzungsverarbeitung

Die Abkürzungsverarbeitung ist Teil der Vorverarbeitung und wird vor dem Entity-Linking-Prozess ausgeführt. In der Abkürzungsverarbeitung erfolgt eine automatische Erkennung und Auflösung selbstdefinierter Abkürzungen im Text. Die Abkürzungsverarbeitung verarbeitet keine offiziellen Abkürzungen und wird nicht zwangsweise für ein Entity Linking benötigt. Enthält die Architekturdokumentation jedoch viele selbstdefinierte Abkürzungen, kann die Abkürzungsverarbeitung das anschließende Entity Linking stark verbessern. Dafür betrachten wir die Entität *service component* aus dem laufenden Beispiel. Wenn die Entität *service component* durch *SE* abgekürzt wird, können alle Vorkommen der Abkürzung im weiteren Textverlauf identifiziert werden. Für die Abkürzungsvorkommen *SE* ist anschließend bekannt, dass sie die *service component* referenzieren. Somit kann ein Link zur gleichnamigen Entität in der Wissensbasis erstellt werden. Aus diesem Grund haben wir uns in diesem Projekt für eine Abkürzungsverarbeitung entschieden.

Dieses Kapitel beschreibt die einzelnen Schritte eines Abkürzungsverarbeitungssystems. Neben der Initialisierung wird die Abkürzungserkennung, der Abkürzungskontext, die Ausrichtung an Präfixen, die automatische Kandidatenauswahl und das Finden weiterer Abkürzungsvokommen im Text diskutiert.

Für die Erstellung des hier beschriebenen Systems wurde sich an anderen Systemen orientiert, welche einen reinen Fokus auf der Abkürzungsverarbeitung haben [31, 18, 32]. Im Folgenden sind die einzelnen Schritte unseres Abkürzungsverarbeitung-Systems im Detail aufgelistet und näher spezifiziert.

Initialisierung

Die Eingabe für den Algorithmus besteht aus mehreren Wortlisten. Wortlisten können den Algorithmus verbessern, sind jedoch nicht notwendig. Mit der Hilfe von Testläufen haben wir entschieden, welche optionale Wortlisten für die Optimierung des Systems verwendet werden sollen. In der folgenden Auflistung sind verschiedene Wortlisten gegeben.

1. Das Textdokument, das nach selbstdefinierten Abkürzungen durchsucht werden soll.
2. Optional: Eine Liste mit bekannten Wörtern, die im Dokument enthalten sind und eine ähnliche Schreibweise wie Abkürzungen haben, aber keine Abkürzungen sind (z. B. „TABLE“, „FIGURE“, Römische Zahlen).

   Bekannten Wörter haben in der Abkürzungsverarbeitung keine Probleme verursacht, da die einzelnen Buchstaben keine Ausrichtung an einer Definition haben. Aus diesem Grund haben bekannte Wörter keine Sonderstellung in der Abkürzungsverarbeitung und werden normal verarbeitet ohne Fehler zu erzeugen. Somit haben wir uns dazu entschieden, keine Liste bekannter Wörter

zu verwenden. Eine Erweiterung mit Berücksichtigung bekannter Wörter ist jedoch mithilfe einer zusätzlichen Abfrage einfach implementierbar.

3. Optional: Eine Datenbank mit Abkürzungen, ihrer Langform und ihren Definitionen.

   Diese Informationen können verwendet werden, um offizielle Abkürzungen zu identifizieren. In dieser Ausarbeitung wurde bereits eine Wissensbasis für das Entity-Linking-System erstellt, welche die benötigten Daten enthält.

   Die Wissensbasis und die darin enthaltenen Informationen über bekannte Abkürzungen werden in der Abkürzungsverarbeitung verwendet. Für jede identifizierte Abkürzung wird überprüft, ob es eine passende offizielle Abkürzung gibt. Wenn eine offizielle Abkürzung identifiziert wird, kann die Abkürzung in der Abkürzungsverarbeitung ignoriert werden.

Abkürzungserkennung

Selbstdefinierte Abkürzung entsprechen nach der Definition des hier beschriebenen Systems großgeschriebenen Wörtern im Text, mit einer begrenzten Anzahl von Buchstaben. Diese Definition wurde gewählt, da sie einen Großteil der Abkürzungen abdeckt und trotzdem einfach und regelbasiert überprüfbar ist.

Für selbstdefinierte Abkürzungen wird ein heuristischer Ansatz verwendet, um die verschiedenen Muster von Abkürzungen zu erkennen. Für die Suche der Abkürzungen werden Textmarkierungen (z. B. „(..)“) und textuelle Eigenschaften der Abkürzungen selbst verwendet. Wenn eine Abkürzung gefunden wird, ist sie entweder selbst definiert oder es handelt sich um eine offizielle Abkürzung. Ob ein gefundenes Wort eine offizielle Abkürzung ist, kann mit der Wissensbasis abgeglichen werden. Offizielle Abkürzungen werden ignoriert. Bei der Definition mit einer Textmarkierung folgt die Abkürzungsdefinition einer einfachen Struktur. Die Textmarkierungen folgen unter anderem dem folgenden Muster: *Langform (Abkürzung)* oder *Abkürzung (Langform)*. Mit diesem Ansatz sucht unser System mögliche Abkürzungs-Langformen innerhalb von Klammern oder außerhalb von Klammern.

Abkürzungskandidatenfenster - *Window*

Für eine Abkürzung bezeichnet N die Anzahl der Buchstaben. Für die Definition des Fensters kann entweder eine Fenstergröße in Abhängigkeit von N gewählt werden (z.B. $2N - 3N$ Wörtern neben dem Abkürzungsvorkommen) oder die Satzgrenzen selbst. In dieser Ausarbeitung werden Satzgrenzen verwendet. Es findet zudem eine Unterteilung des Textfensters in das Vor- und Nachfenster statt. Das Vorfenster enthält die Wörter vor der Abkürzung und das Nachfenster die Wörter hinter der Abkürzung.

Abkürzungen an ihren Präfixen ausrichten

Es gibt verschiedene Ansätze um Abkürzungen an den Präfixen ihrer Definition bzw. Langform auszurichten. In dieser Ausarbeitung werden die Buchstaben der Abkürzung mit den Anfangsbuchstaben der umgebenden Wörter abgeglichen. Zum Beispiel ist die Definition *The **b**ody **o**f **k**nowldge (BOK)* eine korrekte Abkürzung,

die mit den Anfangsbuchstaben der Langform übereinstimmt. Es wird zusätzlich erlaubt, dass die Wörter der Langform nicht direkt hintereinander stehen müssen. Dies ermöglicht das Überspringen von Bindewörtern (z.B. *The **b**ody of **k**nowldge (BK)*). Zusätzlich wird die Anfangsbuchstabenanforderung gelockert, wobei das System Übereinstimmungen mit allen Präfixen zulässt. Beispielsweise kann durch diese Lockerung auch die Definition *The **bod**y **o**f **k**nowldge (BODOK)* ausgerichtet werden.

Alternativ zu den regelbasierten Ansätzen kann ein maschineller Lernansatz gewählt werden. In dem hier beschriebenen System ist die Abkürzungsverarbeitung nur ein Vorverarbeitungsschritt für das Entity-Linking-System. Außerdem behandelt die Abkürzungsverarbeitung nur selbstdefinierte Abkürzungen und betrifft keine offiziellen Abkürzungen. Aus diesem Grund wird die Methodik möglichst einfach gehalten und es wird auf einen maschinellen Lernansatz verzichtet.

Automatische Kandidatenauswahl

Um die verschiedenen Abkürzungskandidaten einer Abkürzung zu bewerten, werden verschiedene Merkmale verwendet. Für die Auswahl des besten Kandidaten wird jedem Kandidat eine Konfidenz (*confidence*) zugeordnet. Die *confidence* berechnet sich aus drei Werten:

1. Die Anzahl der Buchstaben, die an den Anfangsbustaben der Langform ausgerichtet sind.
2. Der Abstand der Abkürzung zum entferntesten Wort der Langform.
3. Die Anzahl der übersprungenen Wörter innerhalb der Langform.

Die *confidence* des besten Kandidaten wird mit einem Schwellenwert verglichen. Falls die *confidence* höher als der Schwellenwert ist, wird die Langform des Kandidaten zur Abkürzung zugeordnet und akzeptiert. Abkürzungen mit einer akzeptierten Langform werden als Entitätserwähnungen hinzugefügt. Dabei entspricht die Langform dem Text der Entitätserwähnung.

Ein alternativer Ansatz basiert auf Komprimierung [36]. Es bietet einen Vorteil gegenüber anderen Methoden, da es weniger auf Heuristiken angewiesen ist. Dieser Ansatz wurde nicht gewählt, da die Wirksamkeit noch nicht vollständig transparent ist und die heuristischen Bewertungskriterien mehr Anwendung in der Praxis aufweisen. Außerdem erzielen heuristische Bewertungskriterien stabile Ergebnisse und sind einfach in der Implementierung.

Finden von weiteren Vorkommen der Abkürzung im Text

Das Dokument wird nach weiteren Vorkommen der definierten Abkürzung durchsucht. Dieser Schritt kann entweder aktiv oder passiv implementiert sein. Ein aktiver Ansatz beschreibt die Suche der definierten Abkürzung im weiteren Textverlauf und eine entsprechende Markierung gefundener Textstellen. Ein passiver Ansatz merkt sich selbstdefinierte Abkürzungen und wird aktiviert, wenn im späteren Verlauf eine Abkürzung erneut identifiziert wird. In diesem System verwenden wir den passiven Ansatz.

8 Candidate-Entity-Generation

In der Vorverarbeitung hat die *Named-Entity-Recognition* bereits die Entitätserwähnung im Text identifiziert. Die *Candidate-Entity-Generation* ist das erste Modul im Entity-Linking-Prozess. Das *Candidate-Entity-Generation*-Modul sucht in der Datenbank nach Entitäten, die am wahrscheinlichsten den Entitätserwähnungen im Text entsprechen. Die gefundenen Entitäten werden als Kandidatenentitäten in das Kandidatenentitäten-Set hinzugefügt. In diesem Kapitel wird die Generierung des Kandidatenentitäten-Sets spezifiziert. Anschließend folgen die Entwurfsentscheidungen der Anfragen an die Wissensbasis und ein illustratives Beispiel für die Kandidatengenerierung in Abbildung 8.1.

Für die Realisierung der *Candidate-Entity-Generation* haben wir uns im Rahmen dieser Ausarbeitung für die *Name-Dictionary-Based-Techniques* entschieden. *Name-Dictionary-Based-Techniques* sind ein Ansatz zur Generierung von Kandidatenentitäten und werden ebenfalls in anderen Systemen (vrgl. [7, 8, 9]) eingesetzt. Die Existenz einer Datenbank mit Entitäten aus der Zieldomäne wird in diesem Ansatz vorausgesetzt. In dem hier beschriebenen System entspricht die Datenbank der bereits erstellten Wissensbasis (siehe Kapitel 5). *Name-Dictionary-Based-Techniques* nutzen verschiedene Kombinationen von Merkmalen der Wissensbasis, um geeignete Kandidatenentitäten zu finden beziehungsweise aus der Wissensbasis zu filtern.

Das Modul erzeugt für jede Entitätserwähnung m im Text ein Kandidatenentitäten-Set E_m. Das Kandidatenentitäten-Set enthält alle Entitäten der Wissensbasis, auf die sich die Entitätserwähnung möglicherweise bezieht. Nur Kandidatenentitäten, die im Kandidatenentitäten-Set enthalten sind, können in kommenden Modulen weiterverarbeitet werden. Außerdem werden Entitätserwähnungen mit leerem Kandidatenentitäten-Set entfernt. Denn wenn das Kandidatenentitäten-Set leer ist, hat das System keine passenden Entitäten in der Wissensbasis für die Entitätserwähnung gefunden. Um sicherzustellen, dass die richtige Entität im Kandidatenentitäten-Set enthalten ist, sollten für die Auswahl der Kandidatenentitäten lockere Aufnahmebedingungen gewählt werden. Zu strikte Aufnahmebedingungen können zur Folge haben, dass passende Entitäten der Wissensbasis nicht ins Kandidatenentitäten-Set einer Entitätserwähnung aufgenommen werden. Da das Kandidatenentitäten-Set in den nachfolgenden Modulen nur gefiltert und nicht erweitert wird, kann eine unvollständige Generierung von Kandidatenentitäten nicht in den anschließenden Modulen ausgeglichen werden. Trotzdem sollten die Aufnahmebedingungen nicht zu locker sein, da sonst viele unpassende Kandidatenentitäten in die Kandidatenentitäten-Sets zugelassen werden. Außerdem verhindern zu lockere Aufnahmebedingungen das Filtern von Entitätserwähnungen ohne passende Entitäten. Entitätserwähnungen ohne passende Entitäten können fälschlicherweise erhalten bleiben, wenn ihre Kandidatenentitäten-Sets fälschlicherweise gefüllt wurden.

