TECHNISCHE
UNIVERSITÄT
DRESDEN

Fakultät Informatik, Institut für Systemarchitektur, Lehrstuhl Rechnernetze

DIPLOMARBEIT

zur Erlangung des akademischen Grades
DIPLOM-MEDIENINFORMATIKER (DIPL.-MEDIEN-INF.)

Informationsextraktion auf Basis strukturierter Daten

Marcus Schramm
geboren am 02.10.1980 in Rüdersdorf b. Berlin

Betreuer TUD: Dr.-Ing. Thomas Springer
Betreuer SAP AG: Dipl.-Medien-Inf. Falk Brauer
Verantwortlicher Hochschullehrer: Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Dresden im Mai 2008

Vertraulichkeitserklärung

Die vorliegende Arbeit enthält vertrauliche Informationen der SAP AG und ihren Partnern. Die Arbeit und ihre Inhalte dürfen weder vollständig noch auszugsweise veröffentlicht oder an Dritte weitergegeben werden. Davon ausgenommen ist die Weitergabe an Mitarbeiter der Technischen Universität Dresden, die der Geheimhaltungspflicht entsprechend dem Rahmenvertrag mit der SAP AG unterliegen.

Aufgabenstellung

Um eine effiziente Verwaltung und Suche von Dokumenten zu ermöglichen, werden heute neuartige Ansätze untersucht, die verstärkt die Semantik von Dokumentinhalten berücksichtigen. Ein Ansatz in diesem Gebiet ist die Informationsextraktion (IE). Durch die maschinelle Verarbeitung von unstrukturierten Daten kann hierbei Wissen über den Inhalt von Dokumenten extrahiert werden. Aktuelle Lösungen in diesem Forschungsbereich sind meist monolithisch und schwer adaptierbar. Im Rahmen des SAP Forschungsprogramms „Data Management & Analytics“ wird ein Framework zur deklarativen Beschreibung von IE-Prozessen entwickelt. Durch Parametrisierung von generischen Operatoren zur Informationsextraktion soll eine geeignete Abstraktion hinsichtlich der eingesetzten Verfahren geschaffen werden. Generische Operatoren lassen sich nach Extraktion von Merkmalen (Feature Extraction), Erkennen von Entitäten (Named Entity Recognition) und Erkennen von Beziehungen (Relation Finding) klassifizieren.
Für das Erkennen von Entitäten wird Domainwissen benötigt, welches in Enterprise-Anwendungsfällen häufig in Form strukturierter Daten (z.B. in Datenbanken) vorliegt. Besonders innerhalb abgeschlossener Domänen (z.B. Datenmanagement im PLM oder CRM Bereich), in denen große Mengen strukturierter Daten vorhanden sind, erscheint dieser Ansatz vielversprechend. Die Problemstellung besteht in der effizienten und präzisen Ermittlung von Korrespondenzen zwischen strukturierten und unstrukturierten Daten.
Ziel der Arbeit ist es, Methoden und Algorithmen zu analysieren welche die Identifizierung der o.g. Korrespondenzen ermöglichen. Diese sollen in Form eines Operators in das in der Entwicklung befindliche Framework integriert werden. Die Funktionsweise dieses Operators wird in Gegenüberstellung zu Standardsuchverfahren validiert.

Schwerpunkte

- Recherche und detaillierte Analyse des State-of-the-Art im Bereich Named Entity Recognition
- Anforderungsanalyse hinsichtlich eines konfigurierbaren Operators zum Erkennen von Entitäten auf Basis strukturierter Daten
- Analyse von Methoden und Algorithmen zur Implementierung dieses Operators
- Konzeption des Operators und dessen Konfigurationsschnittstellen
- Validierung der Konzeption anhand von Standardsuchverfahren
- Bewertung der entwickelten Lösung

Kurzfassung

Softwareentwickler, Berater und Projektmanager nutzen häufig kostengünstige Unterstützungsplattformen wie das SAP Developer Network. Untersuchungen haben jedoch gezeigt, dass es durchschnittlich zwei oder mehr Tage dauert bis die Frage eines Nutzers beantwortet wird. Eine Analyse der Daten hat hervorgebracht, dass für den Großteil der einfachen und häufig wiederkehrenden Anfragen bereits relevante Inhalte durch die Forumgemeinschaft veröffentlicht wurden. Daher ist es unter Anwendung geeigneter Technologien möglich, dem Kunden unmittelbar zu antworten. Um neue Beiträge mit vorhandenen Inhalten abzugleichen werden im OKKAM Projekt Techniken zur Analyse dieser Inhalte untersucht. In der vorliegenden Arbeit wurde ein Ansatz entwickelt, der es erlaubt, Produkte in Forumbeiträgen zu erkennen. Insbesondere ist es damit möglich Produktinstanzen auf Basis existierender strukturierter Daten über SAP Produkte und SAP Terminologie zu identifizieren. Es können mehr als 75% der Produktinstanzen mit einer Genauigkeit von über 82% ermittel werden. Die Ergebnisse dieser Arbeit sind sehr vielversprechend und fließen in die Entwicklung eines umfassenden Frameworks für die zukünftige Generation von Kundendienstanwendungen ein.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Im Bereich von Unternehmenssoftware entwickeln sich Business Intelligence (BI) Lösungen immer mehr zu Kernkomponenten. Sie haben Einfluss auf wichtige Entscheidungsprozesse in Unternehmen, sodass sowohl die funktionellen als auch die technischen Anforderungen an BI ständig erweitert werden. Klassische BI-Verfahren wie Online Analytical Processing (OLAP) und Data Warehouse (DWH) basieren typischerweise auf strukturierten Daten, zumeist Datenbanken. Diese Form von Daten unterstützt deklarative Anfragemethoden und Programmierschnittstellen für den Datenzugriff. Damit lässt sich eine Vielzahl von Geschäftsprozessen sowie die dazugehörigen Kennzahlen eines Unternehmens auswerten.

Eine vollständige Analyse schließt mehr und mehr auch unstrukturierte Daten ein, welche wertvolle zusätzliche Informationen enthalten können. Web 2.0 Technologien wie Foren, Wikis, Blogs, RSS-Feeds, aber auch klassische elektronische Kommunikationsformen wie Emails und Office-Dokumente können dazu als Datenquelle genutzt werden. Diese Technologien werden nicht ausschließlich zur Kommunikation verwendet sondern entwickeln sich zu relevanten Informationsplattformen. Schätzungen gehen davon aus, dass der Anteil der unstrukturierten Informationen in drei bis fünf Jahren bis zu 85% aller geschäftsrelevanten Informationen ausmachen wird [Blumberg und Atre, 2003]. Daher ist die Entwicklung neuer Technologien notwendig, um gezielt Informationen aus den entsprechenden Datenquellen abrufen zu können. Die Informationsextraktion bietet dabei die Möglichkeit der automatisierten Aufbereitung von unstrukturierten Daten mit dem Ziel, diese in strukturierter Form darzustellen.

Neben der Analyse spielt in Verbindung mit großen Datenmengen auch die Suche nach Informationen eine entscheidende Rolle. Abbildung 1.1 stellt die Verteilung der täglich im Internet produzierten Daten dar. Nach Ramakrishnan und Tomkins werden nahezu zwei Drittel des gesamten Inhalts von Nutzern erzeugt. Sie sind somit unstrukturiert und unterliegen keiner redaktionellen Überprüfung [Ramakrishnan und Tomkins, 2007]. Die Möglichkeit der Suche in solchen Inhalten beschränkt sich meist auf eine Stichwortsuche im Volltext. Für SAP sind die im SAP-Marketplace gebündelten Online-Portale wichtige Kanäle für den unmittelbaren Informationsaustausch mit Kunden und Partnern. Nutzer finden dort u.a. Trainingsunterlagen, Codebeispiele, Geschäftsszenarien, SAP-spezifische Terminologien, technische Hinweise und Dokumentationen zu SAP-Lösungen. Supportmitarbeiter der SAP greifen auf diese Inhalte in den SAP Portalen zu, um Kundenanfragen zu Lösungen oder technischen Problemen beantworten zu können. Die von den Benutzern erwarteten Ergebnisse einer zielgerichteten Suche sprengen den Rahmen des Dokumentenparadigmas. Aggregierte Informationen, wie zum Beispiel eine *exakte* Aussage oder eine Aufzählung exakter Antworten ist für einen Kunden nützlicher als eine Menge von Dokumenten, in denen die Information enthalten ist. Die Informationsextraktion schafft die Grundlage dafür, unstrukturierten

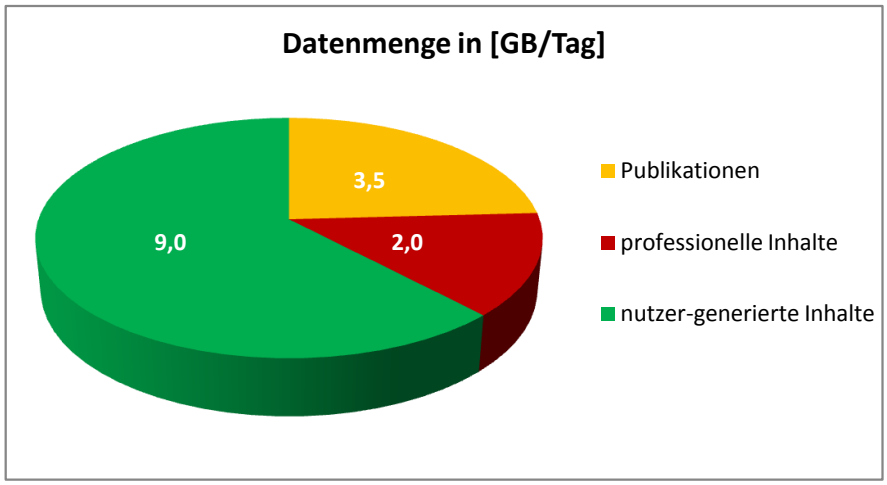


Abbildung 1.1: Verteilung täglich produzierter Inhalte im Internet

Daten eine Semantik zuzuordnen mit deren Unterstützung die Suche verbessert wird.