Anfragen an die Wissensbasis werden über den Programmcode generiert. Unsere Datenbank unterstützt die Abfragesprache Cypher. Cypher ist eine deklarative Graph-

Abfragesprache und ermöglicht Datenabfragen in einem Eigenschaftsgraphen. Im Rahmen der *Candidate-Entity-Generation* stützen sich die Anfragen an die Wissensbasis auf den Namen bzw. die Oberflächenform der Entitätserwähnung. Viele Ansätze zur Generierung der Kandidatenentitäten mit *Name-Dictionary-Based-Techniques* basieren auf Zeichenfolgenvergleichen [10]. Auch in unserem System werden Zeichenfolgenvergleiche eingesetzt, um die Entitätserwähnung mit den Entitätsnamen zu vergleichen. Dabei wird überprüft, ob die Entitätserwähnung eine Übereinstimmung mit dem Namen der Entität in der Wissensbasis hat. Für die Anfragen bezüglich der Zeichenfolgen können mehrere Operatoren verwendet werden. Im Folgenden ist eine Auswahl an Anfragen gegeben, die für die *Candidate-Entity-Generation* in Frage kommen. Dabei wird die Wirksamkeit der einzelnen Anfragen untersucht und argumentiert.

Vergleich auf Gleichheit

Mit dem Vergleich auf Gleichheit wird auf die exakte Übereinstimmung zwischen zwei Zeichenfolgen überprüft. Eine auf Gleichheit basierende Abfrage zwischen dem Namen der Entitätserwähnung und dem Namen der Entität in der Wissensbasis ist sehr restriktiv. Da die Aufnahmebedingungen für eine Entität nicht zu strikt sein sollten, ist eine Abfrage mit Vergleich auf Gleichheit (=) zu restriktiv und im Kontext dieser Ausarbeitung nicht ausreichend.

Vergleich mit *contains*

Eine weniger restriktive Abfrage kann durch die Verwendung des *contains*-Operators umgesetzt werden. Der *contains*-Operator überprüft, ob eine Zeichenkette in einer anderen enthalten ist. Die Abfrage mit dem *contains*-Operator überprüft, ob die Entitätserwähnung im Namen einer Entität enthalten ist und ergänzt die Entität bei Erfüllung in das Kandidatenentitäten-Set. Dieses Vorgehen hat den Nachteil, dass kurze Entitätserwähnung in sehr vielen Entitätsnamen enthalten sind. Dies führt dazu, dass die Abfrage nur sehr geringe Restriktionen aufweist. Ein weiterer Nachteil ist, dass das Resultat nur Entitätsnamen enthält, die mindestens so lang sind, wie die Entitätserwähnung selbst.

Vergleich der längsten gemeinsamen Teilfolge

Eine Alternative zu den zwei aufgezeigten Herangehensweisen ist die längste gemeinsame Teilfolge (*longest-common-substring*). Bei der Abfrage der längste gemeinsame Teilfolge wird überprüft, ob die längste gemeinsame Teilfolge der Entitätserwähnung und des Entitätsnamen eine bestimmte relative Mindestlänge erfüllt. Genau dann, wenn die gefundene Teilfolge die Mindestlänge erfüllt, sind die Strings ähnlich. Eine *longest-common-substring*-Abfrage hat den Vorteil, dass sie bei kurzen Entitätserwähnungen strikter sein kann als eine Abfrage mit dem *contains*-Operator und weniger strikt als der Vergleich auf Gleichheit. Wie strikt die Abfrage tatsächlich ist, wird über den konfigurierbaren Wert der relativen Mindestlänge festgelegt. Eine *longest-common-substring*-Abfrage ist jedoch nicht nativ in Cypher enthalten und muss selbst implementiert werden. Um den *longest-common-substring* zu bestimmen, muss zunächst der Entitätsname jeder Entität in der Wissensbasis abgefragt werden. Anschließend wird dieser auf die längste gemeinsame Teilfolge mit der En-

titätserwähnung überprüft. Dieses Vorgehen skaliert mit einer ausreichend großen Wissensbasis nicht, da über alle Entitäten der Wissensbasis iteriert werden muss.

Aus den beschriebenen Möglichkeiten ist die Verwendung der längsten gemeinsamen Teilfolge am besten für eine Anfrage im Rahmen dieser Arbeit geeignet. Aus diesem Grund haben wir uns dazu entschieden, die Grundidee des *longest-common-substrings* zu übernehmen und eine besser skalierbare Variante zu implementieren.

Für die Umsetzung der abgewandelten *longest-common-substring*-Abfrage haben wir festgelegt, wie viele zusammenhängende Zeichen der Entitätserwähnung ebenfalls im Entitätsnamen enthalten sein müssen. Je höher der Wert gewählt wird, desto restriktiver ist die Suche. Für einen Wert $\alpha \in [0, 1]$ haben wir alle Werte mit einer Schrittgröße von 0.1 überprüft. Nach mehreren Testläufen und einer händischen Evaluation der Kandidatenentitäten-Sets hat sich ein Wert von $\alpha = 0.6$ (60%) als sinnvoll herausgestellt.

Die Entitätserwähnungen werden in alle zusammenhängenden Zeichenketten unterteilt, die mindestens α Prozent der Entitätserwähnungslänge enthalten. In Abbildung 8.1 sind diese Zeichenketten für die Entitätserwähnung *server* aus dem laufenden Beispiel tabellarisch illustriert. Nach der Generierung wird überprüft in welchen Entitätsnamen mindestens eine Zeichenkette enthalten ist. Für diese Abfrage kann der *contains*-Operator verwendet werden. Für alle gefundenen Entitätsnamen wird die zugehörige Entität dem Kandidatenentitäten-Set hinzugefügt.

S	E	R	V	E	R
S	E	R	V		
	E	R	V	E	
		R	V	E	R

Abbildung 8.1: Zusammenhängende Zeichenketten mit einer Wortlänge von mindestens α = (60%) für die Entitätserwähnung *server*

In Abbildung 8.1 werden für die Entitätserwähnung *server* folgende Entitäten der Wissensbasis identifiziert:

1. **server**
2. **server**-side

3. **serv**ice
4. re**serve**

Der Wert von α bezieht sich in diesem Vorgehen ausschließlich auf die Länge der Entitätserwähnung und nicht auf die Länge des Entitätsnamen. Diese Einseitigkeit stellt kein Problem dar, da in der Kandidatengenerierung keine konkreten Werte für die Übereinstimmung der Entitätserwähnung und den Entitäten im Kandidatenentitäten-Set berechnet werden müssen.

Wenn das *Candidate-Entity-Generation*-Modul beendet ist, steht für jede Entitätserwähnung das Kandidatenentitäten-Set fest. Die Entitätserwähnung mit den zugehörigen Kandidatenentitäten-Sets werden an das *Candidate-Entity-Ranking*-Modul übergeben.

9 Candidate-Entity-Ranking

Nach der *Candidate-Entity-Generation* werden die Kandidatenentitäten der Kandidatenentitäten-Sets analysiert. Für die Analyse und Bewertung der Kandidatenentitäten ist das *Candidate-Entity-Ranking* zuständig. Dazu berechnet das Modul die Rankingfunktion $rk : \mathrm{E}_m \rightarrow \mathbb{R}$, $e \rightarrow [0, \infty)$ bezüglich ausgewählter Merkmale, wobei $e \in \mathrm{E}_m$ eine Kandidatenentität aus dem Kandidatenentitäten-Set ist. Ein hoher Rankingwert entspricht einer hohen Wahrscheinlichkeit, dass die Entitätserwähnung der Kandidatenentität entspricht.

Das Kapitel spezifiziert die Rankingfunktion unseres Systems und die darin verwendete Kombination aus unabhängigen und kollektiven Rankingmethoden. Die Kombination der konkreten Merkmale der unabhängigen Rankingmethoden ist in kontextunabhängige und kontextabhängige Merkmale unterteilt. Die Sektion 9.1 spezifiziert die Verwendung aller in unserem System verwendeten kollektiven Rankingmethoden. Anschließend wird in der Sektion 9.2 die Verwendung der unabhängigen Rankingmethoden aufgelistet. Abschließend wird in Sektion 9.3 die Zusammensetzung der Rankingfunktion definiert.

9.1 Kollektive Rankingmethoden

Aus den kollektiven Rankingmethoden haben wir zwei verschiedene Ansätze betrachtet. Die Popularität der Entität wurde ausgewählt und im Rahmen dieser Arbeit implementiert.

Popularität der Entität

In den kollektive Rankingmethoden nehmen wir an, dass eine bereits verlinkte Entität eine höhere Wahrscheinlichkeit hat, erneut verlinkt zu werden. Die Annahme basiert auf der Erkenntnis, dass eine bereits erwähnte Entität der Wissensbasis potenziell noch öfter im weiteren Verlauf einer Architekturdokumentation erwähnt wird. Eine hohe Popularität einer Kandidatenentität induziert eine höhere Wahrscheinlichkeit für ein erneutes Linking. Das Kriterium der Popularität muss nach jedem Satz mit neuen Verlinkungen aktualisiert werden. Wenn eine Entität verlinkt wird, erhöht diese Verlinkung den Rankingwert aller Kandidatenentitäten der gleichen Entität in späteren Sätzen. Somit wird der Rankingwert nach Abschluss aller Verlinkungen innerhalb eines Satzes für die Kandidatenentitäten der folgenden Sätze aktualisiert.

Clustering Verfahren

Eine weitere kollektiven Rankingmethode basiert auf einem *Clustering*-Verfahren. Entitätserwähnungen verweisen innerhalb eines Dokuments oft auf verwandte Themen. Ein *Clustering* der Entitätserwähnungen kann hilfreich für die Auswahl von Verlinkungen und die Berechnung von Rankingwerten sein. Die besten Ergebnisse erzielen *Clustering*-Verfahren aus den kollektiven Rankingmethoden jedoch in Texten ohne festgelegte Domäne. Für Texte aus einer festgelegten Domäne ist es

schwierig, klar separierbare Cluster zu bilden. Bei dem hier beschriebenen System handelt es sich um Texte, die aus einer festgelegten Domäne stammen. Deshalb hat das *Clustering*-Verfahren im Kontext dieser Arbeit eine geringere Wirksamkeit und wird nicht verwendet.

9.2 Unabhängigen Rankingmethoden

Aus den kollektiven Rankingmethoden wird die Popularität der Entität in das Ranking übernommen. Im Rahmen dieser Arbeit werden zusätzlich zur Popularität auch Merkmale von unabhängigen Rankingmethoden verwendet. Unabhängige Rankingmethoden sind unabhängig von bisherigen Verlinkungen und bilden die Basis für die Berechnung von Rankingwerten. Unabhängige Rankingmethoden werden in kontextunabhängige und kontextabhängige Merkmale unterteilt.

In diesem Ansatz verwenden wir ausschließlich nicht überwachte Rankingmethoden, um das System unabhängig von Trainingsdaten zu halten. Nicht überwachte Rankingmethoden benötigen keine annotierte Trainingsdaten, da kein Modell trainiert werden muss. In den folgenden Untersektionen wird die Verwendung der kontextunabhängigen und kontextabhängingen Merkmale spezifiziert.

9.2.1 Kontextunabhängige Merkmale

Kontextunabhängige Merkmale verlassen sich auf die Oberflächenform der Entitätserwähnung und das Wissen über die Kandidatenentität. In der Auflistung wird zwischen Binärenwerten und Verhältniswerten unterschieden. Die zugehörigen Aussagen der Binärwerte können entweder wahr (= 1) oder falsch (= 0) sein, wobei es keine Zwischenwerte gibt. Verhältniswerte berechnen sich aus einem Verhältnis und liegen im Zahlenbereich der reelen Zahlen. In der folgenden Auflistung sind die ausgewählten kontextunabhängigen Merkmale aufgetragen, die für die Rankingfunktion *rk* unseres Systems verwendet werden und somit das *Candidate-Entity-Ranking* bestimmen.

String-Ähnlichkeitsmaße zwischen Entitätserwähnung und Kandidatenentität

1. Binär: Die Entitätserwähnung entspricht genau dem Namen der Kandidatenentität (*perfect match*).

2. Binär: Der Name der Kandidatenentität ist das Präfix oder Postfix der Entitätserwähnung (**p**refix **o**r **p**ostfix of the **e**ntity **m**ention = POPem).

3. Binär: Die Entitätserwähnung ist das Präfix oder Postfix des Namens der Kandidatenentität (**p**refix **o**r **p**ostfix of the **c**andidate **e**ntity = POPce).

4. Binär: Die Entitätserwähnung ist vollständig im Namen der Kandidatenentität enthalten oder umgekehrt (**con**tains = CON).

5. Binär: Die Anzahl der Wörter in der Entitätserwähnung ist gleich der Anzahl der Wörter in der Kandidatenentität (**e**qual **w**ord **c**ount = EWC).

6. Verhältniswert: Die Anzahl gleicher Wörter in der Entitätserwähnung und der Kandidatenentität im Verhältnis zu der Anzahl der Wörter der Entitätserwähnung und der Kandidatenentität (**i**dentical **w**ord **r**atio = IWR). Das IWR-Merkmal entspricht einem Verhältniswert, wenn die Entitätserwähnung oder die Kandidatenentität aus mehreren Wörtern besteht. Wenn die Entitätserwähnung und Kandidatenentität jeweils nur aus nur einem Wort besteht, reduziert sich das IWR-Merkmal auf einen binären Wert.

7. Verhältniswert: Die längste gemeinsame Teilfolge (*longest common substring*) im Verhältnis zur Länge der Entitätserwähnung und des Entitätsnamen (**l**ongest **c**ommon **s**ubstring **r**atio = LCSR).

8. Verhältniswert: Die Länge der Entitätsbeschreibung (**d**escription **l**ength **r**atio = DLR). Der DLR-Wert ist klein für verhältnismäßig lange und groß für verhältnismäßig kurze Entitätsbeschreibungen. Eine lange Entitätsbeschreibung kann mehr Informationen enthalten. Aus diesem Grund eignet sich der DLR-Wert zum gewichten kontextabhängiger Merkmale, die sich auf die Entitätsbeschreibung beziehen. Details bezüglich des DLR-Werts und die Zusammensetzung der anderen Merkmale in der Rankingfunktion befinden sich in Sektion 9.3.

9.2.2 Kontextabhängige Merkmale

Kontextabhängige Merkmale basieren auf dem Kontext der Entitätserwähnung. Sie messen die Ähnlichkeit zwischen dem Kontext der Entitätserwähnung und den Zusatzinformationen der Kandidatenentität in der Wissensbasis. Für eine praktische Anwendung siehe das System von Guo et al. [9]. In dieser Arbeit wird der Kontext der Entitätserwähnung mit den Beschreibungen der Kandidatenentitäten verglichen. Dafür wird für jede Entitätserwähnung der Kontext als eine Sammlung (Beutel) von Wörtern dargestellt.

Wörterbeutel können entweder lokal um die Entitätserwähnung gesammelt werden oder aus allen Bereichen im Dokument, in der die Entitätserwähnung erscheint. In dem hier beschriebenen System werden nur lokale Wörterbeutel verwendet. Diese Entscheidung ist für das System sinnvoll, um das WSD-Problem effektiver angehen zu können. Das WSD-Problem entsteht durch die Existenz von Homonymen. Auch in Softwarearchitekturdokumentationen ist es möglich, dass eine Entitätserwähnung den gleichen Namen hat, wie eine andere Entitätserwähnung und trotzdem auf eine andere Entität referenziert. Zum Beispiel kann der Entitätsname *"function"* auf eine mathematische Funktion verweisen oder auf eine Funktionalität. Falls die Wörterbeutel aus mehreren Bereichen verschiedener Vorkommen einer Entitätserwähnung gesammelt werden, führt dies zu einer Vermischung des Wörterbeutels, wenn die verschiedenen Entitätserwähnung unterschiedliche Bedeutungen haben. Aus diesem Grund wird der Wörterbeutel nur aus dem lokalen Kontext erstellt. Somit können zwei Entitätserwähnungen mit gleichem Namen anhand des unterschiedlichen Kontexts und dem daraus resultierenden Wörterbeutel unterschieden werden.

Die Kontextgröße, auch Fenster (*Window*) genannt, bezeichnet die Bereichsgröße, aus dem der Wörterbeutel erstellt wird. Für unser System haben wir uns dazu entschieden,

die Fenstergröße satzweise zu limitieren. Das Fenster einer Entitätserwähnung ist also der gesamte Satz, in dem die Entitätserwähnung vorkommt. In dem laufenden Beispiel wird für die Entitätserwähnung *server* des zweiten Satzes, der gesamte Satz als Fenster verwendet. Das Fenster ist durch eine Unterstreichung markiert.

The database is an organized collection of structured information.
The service component communicates with the ***server****. Servers can provide [...]* (9.1)

In Softwarearchitekturdokumentationen soll der Text möglichst präzise und ausdrucksstark sein. Außerdem sollte ein Satz genau eine Tatsache, Eigenschaft oder einen Gedanken repräsentieren. Die Sätze sind selten eine Kombination mehrerer Gedanken und somit sind auch selten mehrere nicht zusammenhängende Domänenbereiche in einem Satz enthalten. Daraus folgt, dass der gesamte Satz einer Entitätserwähnung einen thematischen Zusammenhang zu der Entitätserwähnung haben kann und eine vollständige Unabhängigkeit eher unwahrscheinlich ist. Eine Fenstergöße auf Satzebene ist folglich einfach implementierbar und speziell bei wissenschaftlichen Texten und Architekturdokumentationen eine gute Festlegung.

Um nun den Wörterbeutel einer Entitätserwähnung zu befüllen, werden die Wörter des zu betrachtenden Fensters zusätzlich überprüft. Es werden nicht alle Wörter des Satzes in den Wörterbeutel übernommen. Der Wörterbeutel wird ausschließlich mit weiteren Entitätserwähnungen des Satzes befüllt. Das hat den Vorteil, dass der Wörterbeutel nur echte Entitätserwähnungen enthält und keine Bindewörter oder Rauschen. Somit ist durch die *Named-Entity-Recognition* bereits definiert, welche Wörter im Satz relevant sind. In dem laufenden Beispiel wird für die Entitätserwähnung *server* folgender Wörterbeutel erstellt:

$$\text{Wörterbeutel}_{server} = \{\text{service component, service, component}\} \tag{9.2}$$

Nach der Erstellung des Wörterbeutels wird für alle Kandidatenentitäten überprüft, wie ähnlich ihr Kontext zum Wörterbeutel ist. In unserem System wird für jede Kandidatenentität der Beschreibungstext in der Wissensbasis als Kontext verwendet. In den Beschreibungstexten wird nach den Wörtern des Wörterbeutel gesucht. Ein Beispielablauf für die Entität *server* des laufenden Beispiels ist in Abbildung 9.1 abgebildet.

Die hier aufgezeigte Suche unseres Systems ist wenig restriktiv. Die Suche lässt sowohl die originale Textform der Entitätserwähnungen im Wörterbeutel als auch die Namen der Entitäten im Kandidatenentitäten-Set der Entitätserwähnungen zu. Wenn mindestens eine Oberflächenform einer Entitätserwähnung in dem Beschreibungstext der Entität der aktuell betrachten Entitätserwähnung enthalten ist, wird der Zählwert (**r**elated **c**ount = RC) der Entität um eins erhöht. Mit dem Zählwert wird überprüft, zu wie vielen Entitätserwähnungen im Text eine Entität verbunden ist. Der Zählwert einer Entität liegt somit immer zwischen null und der Anzahl an Entitätserwähnung im Satz.

Es ist unerheblich, ob ein Satz mehr oder weniger Entitätserwähnungen hat. Eine Kandidatenentität wird immer nur mit Kandidatenentitäten derselben Entitätserwähnung verglichen. Somit kann der Zählwert als absoluter Wert angegeben und verarbeitet werden. Insgesamt ist es also nicht relevant, wie viele Entitätserwähnungen im Satz enthalten sind, da für alle Entitäten im Kandidatenentitäten-Set der gleiche Wörterbeutel der Entitätserwähnung verwendet wird.

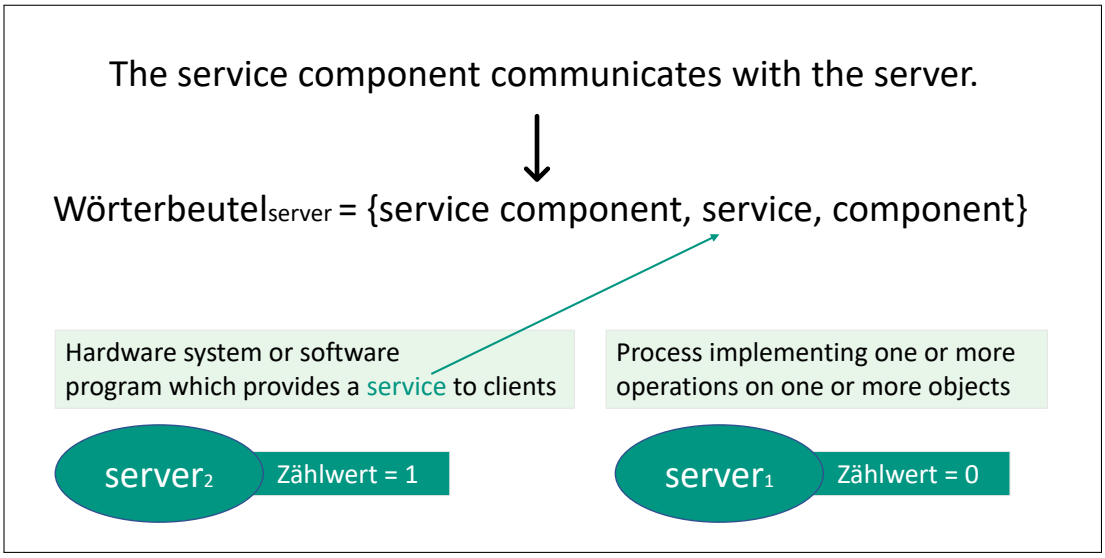

Abbildung 9.1: Berechnung des Zählwerts für die Entitätserwähnung *server*

9.3 Zusammensetzung der Rankingfunktion

Das Kapitel hat die kontextunabhängigen Merkmale in der Untersektion 9.2.1 und die kontextabhängigen Merkmale in der Untersektion 9.2.2 aufgelistet. Diese Sektion spezifiziert die konkrete Rankingfunktion rk und die darin verwendete Kombination aus kontextunabhängigen und kontextabhängigen Merkmalen für unser System. Die Rankingfunktion $rk : \mathrm{E}_m \rightarrow \mathbb{R}, e \rightarrow [0, \infty)$ ist mit Hilfe ausgewählter Merkmale definiert, wobei $e \in \mathrm{E}_m$ eine Kandidatenentität aus dem Kandidatenentitäten-Set E_m ist.

$$\begin{aligned} rk &= w_1 \cdot LCSR + w_2 \cdot IWR + w_3 \cdot (POP + CON + EWC) + w_4 \cdot (DLR \cdot RC) \\ POP &= (POPem|POPce) \end{aligned} \tag{9.3}$$

In der Definition von rk sind die Merkmale in verschiedene Merkmalsgruppen eingeteilt. Eine Merkmalsgruppen ist ein Zusammenschluss zusammengehöriger Merkmale. Merkmalsgruppen werden nur zusammen gewichtet und nicht separat für jedes einzelne Merkmal. Merkmalsgruppen sind jeweils durch eine Klammer gekennzeichnet. Die Merkmale (POP, CON, EWC) sowie (RC, DLR) bilden jeweils eine Merkmalsgruppe.

Die Merkmalsgruppe (POP, CON, EWC) ist ein Zusammenschluss aller binären Merkmale bezüglich der Stringähnlichkeit.

Die Merkmalsgruppe (DLR, RC) ist ein Zusammenschluss aus einem kontextunabhängigen und einem kontextabhängigen Merkmal. Das Merkmal RC bezeichnet den Zählwert einer Entität und das Merkmal DLR bezeichnet einen Verhältniswert bezüglich der Länge der Entitätsbeschreibung. Die beiden Merkmale bilden eine Gruppe, da sie nur in Kombination aussagekräftig sind. Entitäten mit langen Beschreibungstexten haben mehr Wörter und somit eine höhere Wahrscheinlichkeit für einen höheren Zählwert. Um dieses Ungleichgewicht auszugleichen, muss der Zählwert mit der Länge des Beschreibungstextes ins Verhältnis gesetzt werden. Insgesamt repräsentiert die Merkmalsgruppe also die Anzahl der Entitätserwähnungen der Entitätsbeschreibung im Verhältnis zur Länge

der Entitätsbeschreibung. Durch dieses Vorgehen eliminieren wir Vor- und Nachteile für Entitäten mit langen bzw. kurzen Entitätsbeschreibungslängen. Aus diesem Grund bilden die Merkmale RC und DLR eine Merkmalsgruppe und werden nur gemeinsam gewichtet.

Das *Candidate-Entity-Ranking*-Modul hat als Eingabe die Entitätserwähnung mit den zugehörigen Kandidatenentitäten-Sets bekommen. Das *Candidate-Entity-Ranking* ist beendet, sobald jede Kandidatenentität aus den Kandidatenentitäten-Sets einen Rankingwert zugeschrieben bekommen hat. Die Entitätserwähnungen werden anschließend mit den zugehörigen Kandidatenentitäten-Sets an das *Unlinkable-Mention-Prediction*-Modul übergeben.