1.1 Vorhaben

Die Vision des OKKAM Projekts[1] ist die Bereitstellung einer Infrastruktur für das Web of Entities. Das Ziel ist, Entitäten über die Grenzen von Domänen hinweg auffindbar zu machen und sie mit beliebigen anderen Entitäten zu verknüpfen. Zukünftig erstellte Inhalte werden mit entsprechenden Metadaten versehen, die eine Verbindung zu einer Entitäten kennzeichnen. Mit diesen Daten lassen sich eine ganze Reihe von neuartigen Anwendungsfällen realisieren.
Im BI-Bereich sind Anfragen der Art, „Welche Probleme haben unsere Kunden bei der Einführung eines neuen Produktes?“ oder „Welche neuen Funkionalitäten werden am häufigsten nachgefragt?“, möglich. In der Kategorie Question Answering (Intelligente Antwortsysteme) lassen sich Kundenprobleme der Art, „Wie löse ich ein Treiberproblem mit Oracle 9i und SAP NetWeaver 2004?“ beantworten. Außerdem kann die Suche nach Experten adressiert werden. Sie ermöglicht es Kunden, die eine Frage wie, „Wer kennt sich mit der Installation von SAP Process Integration aus?“, haben, direkt an den entsprechenden Ansprechpartner weiterzuleiten.
Die Grundlage für derartige Anwendungen ist die Kennzeichnung von Entitäten in bestehenden unstrukturierten Daten sowie die Speicherung und Auswertung der damit verbundenen Zusammenhänge. Dazu werden Techniken benötigt, welche diese Entitäten identifizieren. Ein vielversprechender Ansatz ist die Informationsextraktion auf Basis strukturierter Daten. Sie macht sich zum Vorteil, dass strukturierte Daten aus der jeweiligen Domäne in großer Menge zur Verfügung stehen. In der vorliegenden Arbeit handelt es sich um Unternehmensdaten, wie Produkte und Terme aus dem SAP Umfeld. Die genaue Betrachtung der vorhandenen Daten erfolgt im Kapitel 2.2. Diese vorhandenen strukturierten Daten werden bei der Erkennung von Entitäten in unstrukturierten Daten eingesetzt. Als Ergebnis einer Extraktion liegen demnach unstrukturierte Daten vor, die mit strukturierten Daten verknüpft sind. Gemeinsam bilden sie die Grundlage für die Umsetzung der zuvor genannten Anwendungsfälle.

[1] http://www.okkam.org

1.2 Zielstellung

Ziel dieser Arbeit ist es, die genannten Grundlagen zu schaffen indem Datenbankentitäten mit Hilfe geeigneter Methoden und Algorithmen identifiziert werden. Die entwickelten Konzepte sollen auf das SAP Research UIM-Framework übertragen werden, womit dieses geeignet erweitert wird.

1.3 Methodik und Aufbau

Eine wichtige Voraussetzung für die Arbeit ist die genaue Analyse der vorhandenen strukturierten und unstrukturierten Daten. Ausgehend von dieser Analyse können die Anforderungen definiert werden, welche durch die Konzeption und Implementierung abgedeckt werden müssen. Desweiteren müssen vorhandene Technologien sowie verwandte Arbeiten im Bereich der Informationsextraktion analysiert werden, um einerseits auf vorhandene Methoden zurückgreifen und andererseits eine Abgrenzung vornehmen zu können.

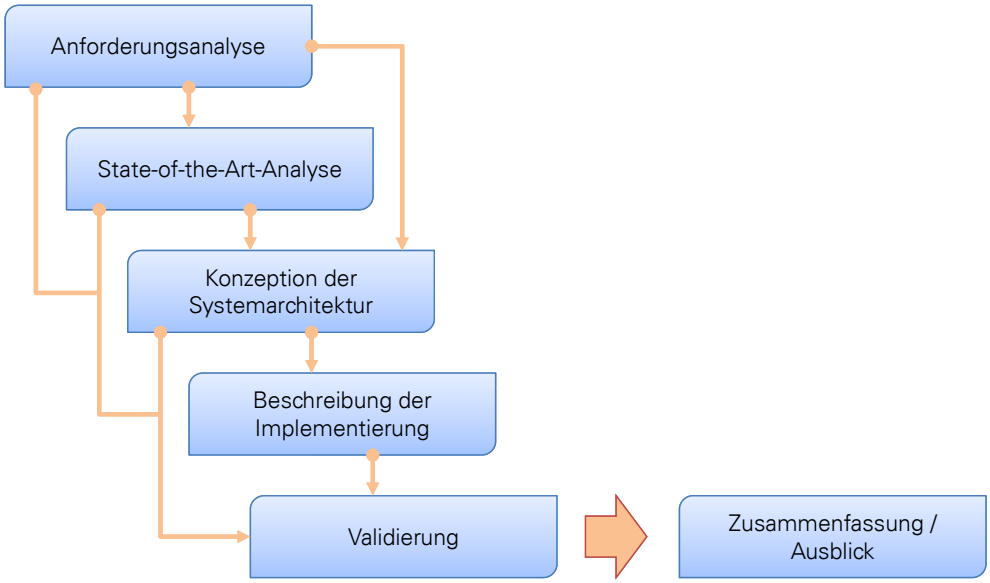


Abbildung 1.2: Vorgehensmodell

Der weitere Verlauf der Arbeit ist in Abbildung 1.2 dargestellt. Im Anschluss an die Einleitung wird in Kapitel 2 die Anforderungsanalyse durchgeführt. Sie beginnt mit der Darstellung von Anwendungsfällen und den technischen Voraussetzungen für die Arbeit. Darauf aufbauend werden die Anforderungen für die Entitätserkennung definiert sowie die sich daraus ergebenden Herausforderungen identifiziert. Daran schließt sich in Kapitel 3 die Analyse zum Stand der Technik mit der Betrachtung verwandter Arbeiten an. Ausgehend von der Anforderungsanalyse und den verwandten Arbeiten wird in Kapitel 4 die Konzeption der Systemarchitektur beschrieben. Hier werden zunächst die Schlüsselkomponenten für die Informationsextraktion auf Basis strukturierter Daten bestimmt. Sie fügen sich modular zu einem System zusammen, das die Erkennung von Entitäten realisiert. Ein wichtiger Aspekt dieser Arbeit ist prototypische Implementierung des konzipierten Systems. In Kapitel 5 werden die Details dieser Implementierung dargestellt. Die Evaluation in Kapitel 6 zeigt, inwiefern die Konzepte aus Kapitel 4 umgesetzt durch die Implementierung, die Anforderungen aus Kapitel 2 erfüllen. Abschließend in Kapitel 7 werden noch einmal alle Aspekte

dieser Arbeit zusammengefasst. Ein Ausblick zeigt die zukünftigen Anwendungen, die mit den Ergebnissen der Arbeit möglich sind. Darüber hinaus wird betrachtet, welche weiteren Entwicklungen auf dem Gebiet gemacht werden können.

Kapitel 2

Anforderungsanalyse

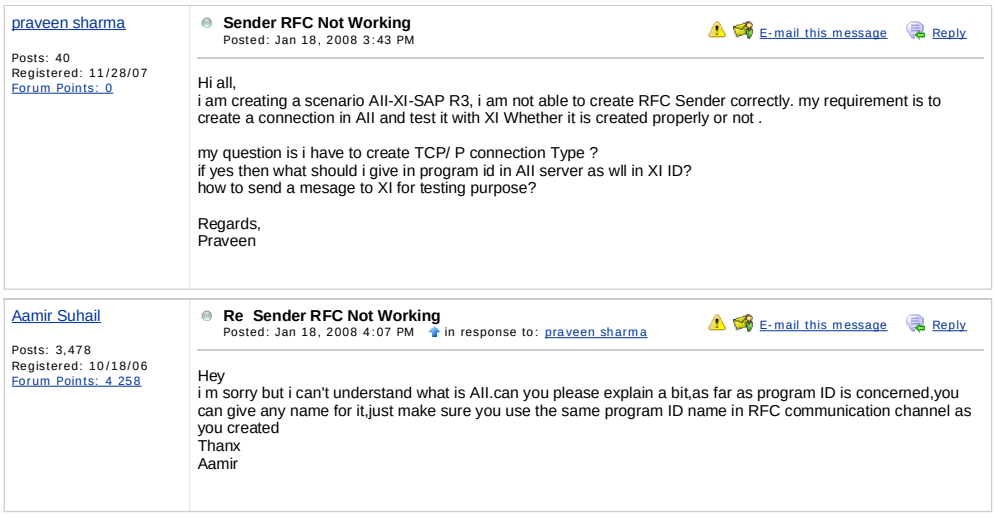
In diesem Kapitel wird die Anforderungsanalyse für die zu entwickelnde Lösung vorgenommen. Dazu werden zuerst die Anwendungsfälle des OKKAM-Projekts betrachtet. Daran wird verdeutlicht welche Aspekte des EU-Projekts durch die vorliegende Arbeit abgedeckt werden. Eine wichtige Voraussetzung für die Formulierung von Anforderungen ist die Betrachtung der vorhandenen Daten welche in die Arbeit einbezogen werden sollen. Ihre Inhalte und ihr Aufbau werden im Folgenden dargestellt. Im weiteren Vorgehen wird das bereits existierende SAP Research Unstructured Information Management (UIM)-Framework vorgestellt. Es bildet die Grundlage der Arbeit und wird als Ergebnis dieser geeignet erweitert. Nachdem alle Vorbedingungen erläutert wurden, wird zur Definition der qualitativen und funktionalen Anforderungen übergegangen. Anschließend werden Metriken bestimmt, anhand derer die Qualität der Ergebnisse im weiteren Verlaufe der Arbeit gemessen werden. Als Abschluss dieses Kapitels werden die Herausforderungen, die sich aufgrund dieser Anforderungsanalyse ergeben, zusammengefasst.

2.1 Anwendungsfälle

Im OKKAM-Projekt werden durch SAP zwei Anwendungsfälle adressiert. Zum einen sollen Antwortzeiten in Hilfeforen reduziert werden. Dieses Problem stammt aus der Kategorie Question Answering. Der zweite Anwendungsfall führt in den Bereich Business Intelligence. Um wichtige strategische Entscheidungen in Unternehmen treffen zu können, sollen unter anderem auch Informationen aus unstrukturierten Daten genutzt werden, ohne diese manuell sichten zu müssen.

2.1.1 Reduzierung von Antwortzeiten in Hilfeforen

Praveen Sharma ist SAP-Entwickler mit Einsteigerkenntnissen. Er möchte ein bestimmtes Szenario mit Hilfe von SAP-Komponenten konfigurieren. Da er dabei auf ein Problem stößt versucht er in der SAP Developer Network (SDN)-Gemeinschaft Unterstützung zu erhalten. Dazu stellt er den in Abbildung 2.1 gezeigten Beitrag in das SDN-Forum. Bereits 25 Minuten später antwortet Aamir Suhail, wie ebenfalls in der Abbildung zu sehen, mit der Bitte, das Problem genauer zu beschreiben.
Vier Tage später antworten dann Vikas und Maheshwari indem sie Praveen jeweils auf eine Seite verweisen, auf denen bereits Problemlösungen beschrieben wurden. Es handelt sich um Experten-Blogs, die ebenfalls aus dem SAP Developer Network stammen. Damit ist Praveens Problem gelöst, wie er auch kurze Zeit später bestätigt (Abbildung 2.2).



praveen sharma

Posts: 40
Registered: 11/28/07
Forum Points: 0

Sender RFC Not Working
Posted: Jan 18, 2008 3:43 PM

E-mail this message Reply

Hi all,
i am creating a scenario AII-XI-SAP R3, i am not able to create RFC Sender correctly. my requirement is to create a connection in AII and test it with XI Whether it is created properly or not .

my question is i have to create TCP/ P connection Type ?
if yes then what should i give in program id in AII server as wll in XI ID?
how to send a mesage to XI for testing purpose?

Regards,
Praveen

Aamir Suhail

Posts: 3,478
Registered: 10/18/06
Forum Points: 4 258

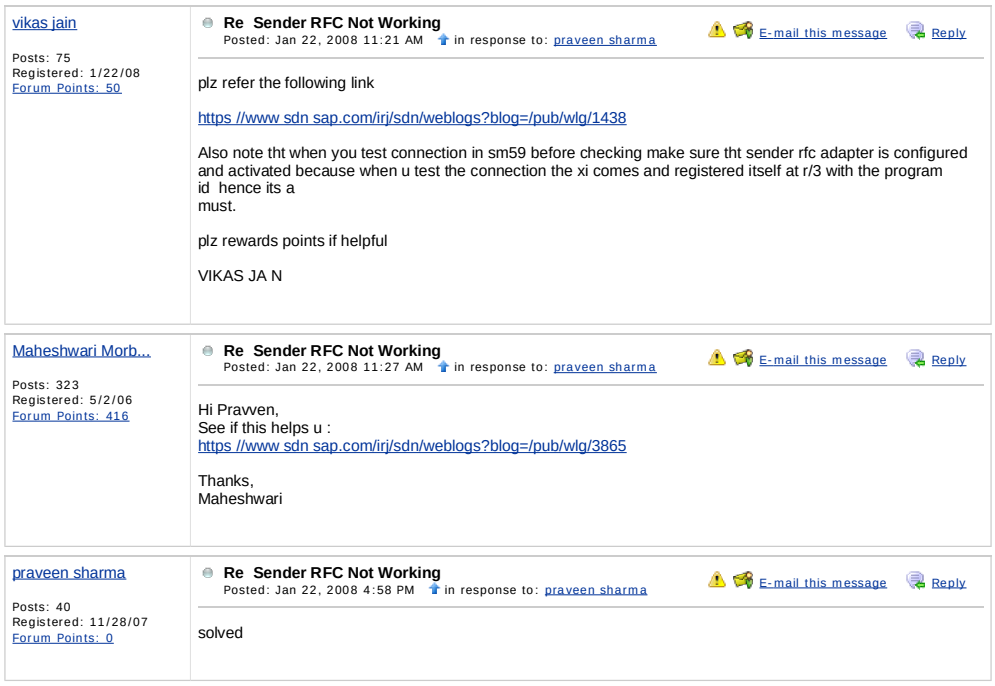
Re Sender RFC Not Working
Posted: Jan 18, 2008 4:07 PM in response to: praveen sharma

E-mail this message Reply

Hey
i m sorry but i can't understand what is AII.can you please explain a bit,as far as program ID is concerned,you can give any name for it,just make sure you use the same program ID name in RFC communication channel as you created
Thanx
Aamir

Abbildung 2.1: Praveen Sharmas Problem

Die Lösung von Praveen Sharmas Problem stellt ein typisches Szenario für Hilfe- und Unterstützungsforen dar. Die Antwort auf die Frage existiert bereits in Form vorhandener Beiträge oder sonstiger Inhalte aus der Gemeinschaft. Die Problemlösung besteht häufig *nur* in der Verknüpfung von Frage und Antwort. Dies kann dementsprechend durch einen Nutzer erfolgen, der womöglich überhaupt nicht mit dem Problem vertraut ist, wohl aber durch Beobachtung auf die Antwort zu der Frage gestoßen ist. Laut Ramakrishnan können in derartigen Foren ca. 80% der Anfragen mit bereits vorhandenen Inhalten beantwortet werden [Ramakrishnan, 2008]. Lediglich für die verbleibenden 20% müssen neue Inhalte erzeugt werden.



vikas jain

Posts: 75
Registered: 1/22/08
Forum Points: 50

Re Sender RFC Not Working
Posted: Jan 22, 2008 11:21 AM in response to: praveen sharma

E-mail this message Reply

plz refer the following link

https //www sdn sap.com/irj/sdn/weblogs?blog=/pub/wlg/1438

Also note tht when you test connection in sm59 before checking make sure tht sender rfc adapter is configured and activated because when u test the connection the xi comes and registered itself at r/3 with the program id hence its a
must.

plz rewards points if helpful

VIKAS JA N

Maheshwari Morb...

Posts: 323
Registered: 5/2/06
Forum Points: 416

Re Sender RFC Not Working
Posted: Jan 22, 2008 11:27 AM in response to: praveen sharma

E-mail this message Reply

Hi Pravven,
See if this helps u :
https //www sdn sap.com/irj/sdn/weblogs?blog=/pub/wlg/3865

Thanks,
Maheshwari

praveen sharma

Posts: 40
Registered: 11/28/07
Forum Points: 0

Re Sender RFC Not Working
Posted: Jan 22, 2008 4:58 PM in response to: praveen sharma

E-mail this message Reply

solved

Abbildung 2.2: Lösung für Praveen Sharmas Problem

Die Intention von SAP ist es, sich diesen Fakt im OKKAM-Projekt zum Nutzen zu machen. Wenn die Antwort auf eine Frage bereits vorhanden ist, sollte diese identifiziert und dem Nutzer direkt angezeigt werden. Hinter der Idee von OKKAM steht, Verbindungen von Frage und Antwort anhand der vorkommenden Entitäten zu erkennen.
Möchte ein Nutzer einen neuen Eintrag hinzufügen, wird der Inhalt dieses Eintrags nach Entitäten durchsucht. Bevor der Nutzer endgültig seinen Beitrag zum Forum hinzufügt, bekommt er eine Liste mit vorhandenen Inhalten angezeigt, in denen gleiche Entitäten erkannt wurden. Er hat die Möglichkeit zunächst dort eine Lösung für sein Problem auszuwählen. Erst wenn dies nicht den gewünschten Erfolg bringt sendet er seinen Eintrag endgültig ab. Wie bereits in der Zielstellung der vorliegenden Arbeit (Abschnitt 1.2) genannt, werden hier Methoden untersucht, die die Identifikation von Entitäten ermöglichen. Dies bildet die Grundlage für die Umsetzung des hier vorgestellten Anwendungsfalls.

2.1.2 Business Intelligence anhand unstrukturierter Daten

Hans Peter ist Vertriebsleiter eines großen Softwareunternehmens. Er muss sich mit wichtigen strategischen Fragestellungen auseinander setzen. Beispiele für diese Fragen sind in Abbildung 2.3 illustriert.

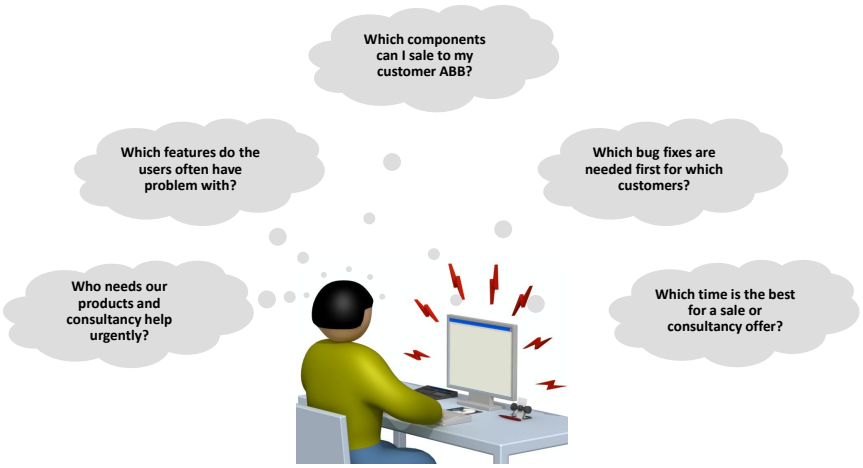


Abbildung 2.3: Hans Peters Problem

Für die Beantwortung seiner Fragen müsste er überwiegend auf aggregierte Daten zurückgreifen, die mit Hilfe von BI-Verfahren erstellt werden. Viele der gewünschten Informationen lassen sich jedoch nicht aus den Datenbanken, also den strukturierten Daten, des Unternehmens entnehmen. Als Datenquelle kommt zusätzlich eine Online-Gemeinschaft wie das SAP Developer Network in Frage. Kunden äußern hier ihre Fragen, Wünsche und Probleme. Viele wichtige Informationen sind implizit in diesen unstrukturierten Daten vorhanden. Für Hans Peter stellt sich das Problem wie er diese Informationen möglichst effizient abfragen oder gar in aggregierter Form darstellen kann.
Die Informationsextraktion wird an dieser Stelle genutzt, um Entitäten drei verschiedener Dimensionen (Kunde, Produkt, Zeit) zu erfassen. Auf diese Art und Weise werden die impliziten Informationen als explizite Daten extrahiert. Mit Hilfe der Dimensionen Kunde und Produkt ist es zum Beispiel möglich, Aussagen darüber zu treffen, wie oft ein Kunde Fragen oder Probleme hat oder zu welchen Produkt die meisten Anfragen gestellt wurden. Verknüpft man beide Dimensionen erhält man Auskunft darüber, für welche Produkte sich ein Kunde aktuell besonders stark interessiert. Die Dimension Zeit erfüllt mehrere Zwecke.