10 Unlinkable-Mention-Prediction

Nach dem *Candidate-Entity-Ranking* folgt die *Unlinkable-Mention-Prediction* als abschließendes Modul des Entity-Linking-Systems. Wie bereits in Kapitel 2 beschrieben, validiert das Modul, ob für eine Entitätserwähnungen der Link zur identifizierten Entität der Wissensbasis erstellt werden soll. Dafür sucht die *Unlinkable-Mention-Prediction* die besten Entitätserwähnungen im Text heraus, sodass möglichst viele Entitätserwähnungen abgedeckt sind, jedoch möglichst wenige Überschneidungen zwischen ihnen existieren. Für die ausgewählten Entitätserwähnungen sucht das Modul die beste Kandidatenentität aus dem Kandidatenentitäten-Set der Entitätserwähnungen heraus und validiert, ob ein Link zu dieser erstellt werden soll.

Das Kapitel behandelt das Modul der *Unlikable-Mention-Prediction* unseres Systems. Untergliedert wird das Kapitel in die Erstellung von Prioritätslisten für Entitätserwähnungen in Sektion 10.1 und die Optimierung der Prioritätslisten in Sektion 10.2. Anschließend wird spezifiziert, zu welchen Kandidatenentitäten ein Link erstellt werden soll und wie dieser Link dargestellt wird.

Für die Vorhersage nicht verknüpfbarer Entitätserwähnungen verwendet das hier beschriebene System keine überwachten maschinellen Lerntechniken. Das System verwendet einfache heuristische Methoden in Kombination mit dem Schwellenwertverfahren.

10.1 Optimale Abdeckung

In der Domäne der Softwareentwicklung gibt es Entitäten, die aus mehreren Worten bestehen, wobei die einzelnen Worte ebenfalls eigenständige Entitäten sind. Zum Beispiel besteht die Entität *service component* des laufenden Beispiels, aus den Entitäten *service* und *component*. Alle drei Entitätserwähnungen haben jeweils einen eigenständigen Entitätseinträg in der Wissensbasis wie in Abbildung 10.1 dargestellt wird.

Die *Named-Entity-Recognition* erkennt in einem Satz alle drei Entitäten. Somit wird auch zu allen Entitäten ein Kandidatenentitäten-Set mit entsprechendem Ranking erstellt. Insgesamt ist es möglich, alle drei Entitätserwähnungen zu den zugehörigen Entitäten der Wissensbasis zu verlinken. Besser wäre es jedoch, nur die größtmögliche Entität zu verlinken und enthaltene bzw. überdeckte Entitäten auszusparen. Um diesen Ansatz umzusetzen, muss zunächst für jede Entitätserwähnung analysiert werden, welche Entitätserwähnungen sie überdeckt. Sobald dies feststeht, erstellt das Modul eine Prioritätsliste mit allen Entitätserwähnungen, welche von keiner anderen Entitätserwähnung überdeckt sind. Jeder Satz enthält genau eine Prioritätsliste. Die Prioritätsliste entspricht einer optimalen Abdeckung über alle Entitätserwähnungen eines Satzes und ist leer, falls ein Satz keine Entitätserwähnung enthält. Satzgrenzen werden nicht überschritten und eine Entitätserwähnung kann sich nicht über Satzgrenzen hinweg erstrecken.

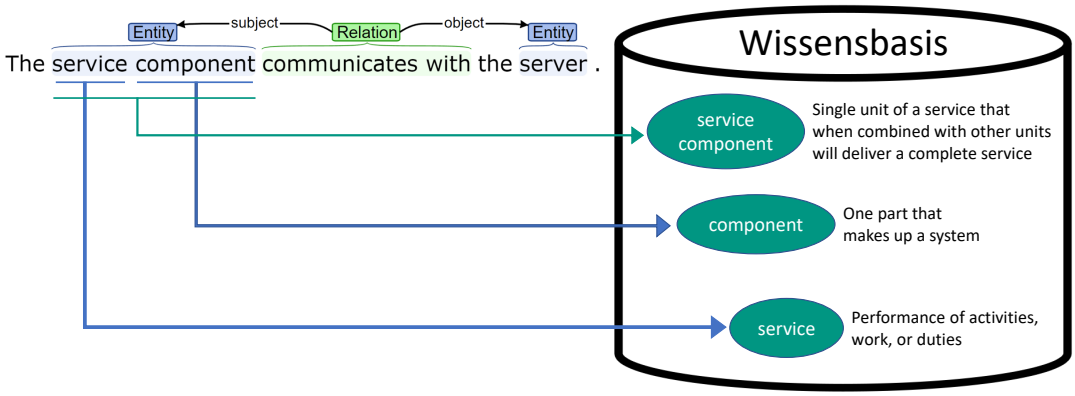

Abbildung 10.1: Überdeckte Entitätserwähnungen

Eine solche Prioritätsliste hat optimale Abdeckung im Sinne der Positionen der einzelnen Entitätserwähnungen. Die Abdeckung ist ein sehr wichtiges aber kein alleinstehendes Kriterium, das optimiert werden muss. Damit eine Prioritätsliste im Sinne dieser Arbeit optimal ist, muss zusätzlich sicher gestellt werden, dass die Entitätserwähnungen der Prioritätsliste mindestens eine gute Kandidatenentität haben. Zur Erfüllung dieses Kriteriums haben wir eine Optimierung der Prioritätsliste implementiert, die in der nächsten Sektion näher behandelt wird.

10.2 Optimierung der Prioritätslisten

Wir nennen ein Kandidatenentitäten-Set zu einer Entitätserwähnung sicher, wenn mindestens eine Kandidatenentität einen hohen Rankingwert hat oder die Entitätserwähnung mit dem Namen einer Kandidatenentität vollständig übereinstimmt. Ein hoher Rankingwert ist im Kontext dieses Moduls ein Wert, der überhalb eines Schwellenwerts $\epsilon \in [0, \infty)$ liegt. Ein unsicheres Kandidatenentitäten-Set enthält nur Kandidatenentitäten mit niedrigen Rankingwerten und hat somit eine unsichere Verbindung zur Entitätserwähnung. Zusätzlich zum Kriterium der optimalen Abdeckung wird überprüft, ob die Kandidatenentitäten-Sets nach obiger Definition sicher sind. Denn es ist möglich, dass eine Entitätserwähnung in der Prioritätsliste mit einem unsicheren Kandidatenentitäten-Set andere Entitätserwähnung überdeckt. Die überdeckten Entitätserwähnungen können jedoch in Kombination den gleichen Bereich abdecken und jeweils sichere Kandidatenentitäten-Sets haben. Um dies zu verhindern, wird für jede Entitätserwähnung mit schwachen Kandidatenentitäten-Set in der Prioritätsliste untersucht, ob die enthaltenen Entitätserwähnungen in Kombination eine annähernd gleiche Abdeckung erreichen. Die Abdeckung muss nicht identisch sein, da zum Beispiel Bindewörter keine Entitäten sind und nicht abgedeckt werden müssen. Falls eine Kombination der enthaltenen Entitätserwähnungen zusätzlich sichere Kandidatenentitäten-Sets hat, werden diese der Prioritätsliste hinzugefügt und ersetzten die ursprüngliche Entitätserwähnung.

Dieses Vorgehen wird ebenfalls auf neu hinzugefügte Entitätserwähnungen angewendet, sodass alle beliebig tief verschachtelte Entitätserwähnungen untersucht werden. Falls

eine Entitätserwähnung kein sicheres Kandidatenentitäten-Set hat, jedoch keine andere Entitätserwähnung überdeckt, kann sie nicht ersetzt werden und bleibt in der Prioritätsliste erhalten. Insgesamt resultiert also eine Prioritätsliste mit einer Auswahl von Entitätserwähnungen der besten Abdeckung und sicheren Kandidatenentitäten-Sets.

Die Entitätserwähnungen der Prioritätslisten werden anschließend mit dem Schwellenwertverfahren gefiltert. Nur Entitätserwähnungen, die Kandidatenentitäten mit einem hohen Rankingwert beinhalten, bleiben in der Prioritätsliste bestehen. Dafür wird die Kandidatenentität mit dem höchsten Rankingwert einer Entitätserwähnung identifiziert und mit dem Schwellenwert ϵ verglichen. Wenn der Rankingwert der Kandidatenentität überhalb des Schwellenwerts ϵ liegt, wird diese Kandidatenentität der Entitätserwähnung fest zugewiesen.

Das Ergebnis der *Unlinkable-Mention-Prediction* ist eine Prioritätsliste für jeden Satz. Die Prioritätsliste enthält alle Entitätserwähnungen, die zu einer Entität in der Wissensbasis zugeordnet wurden. Das Entity Linking ist nach der Ausführung der *Unlinkable-Mention-Prediction* abgeschlossen. Aus den Prioritätslisten können die Entitätserwähnungen entnommen werden und es steht fest, zu welchen Entitäten der Wissensbasis sie zugehörig sind.

11 Evaluation

In diesem Kapitel werden die einzelnen Module und das gesamte Entity-Linking-System evaluiert. Für die Evaluation verwenden wir einen GQM-Plan. Die Auswahl und konkrete Werte der Metriken werden in den anschließenden Sektionen diskutiert. Sektion 11.1 enthält die Evaluationsergebnisse der Wissensbasis. Sektion 11.2 enthält die Evaluationsergebnisse der *Named-Entity-Recognition*. Abschließend enthält Sektion 11.3 die Evaluationsergebnisse des gesamten Entity-Linking-Prozesses. Der GQM-Plan beschreibt **G**oals (Ziele), **Q**uestions (Fragen) und **M**etrics (Metriken).

Das Hauptziel der Ausarbeitung ist die Erstellung einer Wissensbasis für die Domäne der Softwareentwicklung und die Entwicklung eines Entity-Linking-Systems. Das Entity-Linking-System wird verwendet, um Entitätserwähnungen einer Softwarearchitekturdokumentation zu den Entitäten der erstellten Wissensbasis zu verlinken.

1. Teilziel

Die Wissensbasis soll möglichst viele Entitäten aus der Domäne der Softwareentwicklung enthalten.

1.1. Frage

Wie hoch ist der Anteil der Entitätserwähnungen aus der Softwarearchitekturdokumentation, die in der Wissensbasis enthalten ist?

1.1.1. Metrik

$$\text{Abdeckung} = \frac{\text{Enthaltene Entitätserwähnungen der Architekturdokumentation in der Wissensbasis}}{\text{Entitätserwähnungen in der Dokumentation aus der Domäne der Softwareentwicklung}}$$

2. Teilziel

Finde Entitätserwähnungen aus der Domäne der Softwareentwicklung in der Softwarearchitekturdokumentation.

2.1. Frage

Wie ist das Verhältnis aus der Anzahl der gefundenen Entitätserwähnungen in der Softwarearchitekturdokumentation zu den vorhandenen Entitätserwähnungen?

2.1.1. Metrik

$$\text{Ausbeute} = \frac{\text{Korrekt identifizierte Entitätserwähnungen}}{\text{Entitätserwähnungen im Text}}$$

2.2. Frage

Wie hoch ist der Anteil der korrekt identifizierten Entitätserwähnungen im Verhältnis zu der Anzahl der insgesamt identifizierten Entitätserwähnungen?

2.2.1. Metrik

$$\text{Präzision} = \frac{\text{Korrekt identifizierte Entitätserwähnungen}}{\text{Alle identifizierten Entitätserwähnungen}}$$

Zusätzlich zu den Metriken 2.1.1. und 2.2.1. berechnen wir den F_1-Wert als harmonisches Mittel zwischen Präzision und Ausbeute.

$$F_1\text{-Wert} = \frac{(2 * \text{Präzision} * \text{Ausbeute})}{(\text{Präzision} + \text{Ausbeute})}$$

3. Teilziel

Verlinke gefundene Entitätserwähnungen mit den zugehörigen Entitäten der Wissensbasis.

3.1. Frage

Wie oft findet das System die passende Entität in der Wissensbasis für eine Entitätserwähnungen im Text?

3.1.1. Metrik

$$\text{Ausbeute} = \frac{\text{Korrekt verlinkte Entitätserwähnungen}}{\text{Entitätserwähnungen, die verlinkt werden sollen}}$$

3.2. Frage

Wie ist das Verhältnis der korrekt verlinkten Entitätserwähnungen zu den Entitätserwähnungen, die das System insgesamt verlinkt hat.

3.2.1. Metrik

$$\text{Präzision} = \frac{\text{Korrekt verlinkte Entitätserwähnungen}}{\text{Alle verlinkten Entitäten}}$$

Zusätzlich zu den Metriken 3.1.1. und 3.2.1. berechnen wir den F_1-Wert als harmonisches Mittel zwischen Präzision und Ausbeute.

$$F_1\text{-Wert} = \frac{(2 * \text{Präzision} * \text{Ausbeute})}{(\text{Präzision} + \text{Ausbeute})}$$

Für die Evaluation der Module, die im Rahmen dieses Projektes ausgearbeitet wurden, werden Architekturdokumentationen verwendet. Bei den Dokumentationen handelt es sich um dieselben Dokumentationen wie bei dem Forschungsprojekt ArDoCo [15]. Die vier Dokumentationen enthalten die folgenden Fallstudien:

MediaStore

MediaStore ist eine Modellanwendung, die nach dem Vorbild des iTunes Stores aufgebaut ist. Die verwendete Dokumentation wurde aus der Veröffentlichung *Modeling and Simulating Software Architectures: The Palladio Approach* [25] entnommen.

BigBlueButton

Die BigBlueButton Dokumentation beschreibt eine nichtwissenschaftliche Anwendung. Die Anwendung stellt ein Webkonferenzsystem als globale Lehrplattform bereit.

Teammates

Die Anwendung Teammates ist ein quelloffenes Online-Tool. In dem Repository von Teammates wird die Anwendung als Tool zur Verwaltung von Peer-Bewertungen und anderen Feedback-Wegen von Schülern beschrieben. Die Dokumentation von Teammates ist Teil des Repositories.

Teastore

Teastore ist eine wissenschaftliche Anwendung, die als *Microservice-Referenz-Anwendung* verwendet wird. Die Teastore Dokumentation ist im zugehörigen Repository enthalten.

Die Dokumente sind als Textdateien gespeichert und enthalten zwischen 37 und 189 Zeilen. Sie enthalten Abkürzungsdefinitionen und Entitäten der Softwareentwicklung. Die vier Fallstudien unterscheiden sich in der Projekt- und Dokumentationsgröße sowie den Architekturstilen. Mit diesen *Benchmark-Dokumentationen* glauben wir, eine repräsentative Auswahl getroffen zu haben.

Die Gold-Standards entsprechen Musterlösungen für das Ergebnis eines Entity-Linking-Systems der ausgewählten Dokumentationen. Die Gold-Standards sind im JSON-Dateiformat und folgen einer vordefinierten Struktur. In der Struktur hat jede Entitätserwähnung einen eigenen Eintrag mit Index des zugehörigen Satzes und der relativen Position innerhalb des Satzes. Für die Position innerhalb des Satzes wurden die Tokenpositionen aus dem *CoreNLP-Tool* von Stanford verwendet. Für jede Entitätserwähnung wurde überprüft, ob sie eine zugehörige Entität in der Wissensbasis hat. Die identifizierten Entitäten aus der Wissensbasis werden zur Entitätserwähnung im Gold-Standard hinzugefügt. Falls für eine Entitätserwähnung keine Entität in der Wissensbasis identifiziert wurde, bleibt die Entitätserwähnung ohne zugehörige Entitäten im Gold-Standard bestehen.

Mit diesem Vorgehen haben mehrere Forscher, jeweils unabhängig voneinander, einen Gold-Standard für unsere Fallstudien erarbeitet. Die daraus resultierenden Gold-Standards wurden kombiniert. Auftretenden Unterschiede wurden diskutiert und bereinigt. Das System wurde anschließend intrinsisch evaluiert, wobei die Metriken aus dem GQM-Plan entnommen wurden.

11.1 Evaluationsergebnisse der Wissensbasis

Für die Evaluation der Wissensbasis haben wir die Abdeckung (1.1.1.) im GQM-Plan als entscheidende Metrik identifiziert. Die Wissensbasis besteht aus extrahierten Informationen des IEEE 24765:2017 [14] und enthält somit nur Begriffe und Abkürzungen aus der Domäne der Softwareentwicklung. Um die Abdeckung der Wissensbasis zu bestimmen, muss für eine Liste an Begriffen der Softwareentwicklung überprüft werden, wie viele in der Wissensbasis enthalten sind. Damit die Liste echte Begriffe der Softwareentwicklung enthält, verwenden wir die Entitätserwähnungen der Architekturdokumentationen. Für die Auswertung der Abdeckung wurden alle Entitätserwähnungen in einer Architekturdokumentation markiert. Dieser Schritt wurde händisch vorgenommen, um mögliche Fehler durch das *NER-System* zu vermeiden. Anschließend wurde für jede Entitätserwähnung die passenden Entitäten in der Wissensbasis identifiziert. Falls mindestens eine Entität in der Wissensbasis passend für die Entitätserwähnung ist, wird die Entitätserwähnung als *enthalten* gewertet. *Nicht enthaltene* Entitätserwähnungen haben keine passende Entität in der Wissensbasis. Mit dieser Vorgehensweise wurde die Abdeckung separat für jede Architekturdokumentation überprüft. Anschließend haben wir den gewichteten Mittelwert bezüglich der Anzahl der Entitätserwähnungen berechnet. Die Ergebnisse befinden sich in Tabelle 11.1.

Tabelle 11.1: Abdeckung der Wissensbasis für die *Benchmark-Dokumentationen.*

Dokumentation	Anzahl	Gefunden	Abdeckung
MediaStore	154	132	0.8571
BigBlueButton	268	205	0.7649
Teammates	656	572	0.8719
Teastore	167	132	0.7904
Gesamt	1245	1041	0.8361

Fehlende Begriffe in der Wissensbasis haben eine große Auswirkung auf das anschließende Entity-Linking-System. Da das Entity-Linking-System Entitätserwähnungen mit den Entitäten der Wissensbasis verbindet, führen fehlende Begriffe in der Wissensbasis zu fehlenden Verbindungen im Ergebnis des Entity Linkings. Insgesamt ist die Abdeckung also eine wichtige Metrik um die Brauchbarkeit der Wissensbasis für das Entity-Linking-System zu messen. Die Wissensbasis hat mit einer Abdeckung von 83.61% auf den Dokumentationen, ein hohes Verbesserungspotenzial.

11.2 Evaluationsergebnisse der Named-Entity-Recognition

Für die Evaluation der *Named-Entity-Recognition* haben wir im GQM-Plan die Ausbeute (2.1.1.), Präzision (2.2.1.) und den F_1-Wert als entscheidende Metriken identifiziert. Da die *NER* nur ein Vorverarbeitungsschritt für das anschließende Entity Linking ist, sind die Ziele spezieller als bei einem eigenständigem *NER-System.* Der wesentliche Unterschied besteht darin, dass unser *NER-System* für die Vorverarbeitung möglichst viele Entitätserwähnungen im Text enthalten soll, auch wenn dafür mehr *false positives* identifiziert werden. Dieser Unterschied resultiert aus der Tatsache, dass ein *false positive* in der weiteren Verarbeitung von dem Entity-Linking-System aussortiert werden kann. Im Unterschied dazu, ist es nicht möglich, nicht identifizierte Entitätserwähnung im anschließenden Entity Linking nachträglich zu identifizieren. Insgesamt folgt daraus, dass das *NER-System* wenig restriktiv sein soll. Folglich erstellt das *NER-System* bei Unsicherheit einen Eintrag für eine Entitätserwähnung. Die Ausbeute ist somit eine deutlich wichtigere Metrik im Vergleich zur Präzision.

Die Tabelle 11.2 enthält die Metriken Präzision, Ausbeute und F_1-Wert für die einzelnen Architekturdokumentationen. Außerdem sind in der letzten Zeile die gewichteten Mittelwerte der Metriken enthalten. Die Gewichtung entspricht der Anzahl der Entitäten in einer Architekturdokumentation, im Verhältnis zur Anzahl der Entitäten aller Architekturdokumentationen.

Tabelle 11.2: Ausbeute, Präzision und F_1-Wert der *Named-Entity-Recognition* für die *Benchmark-Dokumentationen.*

Dokumentation	Ausbeute	Präzision	F_1-Wert
MediaStore	0.9935	0.4513	0.6206
BigBlueButton	0.9701	0.3147	0.4753
Teammates	0.9832	0.3974	0.5660
Teastore	0.9940	0.4181	0.5886
Gesamt	0.9831	0.3890	0.5562

Ausbeute

Da die Ausbeute die entscheidende Metrik für das *NER-System* ist, haben wir das System auf diese Metrik optimiert. Damit werden möglichst viele Entitätserwähnungen im Text identifiziert. Falls eine Entitätserwähnung nicht von dem *NER-System* identifiziert wird, wird diese auch nicht im anschließenden Entity-Linking-System betrachtet und verarbeitet.

Die größte Fehlerquelle, die eine höhere Ausbeute verhindert, sind Mehrdeutigkeiten der englischen Sprache. Dazu ist das folgende Beispiel in Abbildung 11.1 gegeben. In diesem Beispiel wird die Entitätserwähnung *functions* nach den Wortgruppen in Abbildung 6.1 als *VBZ* kategorisiert. Der *VBZ*-Tag beschreibt Verben in der dritten Person Singular im Präsens. Die Entitätserwähnung *functions* steht in diesem Zusammenhang jedoch nicht als Verb, sondern als Nomen für das Plural von Funktionen. Diese fehlerhafte Kategorisierung führt dazu, dass die Entitätserwähnung nicht als solche identifiziert wird.

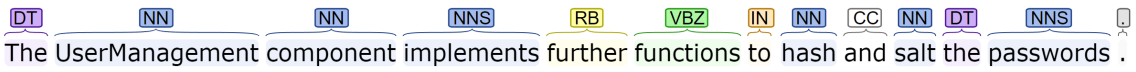

Abbildung 11.1: Mehrdeutigkeit in der *Named-Entity-Recognition*

Präzision

Die Präzision unseres *NER-Systems* ist im Vergleich zur Ausbeute deutlich niedriger. Begründet werden kann dies mit der bekannten Problematik, dass Präzision und Ausbeute nur schwer zusammen optimiert werden können. Im Rahmen der Entwicklung unseres *NER-Systems* wurde der Fokus auf die Optimierung der Ausbeute gelegt, sodass die Relevanz der Präzision geringer ist. Aus diesem Grund ist ein verhältnismäßig niedriger Präzisionswert für die *Named-Entity-Recognition* akzeptabel. Starke Optimierungen der Präzision hätten einen negativen Effekt auf die Ausbeute, was in der Vorverarbeitung in unserem System nicht erwünscht ist.

11.3 Evaluationsergebnisse des Entity-Linking-Systems

In dieser Sektion evaluieren wir das Entity-Linking-System. Dafür haben wir das System auf den ausgewählten *Benchmark-Dokumentationen* getestet.

Für die Evaluation des Entity-Linking-Systems haben wir im GQM-Plan die Ausbeute (3.1.1.), Präzision (3.2.1.) und den F_1-Wert als entscheidende Metriken identifiziert. Die Metriken des GQM-Plans evaluieren wir bezüglich verschiedener Konfigurationen des Entity-Linking-Systems.

Die Konfigurationen unterscheiden wir anhand der Festlegung des Wertes ϵ. Der ϵ-Wert wird in der *Unlinkable-Mention-Prediction* in Kapitel 10 für das Schwellenwertverfahren verwendet. Das Schwellenwertverfahren entscheidet, ob ein Link von einer Entitätserwähnung zur identifizierten Kandidatenentität der Wissensbasis erstellt werden soll. Der Rankingwert der Kandidatenentität wird dabei mit dem Schwellenwert $\epsilon \in [0, \infty)$ verglichen.
Mit dem Ziel, einen optimalen Schwellenwert für unsere Rankingfunktion zu finden, haben wir verschiedene ϵ-Werte mit einer Schrittgröße von 0.25 überprüft. Die Ergebnisse bezüglich verschiedener ϵ-Werte sind für die Dokumentation MediaStore in Abbildung 11.2, BigBlueButton in Abbildung 11.3, Teammates in Abbildung 11.4 und Teastore in Abbildung 11.5 graphisch illustriert.