Mit ihrer Hilfe kann man zum Beispiel die Qualität der Gemeinschaft beurteilen. Dazu wird untersucht wie schnell eine Frage durchschnittlich beantwortet wird. Besonders wichtig ist die Zeit, um kontextuelle Zusammenhänge mit geschäftsbezogenen Ereignissen zu bilden. Wird zum Beispiel ein neues Produkt durch einen Wettbewerber angekündigt und folgen darauf besonders viele Fragen, die thematisch damit zusammenhängen, ist dies ein Indiz dafür, dass Handlungsbedarf besteht, da ansonsten die Gefahr droht, dass Kunden zur Konkurrenz wechseln.
Das Ergebnis dieser Arbeit wird es sein, in erster Linie Entitäten der Dimension Produkt zu identifizieren. Die Informationen werden in geeigneten Datenstrukturen abgelegt, damit diese für unterschiedliche Anfrageverarbeitungen zur Verfügung stehen und mit den Ergebnissen weiterer Extraktionen (für Kunde und Zeit) kombiniert werden können.

2.2 Datenquellen

In diesem Abschnitt wird dargestellt welche Daten für diese Arbeit zur Verfügung stehen. Als unstrukturierte Daten kommen Inhalte aus dem SAP Developer Network zum Einsatz. Für die Identifikation von Entitäten in diesen Inhalten wird die SAP-Terminologie-Datenbank (SAP Term) genutzt. Beide Datenquellen werden im Folgenden vorgestellt.

2.2.1 Das SAP Developer Network

Wie bereits in der Einleitung zu diesem Abschnitt genannt, ist das SAP Developer Network[1] die Datenquelle für unstrukturierte Inhalte, in denen im Rahmen dieser Arbeit Entitäten zu erkennen sind. Das SDN ist eine Online Gemeinschaft für den Austausch zu SAP bezogenen Technologien. Es beinhaltet technische Dokumentationen, Expertenblogs, Wiki-Inhalte, Codebeispiele, umfangreiche eLearning-Angebote sowie aktive, moderierte Diskussionsforen. Die Zielgruppen sind Kunden, SAP-Entwickler, Berater sowie Systemintegratoren und -administratoren.
Sowohl bei den Wiki-Inhalten und Forumeinträgen als auch bei den Expertenblogs handelt es sich um nutzergenerierte Inhalte die kaum einer redaktionellen oder ähnlich gearteten Nachbearbeitung unterliegen. Hinzu kommt der immense Umfang der Daten. Allein in den Diskussionsforen werden täglich zwischen 3000 und 5000 Beiträge eingestellt. Die Informationssuche und -aggregation gestaltet sich daher besonders schwierig. Um dies dennoch zu ermöglichen werden die Inhalte des SDN für die Verwendung in dieser Arbeit herangezogen.

2.2.2 Die SAP-Terminologie-Datenbank

Innerhalb der SAP existiert eine Datenbank (SAP Term), die nahezu die gesamte Terminologie abdeckt, die mit dem SAP-Umfeld insbesondere den SAP-Produkten in Verbindung steht. Die Terminologie steht in insgesamt 33 Sprachen zur Verfügung und sorgt in erster Linie für ein gemeinsames Verständnis von SAP-Produkten und -Technologien. Speziell für Übersetzungen dient diese Datenbank der konsistenten Verwendung von Begriffen. Für jedes SAP-Produkt müssen Begriffe und Glossardefinitionen in deutscher und englischer Sprache definiert werden. Darüber hinaus können Synonyme, Abkürzungen, veraltete und ungültige Schreibweisen sowie Kommentare und sonstige Informationen zur Verfügung gestellt werden.

[1] http://www.sdn.sap.com

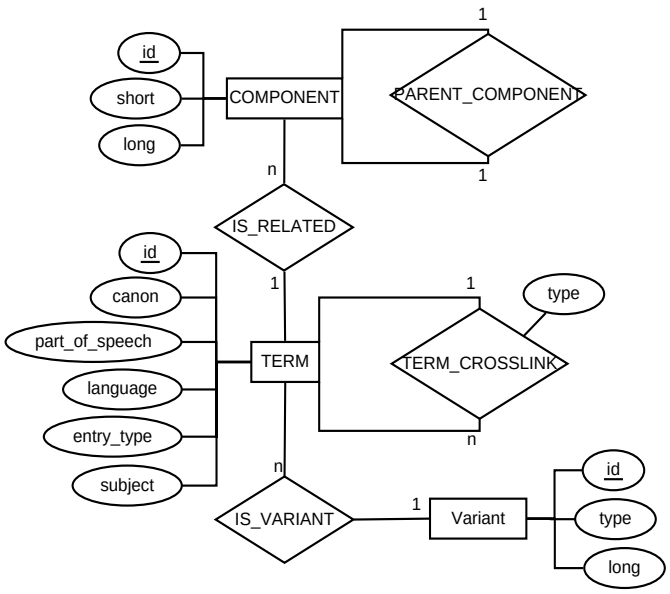


Abbildung 2.4: ER-Modell des SAP Glossars

Für die vorliegende Arbeit steht ein Auszug dieser Terminologie in Form eines Open Lexicon Interchange Format (OLIF)-Exports zur Verfügung. Er beinhaltet ca. 100'000 Begriffe in englischer Sprache, die Bezeichnung sämtlicher 900 SAP-Produktkomponenten und die 2'500 verschiedenen SAP-Produktnamen. Abbildung 2.4 zeigt das Schema der daraus generierten Datenbank.

Ausgangspunkt ist die Entität `Component`. Hier werden SAP-Komponenten mit Kurz- und Langschreibweise gespeichert. Zwischen Komponenten kann eine Eltern-Kind-Beziehung existieren, d.h. eine Komponente kann einer anderen Kompontente untergeordnet sein. Dies wird über die Relation `Parent_Component` ausgedrückt. Zu jeder Komponente werden darüber hinaus ein oder mehrere Terme definiert. Sie werden in der Entität `Term` gespeichert. Ein Term ist durch seine kanonische Form charakterisiert. Außerdem muss die Sprache und die Wortart (`part_of_speech`) angegeben werden. Terme können in Beziehung zueinander stehen. Über die Relation `Term_Crosslink` lassen sich Beziehungen des Typs *orthographische Abwandlung*, *Synonym* und *übergeordneter Term* abbilden. Der letztgenannte ist unidirektional während die ersten beiden Typen bidirektional gelten. Neben den Beziehungen zu anderen Termen lassen sich weitere Varianten eines Terms angeben, die nicht als eigenständiger Term existieren. Diese werden durch die Relation `Variant` abgebildet. Varianten sind immer genau einem Term zugeordnet. Als Typen werden *Abkürzungen* und *unerlaubte Synonyme* verwendet. Zu *unerlaubten Synonymen* zählen unter anderem auch falsche Schreibweisen und veraltete Begriffe.

Wie diese Datenstruktur im Detail eingesetzt wird und welche Eigenschaften dabei ausgenutzt werden wird in Kapitel 4 beschrieben.

2.3 Das SAP Research UIM-Framework

Von SAP Research wird im Rahmen des Forschungsprogramms *Data Management and Analytics* ein Framework zur deklarativen Informationsextraktion entwickelt. Informationsextraktion wird bei diesem neuartigen Ansatz als ein Prozess algebraischer Operationen aufgefasst. Die Algebra zur Informationsextraktion sowie der Aufbau und die Architektur des Frameworks sind detailliert von Barczyński et al. beschrieben, weshalb an dieser Stelle nur eine kurze Zusammenfassung gegeben wird [Barczyński et al., 2007].
Das Framework folgt einer komponentenbasierten Architektur. Es beinhaltet eine Menge von Operatoren, die als generische Komponenten umgesetzt sind und sich entsprechend der Algebra deklarativ kombinieren lassen. Ein Operator wird auf einer Menge von Annotationen angewendet. Annotationen werden als extrahierte Einheiten von Dokumenten aufgefasst. Die initiale Annotation ist das Dokument selbst. Das Ergebnis eines Operators ist ebenfalls eine Annotation. Es sind vier Arten von Operatoren definiert.

Basisoperatoren dienen dazu, unstrukturierten Daten, wie zum Beispiel Texten, grundlegende Eigenschaften und Entitäten zu entnehmen. Dazu werden unter anderem grammatikalische Regeln und Wörterbücher verwendet.

Verknüpfungsoperatoren kombinieren prinzipiell zusammenhängende Grundelemente zu komplexen Objekten.

Aggregationsoperatoren verbinden Annotationen mit realen, externen Entitäten.