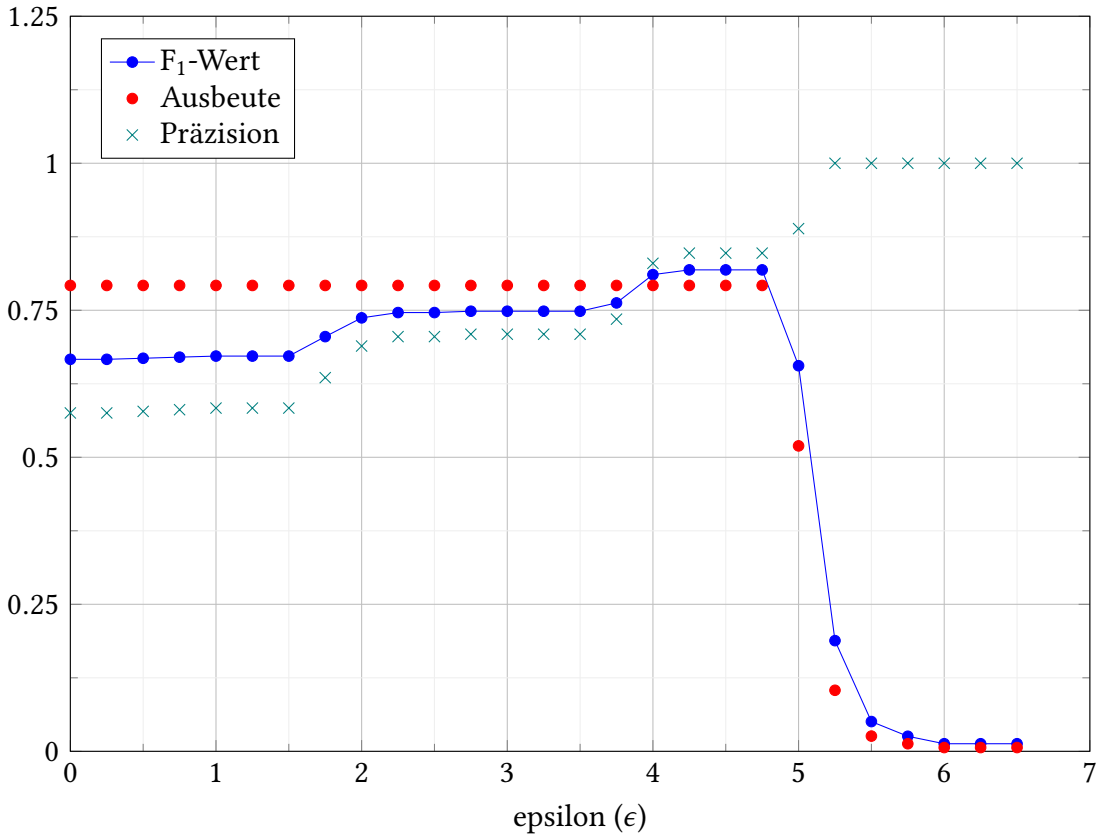

Abbildung 11.2: Präzision, Ausbeute und F_1-Wert bezüglich verschiedener ϵ-Werte für die Dokumentation MediaStore.

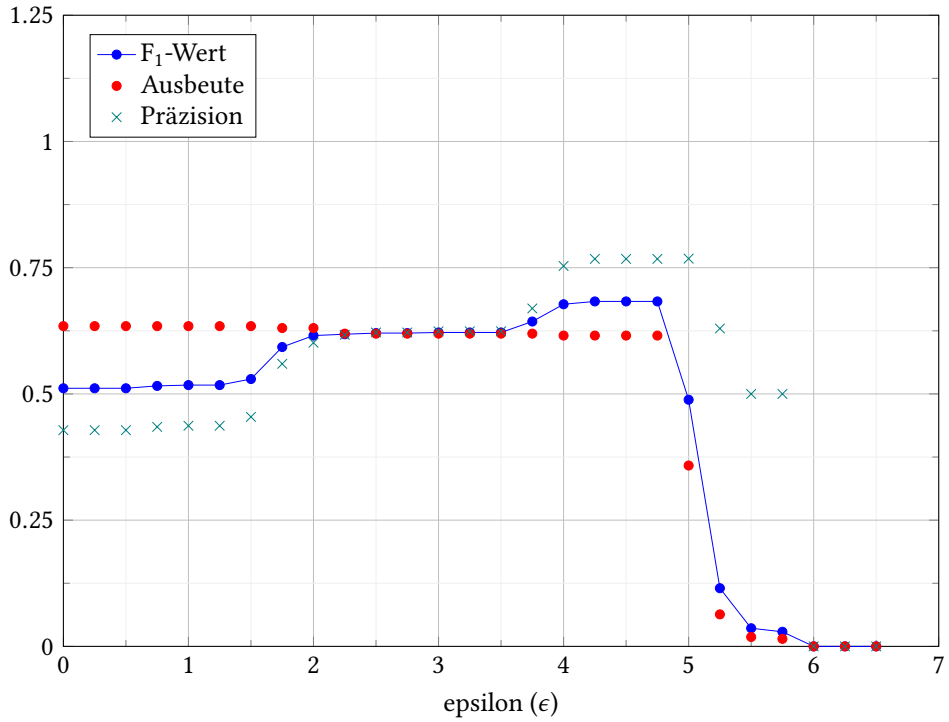

Abbildung 11.3: Präzision, Ausbeute und F_1-Wert bezüglich verschiedener ϵ-Werte für die Dokumentation BigBlueButton.

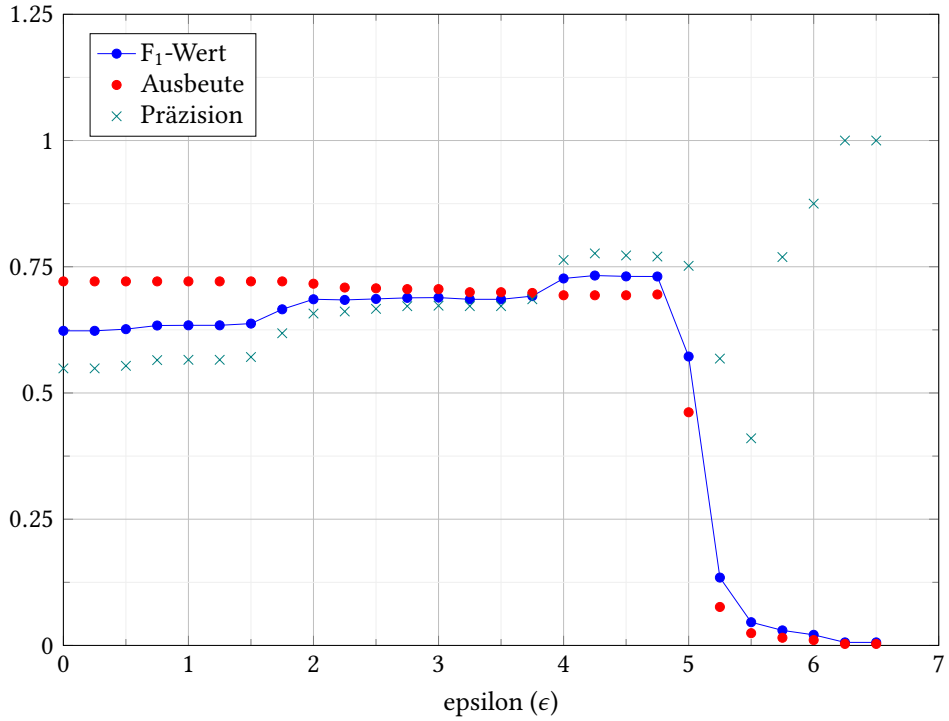

Abbildung 11.4: Präzision, Ausbeute und F_1-Wert bezüglich verschiedener ϵ-Werte für die Dokumentation Teammates.

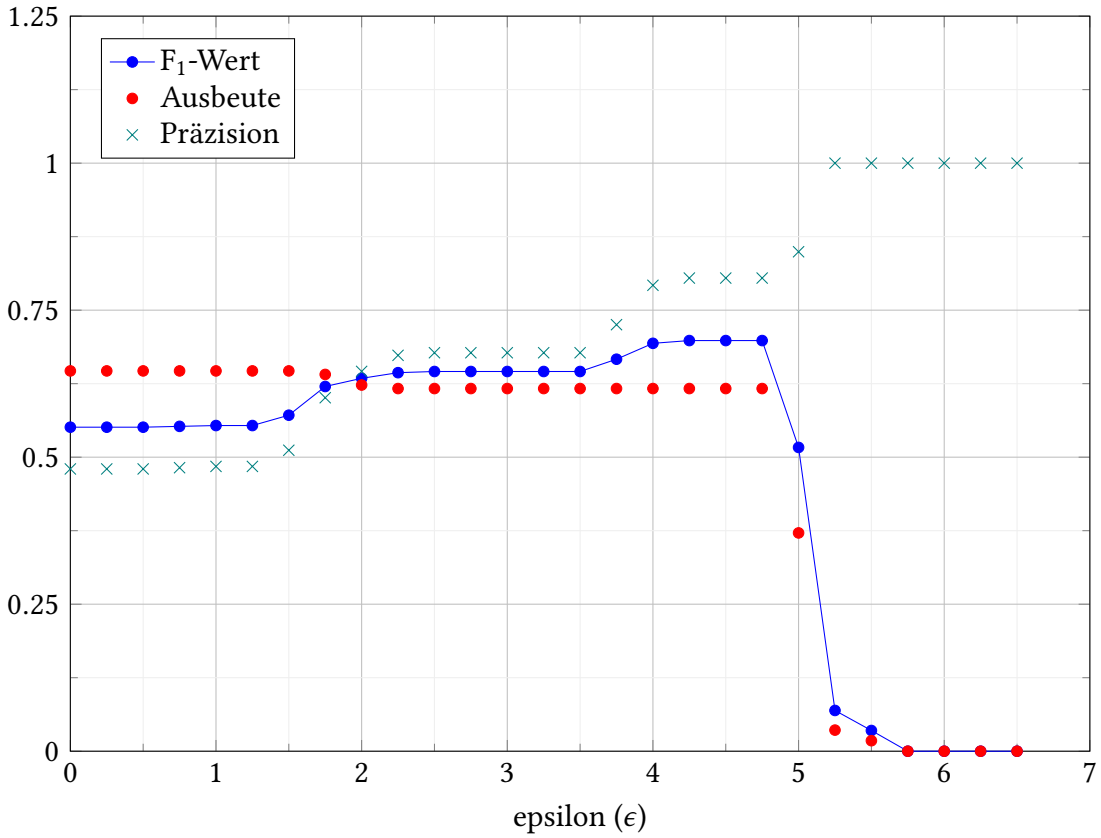

Abbildung 11.5: Präzision, Ausbeute und F_1-Wert bezüglich verschiedener ϵ-Werte für die Dokumentation Teastore.

Aus den Evaluationsergebnissen der Abbildungen lässt sich entnehmen, dass aus niedrigeren ϵ-Werten, eine höhere Ausbeute und niedrigere Präzisionen resultieren. Mit steigendem ϵ-Wert verringert sich die Ausbeute und erhöht sich die Präzision. Die Verschlechterung der Ausbeute ist deutlich geringer als die Verbesserung der Präzision. Aus diesem Grund erhöht sich mit steigendem ϵ-Wert der F_1-Wert. Im Bereich $\epsilon \in [4.25, 4.75]$ halten sich die Präzision und Ausbeute stabil und erreichen in allen Dokumentationen den höchsten F_1-Wert bezüglich aller ϵ-Werte. Der F_1-Wert bricht schließlich bei $\epsilon = 5.00$ ein, da sich die Ausbeute extrem verringert. Auf Grundlage dieser Daten setzen wir $\epsilon = 4.50$ fest. Der Wert $\epsilon = 4.50$ befindet sich innerhalb des optimalen Intervalls $[4.25, 4.75]$ und hat einen sicheren Abstand zum Einbruchswert bei $\epsilon = 5.00$.

Die Konfiguration mit $\epsilon = 4.50$ betrachten wir genauer und vergleichen diese anschließend mit einem anderen Entity-Linking-System. Zunächst unterscheiden wir, wie das Entity-Linking-System evaluiert werden soll. In den Sektionen 11.1 und 11.2 haben wir die Wissensbasis und die *Named-Entity-Recognition* isoliert evaluiert. Mit einer isolierten Evaluation beschreiben wir die Auswertung der Metriken ohne (Stör-)Einflüsse der Umgebung bzw. anderer Module. Die Evaluation des Entity-Linking-Systems können wir ebenfalls isolierten vornehmen. In der isolierten Evaluation des Entity-Linking-Systems setzen wir eine vollständige Wissensbasis und Vorverarbeitung voraus. Eine vollständige Vorverarbeitung entspricht einer *Named-Entity-Recognition*, die alle Entitätserwähnung

erkennt. Vollständigkeit innerhalb der Wissensbasis beschreibt, dass alle korrekt identifizierten Entitätserwähnungen eine zugehörige Entität in der Wissensbasis enthalten. Unvollständigkeiten innerhalb der Wissensbasis oder Vorverarbeitung werden erkannt, gefiltert und beeinflussen anschließend nicht die Auswertung der Metriken für das Entity-Linking-System. Durch dieses Vorgehen erreichen wir eine Auswertung der Metriken des Entity-Linking-Systems, die möglichst wenig von der Wissensbasis und Vorverarbeitung beeinflusst wird. Alternativ kann der gesamte Ablauf inklusive der Vorverarbeitung und Wissensbasis evaluiert werden.

Für den Wert $\epsilon = 4.5$ sind die Metriken der isolierten Evaluation in der Tabelle 11.3 gegeben. Tabelle 11.4 enthält die Metriken des gesamten Ablaufs für den Wert $\epsilon = 4.50$. Die Tabellen enthalten die gewichteten Mittelwerte der Metriken bezüglich der Anzahl an Entitäten in den Dokumentationen.