Mengenoperatoren bieten die Möglichkeit Funktionen, wie Gruppierung, Vereinigung und Selektion, wie sie von Standard-Datenbankanfragesprachen wie SQL bekannt sind, auf Annotationen anzuwenden.

Abbildung 2.5 zeigt die funktionale Architektur des Frameworks, das den algebraischen Ansatz zur Informationsextraktion technisch umsetzt. Die oberste Schicht stellt die Datenintegration dar. Sie sorgt dafür, dass Inhalte für die Verarbeitung zur Verfügung gestellt werden. Die Verarbeitungsschicht, mittig im Bild dargestellt, unterteilt sich in Operatoren, Deskriptoren und die Ausführungseinheit. Am linken Bildrand sind die Operatoren (`Operators`)zu sehen. Die Deskriptoren (`Descriptors`), rechts davon, dienen zur Beschreibung von konkreten Annotatoren. Ein Annotator ist eine konkrete Instanz eines Operators bzw. einer Kombination von Operatoren.
Die Ausführungsschicht, am rechten Rand der mittleren Schicht, kümmert sich um die Ausführung konkreter Informationsextraktionsprozesse. Dazu werden Annotatoren zu einem Extraktionsplan (`Extraction Template`) verbunden. Der `Controller` nimmt einen Extraktionsplan entgegen und stellt die korrekte Abarbeitung sicher. Dabei bedient er sich des Entwurfsmusters der *Factory Method*, um aus der Beschreibung eines `Annotator Descriptors` einen konkreten `Annotator` zu erzeugen. Dieser wird anschließend nach dem Prinzip des *Visitors* beim `Analyzer` registriert. Der `Analyzer` ist mit der Persistenzschicht verbunden, welche dafür zuständig ist, die Ergebnisse der Informationsextraktion in einem generischen Datenbankschema abzuspeichern.
Auf eine detaillierte Abbildung des Schemas wird an dieser Stelle verzichtet und es sei nochmals an die Ausführungen von Barczyński et al. verwiesen. Das Datenbankschema des Frameworks gewährleistet, dass alle relevanten Daten eines Extraktionsplan nachvollziehbar und vollständig abgespeichert werden. Eine Annotation wird mit ihrem Inhalt und der Bereichsinformation (Start- und Endposition), bezogen auf das Ursprungsdokument, abgespeichert. Zu einer Annotation wird außerdem festgehalten mit welcher algebraischen Operation die Annotation erzeugt wurde. Optional können Informationen darüber gespeichert

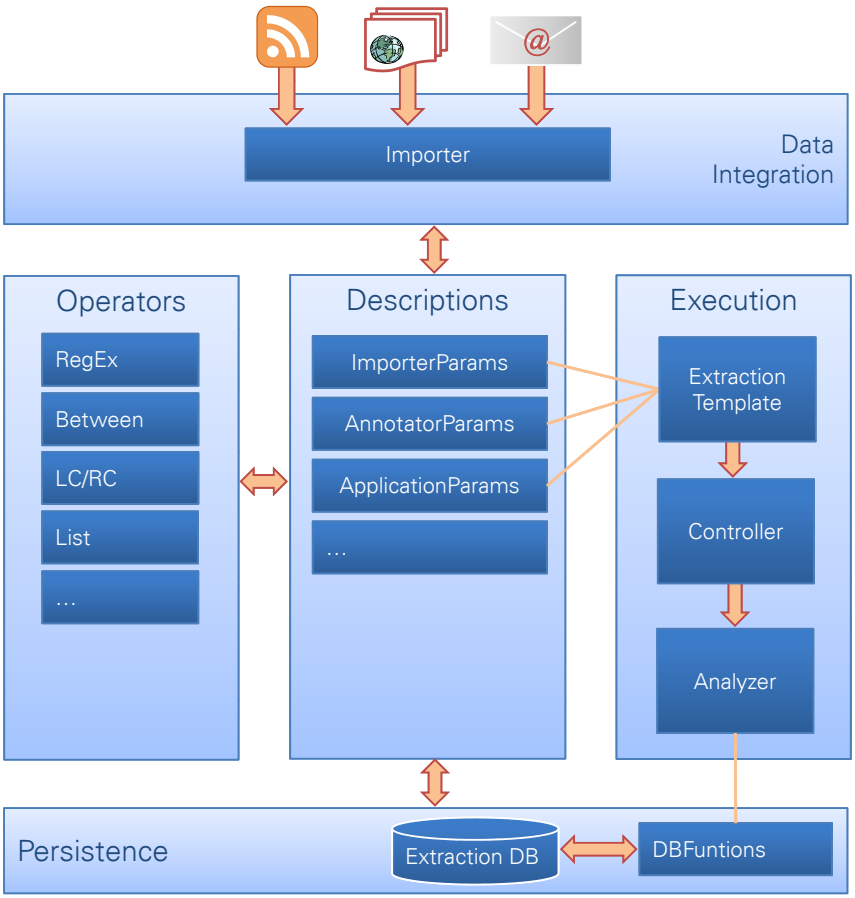


Abbildung 2.5: Architektur des SAP Research UIM Frameworks

werden, ob die Annotation in Verbindung zu anderen Annotationen steht, welcher Datenquelle sie angehört und zu welchen externen Entitäten sie zum Beispiel zugeordnet werden kann.
In Kapitel 5 wird dargestellt, in welcher Form die Ergebnisse dieser Arbeit in das Framework integriert wurden.

2.4 Qualitative Anforderungen

Im aktuellen Kapitel wurden bisher die Rahmenbedingungen dieser Arbeit vorgestellt. Es werden nun zunächst qualitative Anforderungen definiert, die die Güte der Ergebnisse des erarbeiteten Lösungsansatzes beschreiben.
Da die Erkennung von Entitäten unter anderem für die Zuordnung ähnlicher unstrukturierter Inhalte dient, ist die am höchsten bewertete Qualitätsanforderung die Genauigkeit bei der Identifikation von Entitäten. Erst danach reiht sich die Vollständigkeit, also die Anzahl erkannter Entitäten, ein.
Ähnlich wie die Vollständigkeit der Ergebnismenge wird auch die Performanz des vorgeschlagenen Ansatzes nicht mit höchster Priorität bewertet, sondern fügt sich eher an die dritte Stelle dieser Rangfolge ein. Da es sich nicht um eine Echtzeitanwendung handelt kann von entsprechenden Optimierungen abgesehen werden. Es soll vor allem sicher gestellt sein, dass die Datenmenge, die pro Zeiteinheit produziert wird (zum Beispiel 5000

Forumbeiträge pro Tag, siehe Abschnitt 2.2), auch in dieser Zeit verarbeitet werden kann. Dennoch sollten unnötige Ressourcenkosten vermieden werden. Aufgrund der Forderung nach möglichst hoher Genauigkeit ist es möglich, Abstriche bei der Performanz in Kauf zu nehmen, welche der Genauigkeit zu Gute kommen.

2.5 Funktionale Anforderungen

Die funktionalen Anforderungen zielen auf die konzeptionellen Aspekte und die Umsetzung der Lösung ab. Wie bereits der Einleitung zu entnehmen ist wird eine Komponente entwickelt, welche Instanzen von Entitäten einer gegebenen Datenstruktur in unstrukturierten Inhalten erkennt. Die dafür geltenden Anforderungen sind im Folgenden beschrieben.

Implementierung eines Operators Das SAP Research UIM-Framework wird, nach dessen Prinzipien, um einen Operator zur Identifikation von Enitäten erweitert werden. Der Operator fügt sich insbesondere nahtlos in dessen Struktur ein, sodass weiterhin die deklarative Beschreibung von Extraktionsplänen möglich ist.

Verwendung strukturierter Daten Der entwickelte Operator verwendet strukturierte Daten als Wissensbasis für die Identifikation von Entitäten in unstrukturierten Inhalten.

Erweiterung der Framework-Architektur Die Architektur des Frameworks wird um zusätzliche Konzepte erweitert, die es ermöglichen, strukturierte Daten für die Verarbeitung einzubinden. Diese Integration erfolgt generisch, sodass beliebige Operatoren auf beliebige strukturierte Daten zugreifen können.

Abstraktion von der konkreten Datenstruktur Die für die vorliegende Arbeit zur Verfügung stehenden Datenstrukturen dienen als Anhaltspunkt und für die Implementierung eines Prototyps. Für die Identifikation der Entitäten wird jedoch von der konkreten Datenstruktur abstrahiert, sodass dies auch mit weiteren Datenstrukturen möglich ist, deren Struktur und Instanzdaten sich von den hier vorliegenden Daten unterscheidet.

Die Überprüfung dieser Anforderungen wird in Kapitel 6 vorgenommen.

2.6 Metriken

Während sich funktionale Anforderungen relativ eindeutig und objektiv bewerten lassen, müssen insbesondere für die nichtfunktionalen Anforderungen Metriken definiert werden, die es ermöglichen, auch diese zu bewerten.
Zunächst muss klar gestellt werden, dass es sich bei den Forderungen nach Genauigkeit, Vollständigkeit und Performanz um drei konkurrierende Anforderungen handelt. Eine Verbesserung in Bezug auf eine Anforderung wird im Allgemeinen immer Einbußen in Bezug auf die beiden weiteren Anforderungen zur Folge haben. Das Ergebnis wird demnach einen pareto-optimalen Charakter aufweisen. Dafür sind die im vorherigen Abschnitt vergebenen Prioritäten maßgeblich.

2.6.1 Genauigkeit und Vollständigkeit

Precision und Recall sind klassische Metriken zur Bestimmung der Güte eines Rechercheverfahrens. Man kann sie als Genauigkeit (Precision) und Vollständigkeit (Recall) ins Deutsche übersetzen. Die Identifikation von Entitäten wird in der vorliegenden Arbeit als Rechercheverfahren aufgefasst.

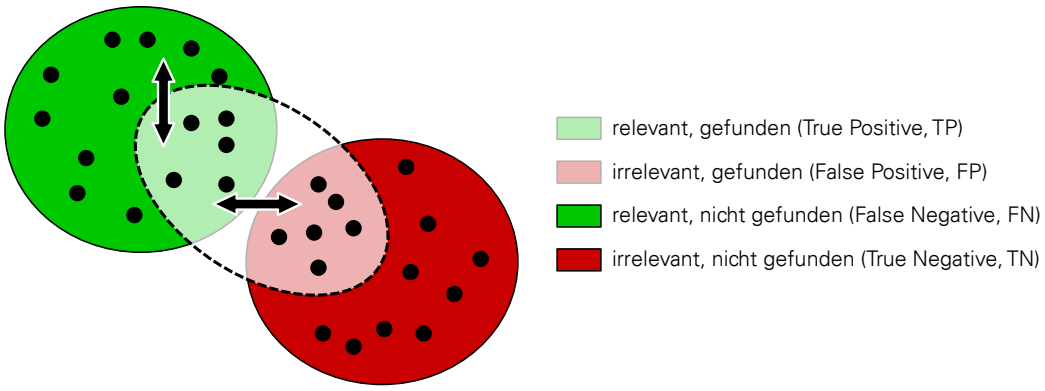


Abbildung 2.6: Zusammenhang von Precision und Recall

Abbildung 2.6 stellt den Zusammenhang von Precision und Recall dar. Die zwei großen Kreise bezeichnen die Gesamtheit aller Inhalte, die für ein Rechercheverfahren zur Verfügung stehen. Die Inhalte der roten Menge sind irrelevant, die der grünen Menge sind relevant. Das Oval umschließt die Menge der Inhalte, die durch das Verfahren gefunden wurden. Die Genauigkeit ist durch einen waagerechten Pfeil gekennzeichnet. Formell wird sie durch die Gleichung 2.2 beschrieben. Informell gesprochen drückt sie den Anteil relevanter Inhalte des Ergebnisses aus.
Die Vollständigkeit wird in Abbildung 2.6 durch einen senkrechten Pfeil dargestellt. Die Formelle Beschreibung erfolgt in Gleichung 2.2. Informell wird sie als Anteil gefundener relevanter Inhalte in Bezug zur Gesamtmenge relevanter Inhalte betrachtet.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Wie Eingangs des Abschnitts 2.6 beschrieben, handelt es sich bei Precision und Recall um konkurrierende Anforderungen. Aus diesem Grund werden sie stets in direktem Zusammenhang betrachtet. Eine Hilfestellung dafür bieten das F-Maß und E-Maß. Das F-Maß (Gleichung 2.3) ist das harmonische Mittel aus Precision(p) und Recall(r).

$$F = \frac{2}{\frac{1}{p} + \frac{1}{r}} \tag{2.3}$$

Sein Wertebereich liegt zwischen 0 und 1. Dabei bedeutet 0, dass keine relevanten Treffer erzielt und 1, dass alle relevanten Treffer erzielt wurden. Der maximal erreichbare Wert von F in einem bestimmten Kontext stellt den optimalen Kompromiss zwischen Precision und Recall dar. Das E-Maß (Gleichung 2.4), auch als Effektivitätsmaß bezeichnet, ist ebenfalls ein harmonisches Mittel aus Precision(p) und Recall(r).

$$E_\alpha = 1 - \frac{1 + \alpha^2}{\frac{\alpha^2}{r} + \frac{1}{p}} \tag{2.4}$$

Durch den Parameter α ermöglicht es jedoch die Priorisierung eines der beiden Werte. Für $\alpha = 1$ entspricht es dem F-Maß, weshalb dieses auch F_1-Maß genannt wird. $\alpha > 1$ gewichtet die Precision höher als den Recall und $\alpha < 1$ gewichtet den Recall höher als die Precision.

Ein Problem bei der Bestimmung von Precision und Recall besteht darin, dass man sie nicht immer eindeutig bestimmen kann. Dies ist der Tatsache geschuldet, dass in den meisten Fällen die Gesamtmengen relevanter (True Positive (TP) + False Negative (FN)) und irrelevanter (False Positive (FP) + True Negative (TN)) Inhalte nicht bekannt sind. Das macht die genaue Berechnung der Vollständigkeit unmöglich. Baeza-Yates und Ribeiro-Neto unterbreiten Vorschläge wie man dieser Problematik begegnen kann [Baeza-Yates und Ribeiro-Neto, 1999].
Welcher Weg für diese Arbeit gewählt wird, um Aussagen über die Genauigkeit und die Vollständigkeit des vorgestellten Ansatzes zu treffen, wird in Kapitel 6 beschrieben.

2.6.2 Performanz

Die Betrachtung von Performanzaspekten eines Softwaresystems kann in vielen Richtungen erfolgen. Dazu gehört zum Beispiel die Auslastung von Systemressourcen wie Prozessorbelastung, Arbeitsspeicherverbauch und Netzwerklast. Auf eine detaillierte Analyse dieser Parameter und deren Abhängigkeiten wird in dieser Arbeit verzichtet. In Anlehnung an die Priorisierung aus Abschnitt 2.4 wird für die Evaluation dieser Arbeit der Datendurchsatz auf der Hardwarekonfiguration eines Standard-Entwicklungssystems gemessen. Ein Belastungstest wird zeigen inwiefern die in Abschnitt 2.2.1 beschriebenen Datenmengen unstrukturierter Inhalte durch das System verarbeitet werden können.

2.7 Herausforderungen

Betrachtet man die in diesem Kapitel vorgestellten Voraussetzungen und Anforderungen für diese Arbeit stellt sich die Frage, ob es nicht möglich ist auch mit einem naiven Ansatz ausreichend gute Ergebnisse zu erzielen. Die Erarbeitung eines qualitativ ansprechenden Lösungsansatzes ist durchaus nicht trivial. Um dies zu zeigen wird in diesem Absatz ein naiver Ansatz vorgestellt. Ausgehend von diesem Ansatz lassen sich Herausforderungen ableiten, die sich aus der gestellten Aufgabe ergeben.
Folgende Bedingungen liegen diesem ersten Lösungsansatz zu Grunde. Aus SDN-Forumseiten wurde eine Menge von 1000 Links extrahiert deren Inhalte (Text, der von Anker-Tag eingeschlossen ist) genau 100 SAP-Produktinstanzen enthalten. Diese 1000 Ankertexte werden in einer einfachen Textdatei als Liste abgespeichert. Darüber hinaus liegt eine weitere Textdatei vor, die eine Liste mit 143 Produktnamen von SAP enthält. Anhand dieser Liste sollen die 100 Vorkommen von Produktinstanzen in den 1000 Links erkannt werden.
Die Umsetzung erfolgt in einem einfachen Java-Programm. Als erstes wird die Datei mit den Produktnamen geladen und als Listenstruktur im Speicher gehalten. Die Ankertexte können nun einzeln daraufhin überprüft werden, ob sie mit einem Eintrag in der Produktliste übereinstimmen. Der Aufwand hier beträgt im Maximalfall 143'000 Vergleiche von Zeichenketten (Java-Strings). Das Ergebnis liefert kein einziges gefundenes Produkt. Der Grund dafür ist, dass die Ankertexte, wenn vorhanden, jeweils nicht nur das Produkt enthalten sondern noch zusätzlichen Text. Die zweite Möglichkeit ist die Methode umzukehren und für jeden Eintrag der Produktliste zu überprüfen, ob dieser in einem Ankertext enthalten ist. Dazu kann die Java-String-Methode `String.contains(String)` genutzt werden. Mit dieser Methode werden von 100 möglichen Produkten 44 erkannt, was einem Recall von 44% entspricht. Allerdings werden auch fälschlicherweise 165 Produktinstanzen in Ankertexten erkannt, in denen keine vorhanden sind. Dies führt zu einer Precision von 21%, was den Anforderungen keinesfalls gerecht wird.

Ankertext	**Produktname**	**ist**	**soll**
Netweaver Installation	SAP NetWeaver 2004s	FN	TP
R**ep**etitive Manufacturing	EP	TN	FP
SAP **Solution Manager**	Solution Manager	TP	TP

Tabelle 2.1: Beispiel eines einfachen String-Vergleichs

Tabelle 2.1 zeigt, welche Probleme bei einer naiven Herangehensweise am häufigsten auftreten. Es werden drei Beispiele gezeigt bei denen links der Ankertext dargestellt wird wie er im Dokument vorhanden ist. Rechts daneben befindet sich der Produktname der im Text erkannt wurde bzw. erkannt werden sollte. Ist- und Sollwerte lassen sich aus den Spalten rechts ablesen. Dabei wird die in Abschnitt 2.6.1 eingeführte Notation verwendet. Die im Ankertext fett dargestellten Teile markieren die Übereinstimmung mit dem Produktnamen, wodurch deutlich wird, warum das Produkt identifiziert wurde oder nicht. Daraus lassen sich einige prinzipielle Herausforderungen ableiten. Zum einen müssen Methoden entwickelt werden, die unstrukturierte Inhalte bis auf einzelne Wörter zerlegen. Um den Aufwand zu Verringern muss die Menge an Wörtern, die an die Datenstruktur herangeführt wird geeignet minimiert werden, sodass nur relevante Inhalte verarbeitet werden. Der Anfragemechanismus muss darüber hinaus eine gewisse Unschärfe zulassen, damit ein Vergleich wie im ersten Beispiel in Tabelle 2.1 einen positiven Treffer erzielt.
Eine weitere Herausforderung besteht in der Einbindung der strukturierten Daten. In dem hier vorgestellten Beispiel wird eine einfache Liste verwendet deren Umfang ohne Weiteres die Verwendung im Hauptspeicher zulässt. In Abschnitt 2.2 wurden jedoch weitaus umfangreichere Daten vorgestellt, die außerdem eine komplexe Struktur aufweisen. Das bedeutet, dass ausgereifte Mechanismen entwickelt werden die sowohl die Abbildung der Datenstruktur als auch Anfrage an diese Datenstruktur sicherstellen. Desweiteren müssen diese Mechanismen performant sein und die korrekte Identifikation von Entitäten ermöglichen.