Tabelle 11.3: Ausbeute, Präzision und F_1-Wert des Entity-Linking-Systems bei isolierter Evaluation mit $\epsilon = 4.5$ für die *Benchmark-Dokumentationen*.

Dokumentation	Ausbeute	Präzision	F_1-Wert
MediaStore	0.9384	0.8472	0.8905
BigBlueButton	0.8684	0.7674	0.8148
Teammates	0.8425	0.7724	0.8060
Teastore	0.8373	0.8046	0.8207
Gesamt	0.8595	0.7853	0.8207

Tabelle 11.4: Ausbeute, Präzision und F_1-Wert des Entity-Linking-Systems bei dem gesamten Ablauf mit $\epsilon = 4.5$ für die *Benchmark-Dokumentationen*.

Dokumentation	Ausbeute	Präzision	F_1-Wert
MediaStore	0.7922	0.8472	0.8187
BigBlueButton	0.6156	0.7674	0.6832
Teammates	0.6935	0.7724	0.7309
Teastore	0.6167	0.8046	0.6983
Gesamt	0.6786	0.7848	0.7271

Insgesamt folgen die Ergebnisse der isolierten Evaluation in Tabelle 11.3 der gleichen Struktur wie die Ergebnisse des gesamten Ablaufs in Tabelle 11.4. Ausschließlich die Ausbeute ist im isolierten Ablauf konstant höher als im gesamten Ablauf und verbessert somit auch den F_1-Wert. Die isolierte Evaluation hat im Vergleich zum gesamten Ablauf eine 18.09% ($0.8595 - 0.6786 = 0.1809$) höhere Ausbeute. Aus diesem Wert wird gefolgert, dass eine Vervollständigung der Entitäten innerhalb der Wissensbasis und eine Vorverarbeitung, die alle Entitätserwähnungen identifiziert, die Ausbeute des gesamten Systems um 18.09% verbessern würde.

Die Werte aus der Tabelle 11.4 des gesamten Ablaufs verwenden wir, um unser System mit anderen Entity-Linking-Systemen zu vergleichen. Wir haben unser System nicht mit

General-Purpose-Systemen verglichen, da eine Vergleichbarkeit zu domänenunabhängigen Systemen nur eine bedingte Aussagekraft hat. Stattdessen betrachten wir die Evaluationsergebnisse anderer domänenspezifischer Entity-Linking-Systeme. Für den Vergleich haben wir das System *Domain-specific entity linking via fake named entity detection* von Zhang et al. [37] verwendet. Details zum domänenspezifischen System von Zhang et al. sind in Kapitel 3 gegeben. Das System von Zhang et al. wurde von den Autoren mit vier verschiedenen Konfigurationen ausgeführt und evaluiert. Für die Evaluation wurde ein Gold-Standard aus Filmkommentaren und Dokumenten von Nutzern erstellt. Verlinkt wurden die Entitätserwähnungen zur Wissensbasis MKB (*Movie-Knowledge-Base*) der Universität Tsinghua in Peking.

Die Ergebnisse für die einzelnen Konfigurationen werden ebenfalls separat mit den Metriken Ausbeute, Präzision und F_1-Wert gemessen. Wir nehmen die Werte der Metriken bezüglich der Konfigurationen und repräsentieren diese als Intervalle. Beispielsweise hat die Konfiguration mit der schlechtesten Ausbeute den Wert 0.627 und die Konfiguration mit der besten Ausbeute den Wert 0.690. Daraus resultiert der Intervall Ausbeute $\in [0.627, 0.690]$. In diesem Intervall liegen die Werte der Ausbeuten aller vier Konfigurationen des Systems. Die Evaluationswerte der vier Konfigurationen sind in den folgenden Intervallen enthalten.

$$\begin{aligned} \text{Ausbeute} &\in [0.627, 0.690] \\ \text{Präzision} &\in [0.646, 0.741] \\ F_1\text{-Wert} &\in [0.636, 0.714] \end{aligned} \tag{11.1}$$

Diese Intervalle haben wir mit den gewichteten Durschnittswerten unseres Systems aus Tabelle 11.4 verglichen. Die Ausbeute unseres Systems liegt innerhalb des Ausbeute-Intervalls des Vergleichsystems ($0.6786 \in [0.627, 0.690]$). Daraus lässt sich schließen, dass wir ähnliche Ergebnisse bezüglich der Ausbeute erzielen. Die Präzision unseres Systems ist dafür aber deutlich höher als die beste Präzision des Systems von Zhang et al.. Unser System erreicht einen Präzisionswert von 0.7848 und liegt damit überhalb aller Präzsionswerte der Konfigurationen des Vergleichsystems. Durch den deutlich besseren Präzisionswert erreicht unser System ebenfalls einen höheren F_1-Wert. Der F_1-Wert unseres Systems ist mit 0.7271 besser als der F_1-Wert aller Konfigurationen des Vergleichsystems. Insgesamt verbessert unser System also besonders die Präzision im Vergleich zum System von Zhang et al. [37]. In Abbildung 11.6 sind die F_1-Werte unseres Systems für die vier *Benchmark-Dokumentationen* in einer Graphik zusammengefasst.

Die Ergebnisse der Evaluation sind vielversprechend und bestärken weitere Forschungen im Bereich domänenspezifischer Entity-Linking-Systeme. Nach unseren Einschätzungen ist die Ausbeute von 98.31% in der *Named-Entity-Recognition* bereits ein sehr gutes Ergebnis. Außerdem übertrifft die Präzision unseres Entity-Linking-Systems mit 78.48% die Präzisionswerte des System von Zhang et al. [37]. Auch der F_1-Wert von 72.71% ist in unserem System höher als die F_1-Werte des Vergleichsystems. Das hier vorgestellte System ist jedoch nicht fehlerfrei und kann in vielen Bereichen verbessert werden.

In dieser Arbeit bietet besonders die Abdeckung der Wissensbasis starkes Verbesserungspotenzial. Die Wissensbasis limitiert mit der aktuellen Abdeckung von 83.61% die Ausbeute des Entity-Linking-Systems. Die Limitierung durch die Wissensbasis wird durch die Auswertung der isolierten Evaluation in Tabelle 11.3 bestätigt.

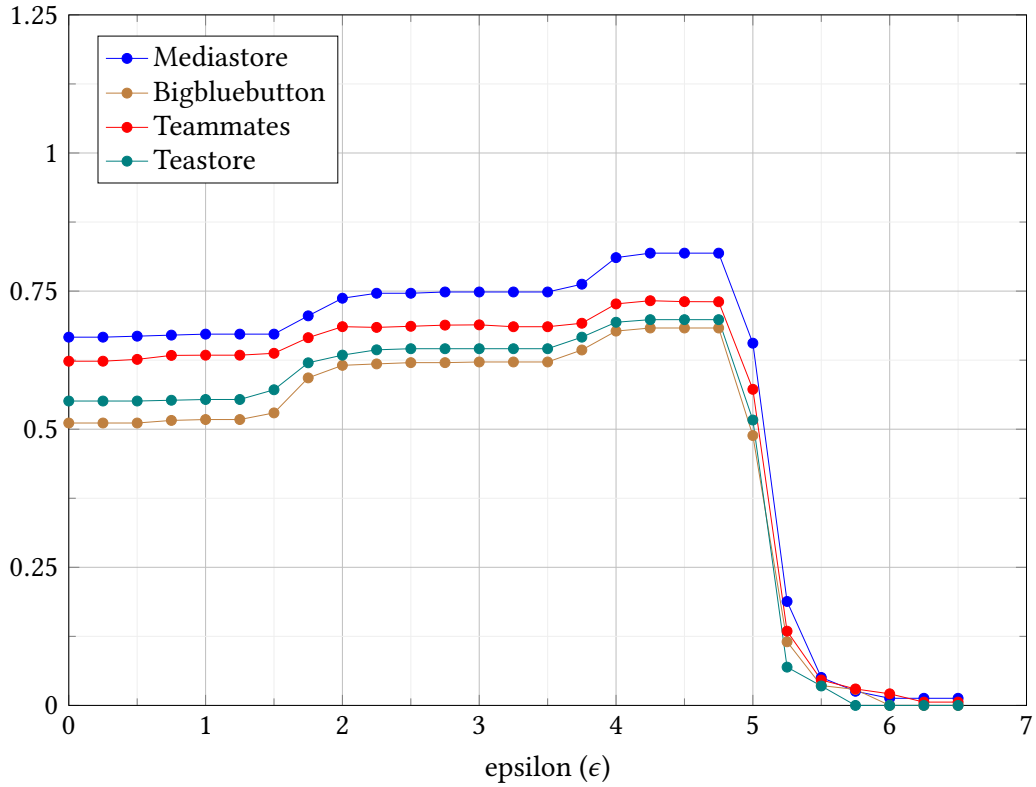

Abbildung 11.6: F_1-Wert bezüglich verschiedener ϵ-Werte.

Bei einer vollständigen Eingabe ohne Unvollständigkeiten in der Wissensbasis und Vorverarbeitung hat das Entity-Linking-System eine Ausbeute von 85.95% erreicht. Wenn es also möglich ist, die Wissensbasis zu vervollständigen, kann dies die Ausbeute des Entity-Linking-Systems deutlich verbessern und damit auch das Ergebnis des F_1-Wertes.

Insgesamt schätzen wir die Ergebnisse als vielversprechend ein, haben jedoch auch Verbesserungspotenzial identifiziert. Vorschläge möglicher Verbesserungen befinden sich in Kapitel 13 bezüglich zukünftiger Arbeiten.

12 Gefährdung der Validität

In diesem Kapitel wird die Gefährdung der Validität des hier entwickelten Systems geprüft. Wir unterteilen die Gefährdung der Validität in die Kategorien von Runeson und Höst [26]. Die vier Kategorien sind *Konstrukt*, *Intern*, *Extern* und *Zuverlässigkeit*, wie in [26] definiert. Das Kapitel zur Gefährdung der Validität orientiert sich an dem System zur *Trace Link Recovery for Software Architecture Documentation* von Keim et al. [16].

Konstruktvalidität

Wir haben häufig verwendete experimentelle Designs und Metriken aus der Entity-Linking-Community angewendet. Durch die Verwendung klassischer Metriken versuchen wir potenzielle Risiken in Bezug auf die Konstruktvalidität zu minimieren. Für die Evaluation verwenden wir dieselben Architekturdokumentationen wie das Forschungsprojekt ArDoCo [15]. Die vier Fallstudien haben unterschiedliche Projekt- und Dokumentationsgrößen sowie Architekturstile. Mit dieser Auswahl glauben wir, eine repräsentative Auswahl an Dokumentationen zu haben, um die Konstruktvalidität sicherzustellen. Allerdings handelt es sich dabei um öffentlich zugängliche *Open-Source*-Systeme. Dokumentationen anderer *nicht-Open-Source*-Projekte können abweichen.

Interne Validität

In der Evaluation haben wir die bereitgestellten Dokumentationstexte analysiert und die identifizierten Entitätserwähnungen mit den Entitäten der Wissensbasis verglichen. Dieses Vorgehen setzt voraus, dass die Entitäten der Architekturdokumentation den Entitäten der Wissensbasis entsprechen. Bei der Erstellung der Wissensbasis haben wir die Auswahl der Quelle als mögliche Bedrohung identifiziert. In der Domäne der Softwareentwicklung gibt es möglicherweise umfangreichere Quellen mit einer besseren Struktur. Es möglich, dass andere Quellen eine bessere Auswahl von Entitäten enthalten, die mit den Entitätserwähnungen der Architekturdokumentationen besser übereinstimmen. Außerdem wurde die Quelle nicht auf Fehler überprüft und kann somit Fehler in die Wissensbasis übertragen. Sowohl Fehler in der Quelle als auch eine fehlerbehaftete Informationsextraktion können Störfaktoren für die interne Validität sein, da sie einen willkürlichen Störeinfluss auf die Messungen in der Evaluation haben können.

Wir haben diesem Faktor entgegengewirkt, indem wir verschiedene Quellen analysiert und miteinander verglichen haben. Außerdem haben wir nach der Informationsextraktion eine regelbasierte Überprüfung der Entitäten unternommen, um Ausreißereinträge zu finden und zu verbessern.

Externen Validität

In unserer Evaluation haben wir vier Fallstudien untersucht, die sorgfältig von uns ausgewählt wurden. Die Fallstudien MediaStore und Teastore stammen aus dem Umfeld der Forschung. Teammates wird in der Praxis verwendet, stammt jedoch auch aus dem universitären Kontext. Die Anwendung BigBlueButton kommt nicht aus der Forschung. Trotz der sorgfältigen Auswahl der Architekturdokumentationen besteht die Gefahr, dass nicht alle Aspekte und Facetten des Entity-Linking-Problems durch diese Fallstudien abgedeckt werden. Aufgrund einer möglichen Abweichung zwischen den hier untersuchten und anderen Architekturdokumentationen, ist eine Abweichung unserer Ergebnisse nicht ausgeschlossen. Um die externe Gültigkeit des Systeme zu analysieren, haben wir die Übertragbarkeit des Systems auf einen Austausch der Sprache und der Domäne näher untersucht.