2.8 Zusammenfassung

Dieses Kapitel hat sowohl die Voraussetzungen als auch die Anforderungen für den hier vorgestellten Lösungsansatz beschrieben. Dies beinhaltet vor allem die Beschreibung der Anwendungsfälle sowie der zu Grunde liegenden Daten. Mit dieser Arbeit soll erreicht werden, dass Produktinstanzen anhand strukturierter SAP-Produkt- und Terminologiedaten in Foruminhalten des SDN identifiziert werden. Es wurden außerdem Metriken beschrieben, die es ermöglichen, die Ergebnisse qualitativ zu bewerten. Darüber hinaus wurde anhand einer Beispielimplementierung gezeigt, dass einige nicht-triviale Herausforderungen zu bewältigen sind, um ein qualitativ akzeptables Ergebnis für die Identifikation von Entitäten zu erzielen.

Kapitel 3

Stand der Technik

Nachdem in Kapitel 2 die der Arbeit zu Grunde liegenden Anforderungen definiert wurden, wird in diesem Kapitel der aktuelle Stand der Technik analysiert. Die Informationsextraktion stellt einen sehr breiten Forschungs- und Anwendungsbereich dar. Deshalb wird entsprechend der Anforderungen eine Einschränkung für die vorliegende Analyse getroffen. Ziel der Arbeit ist es, Entitäten strukturierter Daten in unstrukturierten Informationen zu identifizieren. Dies stellt ein Teilgebiet der Informationsextraktion dar, welches allgemein als Named Entity Recognition (NER) bezeichnet wird. In der Literatur wird synonym der Begriff Named Entity Recognition and Classification (NERC) verwendet. Als deutsche Übersetzung dient in dieser Arbeit *Identifikation von Entitäten*. Die Betrachtungen dieses Kapitels beschränken sich auf diesen Bereich. Im Folgenden wird ein Überblick zu klassischen Verfahren von NER gegeben. Aus diesem Überblick werden Kategorien abgeleitet, die anschließend vorgestellt werden. Letztendlich wird zum Hauptteil, der Analyse verwandter Arbeiten aus Forschung und Entwicklung, übergegangen. Die Arbeiten werden in die zuvor vorgestellten Verfahren eingeordnet. Abschließend erfolgt eine Bewertung der Verfahren woraus sich jeweils eine Abgrenzung zu der vorliegenden Arbeit ableitet.

3.1 Überblick klassischer Named Entity Recognition

Wie für die Informationsextraktion allgemein tut sich auch für das Teilgebiet der Named Entity Recognition ein sehr weites Feld auf. Dieser Abschnitt dient dazu, einen allgemeinen Überblick zu Techniken und Ansätzen in diesem Bereich zu geben. Eine eindeutige Unterteilung lässt sich nicht sehr leicht vornehmen. In verschiedenen Quellen werden unterschiedliche Gruppierungen vorgenommen. Sehr weit verbreitet ist die Unterscheidung zwischen Wissensmodellierung (engl. *Knowledge Engineering*) und Maschinellem Lernen (engl. *Machine Learning*). Die Wissensmodellierung wird auch als deklarative NER bezeichnet und ist wiederum in listenbasierte und regelbasierte Verfahren unterteilt. Unabhängig davon, für welche Einteilung man sich entscheidet, ist es immer schwierig einen konkreten Ansatz in eine feste Kategorie einzuordnen. Oft werden verschiedene Verfahren miteinander kombiniert. Die Betrachtung verwandter Arbeiten wird darauf eingehen, welche der genannten Aspekte sich in ihnen wiederfinden. Als erstes werden die klassischen Verfahren vorgestellt. Sie werden nach listenbasierten, regelbasierten und Verfahren zum Maschinellen Lernen unterteilt.

3.1.1 Listenbasierte Verfahren

Listenbasierte Verfahren stellen die einfachste Form der Identifikation von Entitäten dar. Ein Beispiel dafür ist der naive Ansatz, der in Abschnitt 2.7 beschrieben wurde. Die Idenfikation unstrukturierter Inhalte wird auf den Abgleich mit einer Liste reduziert. Die Liste, auch als Lexikon oder Wörterbuch (engl. *dictionary*) bezeichnet, muss dazu explizit alle zu identifizierenden Ausprägungen von Entitäten enthalten. Dazu gehören nicht nur Synonyme sondern auch Varianten wie Abkürzungen und abweichende Schreibweisen. Für stark eingeschränkte Domänen mit eindeutigen Begriffen und geringem Umfang des Wörterbuchs liefern listenbasierte Verfahren eine gute Präzision. Je weniger abgeschlossen eine Domäne ist, desto größer wird das ihr zugeordnete Wörterbuch und desto höher ist der Aufwand für den Abgleich mit der Liste. Mit der Größe des Wörterbuchs steigt außerdem die Auftrittswahrscheinlichkeit von Mehrdeutigkeiten. Sie lassen sich mit rein listenbasierten Verfahren nicht auflösen.

3.1.2 Regelbasierte Verfahren

Unstrukturierte Inhalte werden überwiegend durch natürlichsprachige Dokumente repräsentiert. Diese sind nach grammatikalischen Regeln aufgebaut. Regelbasierte Verfahren nutzen diese Eigenschaft, um anhand eigener Grammatiken Entitäten zu identifizieren. Teilweise werden mehrere Grammatiken erstellt, die stufenweise auf einem Dokument oder Korpus ausgeführt werden (engl. *Cascading Grammars*). Dabei dient zum Beispiel die zuerst ausgeführte Grammatik zur Erkennung von Substantiven und die zweite für die Identifikation einer Entität.
Betrachtet man die Phrase `SAP NetWeaver 2007` sind zwei Stufen vorstellbar. Stufe eins identifiziert `SAP` als *Unternehmen*, `NetWeaver` als *Produkt* und `2007` als *Zahl*. Die zweite Regel besagt nun, dass

```
<Unternehmen>+<LEERZEICHEN>+<Produkt>+<LEERZEICHEN>+<Zahl>
```

als eine Langschreibweise für Produkte interpretiert wird und die Entitäten entsprechend der Form

```
<Hersteller><Produktname><Version>
```

benannt werden. Da solche Regeln sehr komplex werden können, viele Regeln benötigt werden und diese vorrangig manuell erstellt werden, bedeutet dies einen sehr großen Zeitaufwand für die Einführung eines regelbasierten Systems. Ebenso ist die Übertragung auf einen anderen Kontext oder die Anwendung auf eine andere Sprache gleichbedeutend mit der Neuerstellung der überwiegenden Mehrheit an Regeln.

3.1.3 Maschinelles Lernen

NER-Verfahren, die sich den Methoden des Maschinellen Lernens bedienen, haben sich nach Nadeau und Sekine gegenüber anderen Verfahren durchgesetzt [Nadeau und Sekine, 2007]. Sie sind somit auch am weitesten verbreitet. Sie lassen sich in *Überwachtes Lernen* (engl. *supervised learning*), *Teilüberwachtes Lernen* (engl. *semi-supervised learning*) und *Selbstständiges Lernen* (engl. *unsupervised learning*) untergliedern. Die Verfahren werden von Nadeau und Sekine beschrieben.
Als **Überwachtes Lernen** versteht man im Allgemeinen die Anwendung stochastischer Verfahren zur Identifikation von Entitäten in unstrukturierten Daten. Dabei kommen zum Beispiel das Hidden Markov Model (HMM) oder die Maximum Entropy Method (MEM) zum

Einsatz [Fink, 2003; Wu, 1997]. Ausgangspunkt ist ein großer Textkorpus in dem zu erkennende Entitäten gekennzeichnet sind. Er wird als *Trainingsdatensatz* verwendet, der in der Lernphase vom NER-System eingelesen wird. In dieser Phase wird ein Modell unter anderem anhand syntaktischer Eigenschaften, wie zum Beispiel Groß- und Kleinschreibung erstellt. Dieses Modell enthält Statistiken über diese Eigenschaften und bringt sie mit gekennzeichneten Entitäten in Verbindung. Dabei fließt zum Beispiel mit ein, welche Arten von Wörtern vor und nach einer Entität auftauchen, welche syntaktischen Eigenschaften die Entitäten aufweisen und vieles mehr. Anhand der Regeln kann in der Anwendungsphase für jedes Textsegment berechnet werden, wie hoch die Wahrscheinlichkeit ist, dass es sich um eine der gesuchten Entitäten handelt. Das NER-System bezieht dementsprechend sein gesamtes Wissen aus dem genannten Trainingskorpus. Dieser Trainingskorpus ist statisch. Folglich hängt die Genauigkeit davon ab, wie umfangreich der Korpus ist und wie gut die Eigenschaften der zu bearbeitenden Daten dadurch repräsentiert werden. Nadeau und Sekine stellen Ansätze vor, bei denen der Recall je nach Anwendungsdomäne (25% für Personen, 50% für Organisationen, 75% für Orte) stark variiert.

Teilüberwachtes Lernen unterscheidet sich vom Überwachten Lernen in dem Punkt, dass das System nicht aus einem statischen Trainingsdatensatz lernt, sondern diesen während der Arbeit ständig erweitert und zustätzliche Regeln für die Identifikation generiert werden. Man bezeichnet dies als *bootstrapping*, auf Deutsch, Hochfahren eines selbsterzeugenden Systems oder Kreislaufs. Das bietet den Vorteil, dass der Trainingsdatensatz nicht sehr umfangreich sein muss, was den Aufwand bei dessen Erzeugung erheblich reduziert.