Austauschbarkeit der Sprache

Die Vorverarbeitung ist nicht unabhängig von der Sprache. Auch das *CoreNLP*-Tool von Stanford ist nicht für alle Sprachen verfügbar. Ein Austausch der Sprache benötigt außerdem einen Austausch der Wissensbasis. Die Funktionsweise des gesamten Entity-Linking-Systems ist nicht auf verschiedene Sprachen generalisierbar und auch die Evaluationsergebnisse können nicht auf andere Sprachen übertragen werden.

Austauschbarkeit der Domäne

Die Vorverarbeitung ist domänenunabhängig und somit auf alle Domänen übertragbar. Die Wissensbasis muss entsprechend der Domänen ausgetauscht oder angepasst werden. Das Entity-Linking-System ist auf andere Domänen übertragbar und enthält keine domänenspezifischen Anpassungen. Aus diesem Grund gehen wir davon aus, dass die Evaluationsergebnisse auf andere Domänen übertragbar sind. Das System wurde jedoch nicht bezüglich anderer Domänen getestet oder evaluiert.

Zuverlässigkeit

Für die Evaluation dieser Arbeit mussten wir eigenständig einen Gold-Standard für das Entity Linking erstellen. Mehrere Forscher haben jeweils unabhängig voneinander einen Gold-Standard für unsere Fallstudien erstellt. Diese Gold-Standards wurden kombiniert und die auftretenden Unterschiede diskutiert. Auf diese Weise haben wir versucht, eine Verzerrung durch einzelne Individuen zu minimieren. Dennoch ist eine gewisse Verzerrung nicht ausgeschlossen.

13 Schlussfolgerung und zukünftige Arbeiten

Von Verlinkung zwischen Entitätserwähnungen und Entitäten in einer Wissensbasis können sowohl menschliche Leser als auch automatisierte Textverarbeitungssysteme profitieren. Insbesondere werden Entity-Linking-Systeme zur Vorverarbeitung für eine Vielzahl an *Natural-Language-Processing-Systemen* verwendet.

Unser System ist ein domänenspezifischer Ansatz, wobei die Entitäten aus der Domäne der Softwareentwicklung sind. Bei den Texten, die analysiert werden sollen, handelt es sich um Architekurdokumentationen aus Softwareprojekten. Verlinkungen zwischen Entitätserwähnungen im Text und Entitäten in einer Wissensbasis zu identifizieren, ist keine triviale Aufgabe. Das System muss mit limitiertem Kontext der Entitätserwähnungen entscheiden, ob und welche Entität in der Wissensbasis zugehörig ist. Außerdem ist der Aufbau einer Wissensbasis bezüglich der Domäne der Softwareentwicklung potenziell sehr umfangreich, da die Wissensbasis beliebig erweitert und verbessert werden kann.

In unserem Ansatz haben wir zunächst eine Wissensbasis erstellt und anschließend ein Modul zur Vorverarbeitung für das Entity-Linking-System implementiert. Das Entity-Linking-System ist in drei Module unterteilt und findet Links der Entitätserwähnungen im Text zu Entitäten der selbsterstellten Wissensbasis.

Wir haben unser System mithilfe von vier Dokumentationen getestet. Die Dokumentationen entsprechen den *Benchmark-Dokumentationen* des Forschungsprojekts ArDoCo [15]. Mit unserem Ansatz erreichen wir durchschnittlich einen gewichteten F_1-Wert von 0.7271.

Durch den modularen Aufbau ist das System einfach erweiterbar. Neue Module können in das System integriert werden und bestehende Module sind austauschbar. Eine Kategorisierung der möglichen Verbesserungen für unser System, um die Leistungsfähigkeit unseres Ansatzes zu erhöhen, ist im Folgenden gegeben.

Wissensbasis

Zur Erstellung der Wissensbasis wurde nur eine Quelle für die Population der Wissensbasis verwendet. Durch die Hinzunahme weiterer Quellen könnten mehr Entitäten in die Wissensbasis integriert werden. Außerdem wurde die aktuelle Quelle nicht auf interne Fehler überprüft. Fehler innerhalb der Quelle werden somit in die Wissensbasis übertragen. In zukünftigen Arbeiten kann die Wissensbasis also mit weiteren Quellen erweitert und auf Korrektheit überprüft werden.

Named-Entity-Recognition

Für die *Named-Entity-Recognition* wird das *CoreNLP*-Tool von Stanford verwendet. Das *CoreNLP*-Tool von Stanford wurde nicht mit anderen Methoden der NER verglichen. Aus diesem Grund ist es möglich, dass NER-Systeme mit einer höheren

Präzision und Ausbeute existieren. In zukünftigen Arbeiten kann das NER-System mit anderen verglichen und ggf. ausgetauscht werden.

Candidate-Entity-Generation

In der *Candidate-Entity-Generation* wird die Oberflächenform der Entitätserwähnung verwendet. Zusätzlich könnten weitere Merkmale verwendet werden, um die Kandidatenentitäten-Sets zu generieren. Eine Auswahl weiterer Merkmale wurde nicht getestet oder evaluiert und kann in zukünftigen Arbeiten ergänzt werden.

Candidate-Entity-Ranking

Im *Candidate-Entity-Ranking* orientiert sich die Auswahl der Merkmale an anderen Entity-Linking-Systemen. In zukünftigen Arbeiten kann überprüft werden, ob eine andere Auswahl an Merkmalen, eine signifikante Verbesserung im *Candidate-Entity-Ranking* bewirkt.

Unlinkable-Mention-Prediction

Die *Unlinkable-Mention-Prediction* erstellt die Prioritätsliste regelbasiert. Regelbasierte Ansätze können durch die Hinzunahme oder Entfernung von Regeln modifiziert werden. Die Auswahl der Regeln ist in der Theorie argumentiert, aber nicht für verschiedene Regeln in der Praxis evaluiert. In zukünftigen Arbeiten kann der heuristische Ansatz der *Unlinkable-Mention-Prediction* verbessert oder ausgetauscht werden.

Evaluation

Die Gold-Standards wurden händisch erstellt. Von Menschen erstellte Musterlösungen können Fehler enthalten und sind durch subjektive Entscheidungen geprägt. Auch mit mehreren Annotatoren und einer Konsensentscheidung bei Uneinigkeit, sind keine fehlerfreien Musterlösungen zu erwarten. Die Evaluation kann durch Hinzunahme von weiteren Architekturdokumentationen erweitert und verbessert werden. Außerdem sollten Architekturdokumentationen mit größerem Umfang ergänzt werden, die nicht aus der Forschung stammen.

Wie die Evaluation gezeigt hat, erzielt unser Ansatz für domänenspezifisches Entity Linking bereits gute Ergebnisse. Trotzdem hat das von uns entwickelte Entity-Linking-System noch viel Erweiterungs- und Verbesserungspotenzial.

Literatur

[1] Sören Auer u. a. „Dbpedia: A nucleus for a web of open data“. In: *The semantic web*. Springer, 2007, S. 722–735.

[2] Amit Bagga und Breck Baldwin. „Entity-based cross-document coreferencing using the vector space model“. In: *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*. 1998.

[3] Michele Bevilacqua und Roberto Navigli. „Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information“. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, S. 2854–2864.

[4] Zheng Chen u. a. „CUNY-BLENDER TAC-KBP2010“. In: (2010).

[5] Tao Cheng, Xifeng Yan und Kevin Chen-Chuan Chang. „EntityRank: searching entities directly and holistically“. In: *Proceedings of the 33rd international conference on Very large data bases*. 2007, S. 387–398.

[6] Teresa Mihwa Chung und Paul Nation. „Technical vocabulary in specialised texts“. In: (2003).

[7] Silviu Cucerzan. „Large-scale named entity disambiguation based on Wikipedia data“. In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 2007, S. 708–716.

[8] Abhishek Gattani u. a. „Entity extraction, linking, classification, and tagging for social media: a wikipedia-based approach“. In: *Proceedings of the VLDB Endowment* 6.11 (2013), S. 1126–1137.

[9] Stephen Guo, Ming-Wei Chang und Emre Kiciman. „To link or not to link? a study on end-to-end tweet entity linking“. In: *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2013, S. 1020–1030.

[10] Marios Hadjieleftheriou und Divesh Srivastava. *Approximate string processing*. Now Publishers Inc, 2011.

[11] Xianpei Han, Le Sun und Jun Zhao. „Collective entity linking in web text: a graph-based method“. In: *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 2011, S. 765–774.

[12] Xianpei Han und Jun Zhao. „Named entity disambiguation by leveraging wikipedia semantic knowledge“. In: *Proceedings of the 18th ACM conference on Information and knowledge management*. 2009, S. 215–224.

[13] Xianpei Han und Jun Zhao. „NLPR_KBP in TAC 2009 KBP Track: A Two-Stage Method to Entity Linking." In: *TAC*. Citeseer. 2009.

[14] „ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary". In: *ISO/IEC/IEEE 24765:2017(E)* (2017), S. 1–541. DOI: 10.1109/IEEESTD.2017.8016712.

[15] 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)*. IEEE. 2019, S. 250–253.

[16] Jan Keim u. a. „Trace Link Recovery for Software Architecture Documentation". In: *European Conference on Software Architecture*. Springer. 2021, S. 101–116.

[17] Sayali Kulkarni u. a. „Collective annotation of wikipedia entities in web text". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, S. 457–466.

[18] Leah S Larkey u. a. „Acrophile: an automated acronym extractor and server". In: *Proceedings of the fifth ACM conference on Digital libraries*. 2000, S. 205–214.

[19] Thomas Lin, Oren Etzioni u. a. „Entity linking at web scale". In: *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*. 2012, S. 84–88.

[20] Christopher D Manning u. a. „The Stanford CoreNLP natural language processing toolkit". In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014, S. 55–60.

[21] Behrang Mohit. „Named entity recognition". In: *Natural language processing of semitic languages*. Springer, 2014, S. 221–245.

[22] William Nagy und Dianna Townsend. „Words as tools: Learning academic vocabulary as language acquisition". In: *Reading research quarterly* 47.1 (2012), S. 91–108.

[23] Roberto Navigli. „Word sense disambiguation: A survey". In: *ACM computing surveys (CSUR)* 41.2 (2009), S. 1–69.

[24] Thomas Pellissier Tanon, Gerhard Weikum und Fabian Suchanek. „Yago 4: A reasonable knowledge base". In: *European Semantic Web Conference*. Springer. 2020, S. 583–596.

[25] Ralf H Reussner u. a. *Modeling and simulating software architectures: The Palladio approach*. MIT Press, 2016.

[26] Per Runeson und Martin Höst. „Guidelines for conducting and reporting case study research in software engineering". In: *Empirical software engineering* 14.2 (2009), S. 131–164.

[27] Wei Shen, Jianyong Wang und Jiawei Han. „Entity linking with a knowledge base: Issues, techniques, and solutions". In: *IEEE Transactions on Knowledge and Data Engineering* 27.2 (2014), S. 443–460.

[28] Wei Shen u. a. „Linden: linking named entities with knowledge base via semantic knowledge“. In: *Proceedings of the 21st international conference on World Wide Web*. 2012, S. 449–458.

[29] Avirup Sil und Alexander Yates. „Re-ranking for joint named-entity recognition and linking“. In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, S. 2369–2374.

[30] Nikolaos Stylianou und Ioannis Vlahavas. „A neural entity coreference resolution review“. In: *Expert Systems with Applications* 168 (2021), S. 114466.

[31] Kazem Taghva und Jeff Gilbreth. „Recognizing acronyms and their definitions“. In: *International Journal on Document Analysis and Recognition* 1.4 (1999), S. 191–198.

[32] Klaar Vanopstal, Bart Desmet und Véronique Hoste. „Towards a Learning Approach for Abbreviation Detection and Resolution.“ In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. 2010.

[33] Vasudeva Varma u. a. „IIIT Hyderabad at TAC 2009.“ In: *TAC*. 2009.

[34] Aleksa Vukotic u. a. *Neo4j in action*. Bd. 22. Manning Shelter Island, 2015.

[35] Chris Welty u. a. „A comparison of hard filters and soft evidence for answer typing in watson“. In: *International Semantic Web Conference*. Springer. 2012, S. 243–256.

[36] Stuart Yeates, David Bainbridge und Ian H Witten. „Using compression to identify acronyms in text“. In: *arXiv preprint cs/0007003* (2000).

[37] Jiangtao Zhang u. a. „Domain-specific entity linking via fake named entity detection“. In: *International Conference on Database Systems for Advanced Applications*. Springer. 2016, S. 101–116.

[38] Wei Zhang u. a. „NUS-I2R: Learning a Combined System for Entity Linking.“ In: *TAC*. 2010.