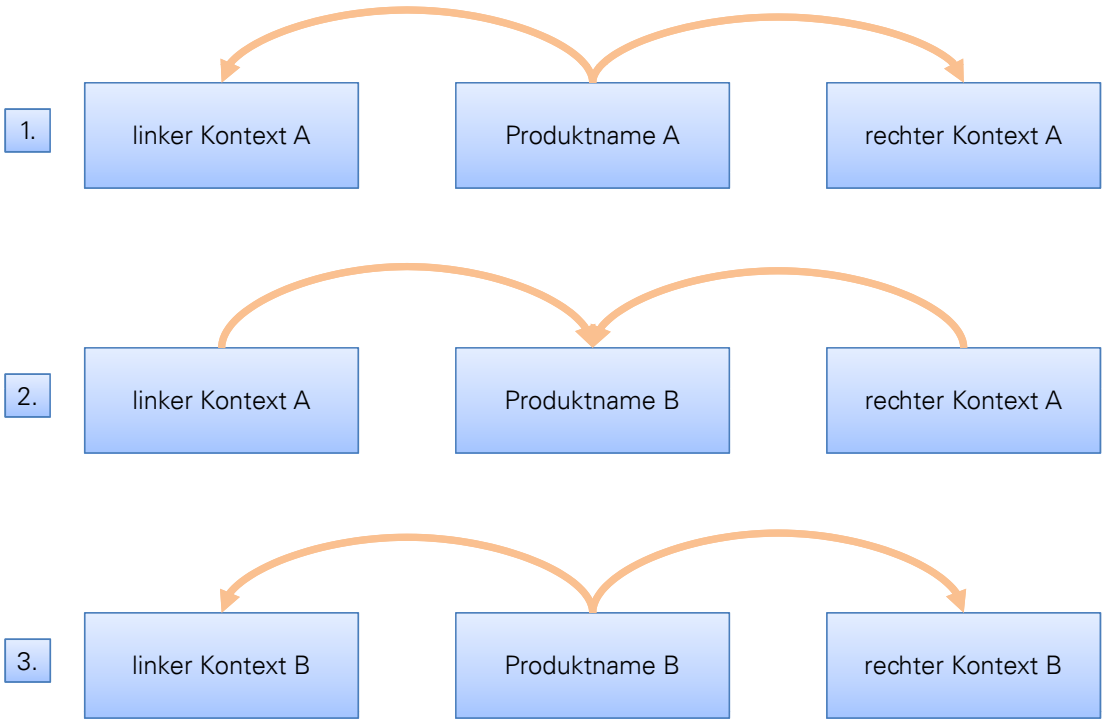


Abbildung 3.1: Beispiel für Teilüberwachtes Lernen

Ein Beispiel für Teilüberwachtes Lernen wird in Abbildung 3.1 dargestellt und im Folgenden kurz erläutert. Ein NER-System wird mit einem kleinen Korpus initialisiert, in dem einige wenige Produktnamen, wie *Produktname A*, gekennzeichnet sind. Das System wird mit einer Regel ausgestattet, die die Eigenschaft *linker und rechter Kontext eines Produktnamen* nutzt, um Produktnamen anhand der entsprechenden Zeichenketten rechts und links eines

Wortes zu identifizieren.

1. Der linke und rechte Kontext eines identifizierten Produktnamen wird gespeichert. (*linker und rechter Kontext A*)

2. Im Verlaufe der Anwendung wird in diesem Kontext ein weiteres Produkt erkannt. (*Produktname B*)

3. Das neu erkannte Produkt kommt wiederum in einem weiteren Kontext vor. (*linker und rechter Kontext B*)

4. Die Schritte 1-3 werden im Verlaufe der Anwendung wiederholt, wodurch der Trainingskorpus ständig erweitert wird.

Selbstständiges Lernen kommt vollkommen ohne Trainingskorpus aus. Hierbei wird eine Menge von Entitäten ohne Trainingsschritt übergeben, welche zu identifizieren sind. Im Gegensatz zu einem listenbasierter Ansatz werden die Eigenschaften der vorhandenen Entitäten untersucht. Auf Basis dieser Eigenschaften können weitere Entitäten erkannt werden. Im Allgemeinen kommen dazu lexikalische Eigenschaften und Statistiken zum Einsatz. Neben der soeben genannten Methode des Clustering (*dt. Klassifizierung*), welche Entitäten bestimmter Typen gruppiert an das System übergibt, kommen auch weitere Verfahren zum Einsatz. Baronchelli et al. haben zum Beispiel einen Ansatz entwickelt, der auf Eigenschaften von Datenkompressionsmethoden zurückgreift um Informationsextraktion zu realisieren [Baronchelli et al., 2004].
Die bisher vorgestellten Verfahren adressieren vor allem allgemeine Anwendungsgebiete wie das Erkennen von Personen oder Orten. Sie werden in domänenspezifischen Ansätzen teilweise kombiniert eingesetzt, um Lösungen für die konkreten Problemstellungen zu bieten.

3.2 Verwandte Arbeiten

In diesem Abschnitt wird eine Auswahl verwandter wissenschaftlicher Arbeiten betrachtet, die unter ähnlichen Voraussetzungen oder Zielstellungen wie die vorliegende Arbeit entwickelt wurden. Gemeinsam ist ihnen, dass die Identifikation von Entitäten in unstrukturierten Daten eine Rolle spielt und dass strukturierte Daten dafür als Wissenbasis verwendet werden. Sie unterscheiden sich in ihrer prinzipiellen Zielvorgabe und der Art der eingesetzten strukturierten Daten.

3.2.1 Integration von strukturierten und unstrukturierte Daten

Die Integration von strukturierten und unstrukturierten Daten ist das Forschungsthema einer Gruppe des Indian Institute of Technology in Bombay[1]. Ein Aspekt dieser Arbeit widmet sich der Identifikation von Entitäten mit Hilfe strukturierter Daten. Der entwickelte Ansatz lässt sich als teilüberwachtes Lernen einordnen, jedoch ist die Erweiterung der Regelbasis komplexer als in Abschnitt 3.1.3 beschrieben.
Mansuri und Sarawagi beschreiben das System zur Integration von unstrukturierten Daten in relationale Datenbanken [Mansuri und Sarawagi, 2006]. Sie schließen damit die Lücken, die sie darin begründet sehen, dass viele selbstlernende Verfahren (Abschnitt 3.1.3) unstrukturierte Daten in eine Struktur überführen, die Struktur an sich aber nicht für die Identifikation

[1] http://www.iitb.ac.in

von Entitäten nutzen. Die Autoren gehen noch einen Schritt weiter. Die Datenstruktur wird in der Lernphase zum Aufbau eines Modells verwendet. Während der Anwendungsphase werden die Instanzdaten als Evidenzen für die Identifikation von Entitäten genutzt und außerdem durch neu erkannte Daten erweitert. Die detaillierte Systemarchitektur wird im Folgenden beschrieben.

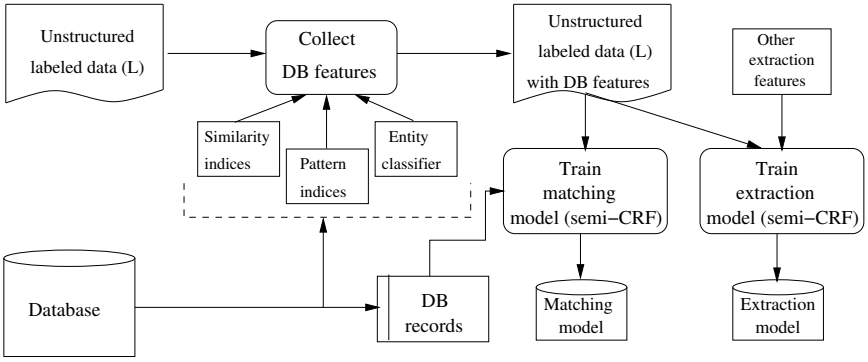


Abbildung 3.2: Lernphase aus [Mansuri und Sarawagi, 2006, S. 7]

Abbildung 3.2 zeigt die Lernphase des Systems in der zwei Modelle entstehen, das *Matching model* und das *Extraction model*. Das *Extraction model* dient der Identifikation von Entitäten in unstrukturierten Daten. Das *Matching model* hilft dabei herauszufinden, ob eine erkannte Entität bereits exakt oder in ähnlicher Form in der Datenbank vorliegt.
Ausgangspunkt der Modellierung ist ein Trainingsdatensatz mit gekennzeichneten Entitäten, wie in Abschnitt 3.1.3 beschrieben. Dieser Datensatz wird mit drei Charakteristiken der Datenbasis angereichert. Die erste Charakteristik wird durch Ähnlichkeitsindizes (*Similarity indices*) beschrieben. Zur Erstellung dieser Indizes wird auf den Instanzdaten eine Funktion ausgeführt, die in der Schreibweise leicht abweichenden, konkreten Werten einen gemeinsamen Ähnlichkeitswert zuweist. Dies ermöglicht die effiziente Durchführung von Ähnlichkeitsvergleichen auf der gesamten Datenbasis. Dieses Prinzip wird von Cohen und Sarawagi beschrieben [Cohen und Sarawagi, 2004]. Die zweite Charakteristik der Datenbasis sind Muster, welche ebenfalls indiziert werden. Sie lassen sich zum Beispiel als reguläre Ausdrücke darstellen. Anhand von Mustern lässt sich feststellen, ob ein Textsegment einem bestimmten Attribut einer Entität entspricht, auch wenn es keine syntaktische Ähnlichkeit aufweist. Die zwei bisher beschriebenen Charakteristiken werden auch während der Anwendungsphase regelmäßig aktualisiert und in die Modelle übertragen. Die dritte Charakteristik wird nur einmalig für die Lernphase bestimmt. Es handelt sich um die Klassifizierung von Entitäten über Eigenschaften, die durch die ersten zwei Charakteristiken noch nicht abgedeckt sind. Dazu gehören zum Beispiel die typische Länge einer Entität oder charakteristische Wörter. Ebenso werden sogenannte *negative Eigenschaften* erfasst, die man als Abbruchkriterium bei der Erkennung von Entitäten verwenden kann. Sie repräsentieren Eigenschaften, die eine Entität auf keinen Fall hat.
Der nun vorhandene Trainingsdatensatz wird für die Konditionierung des *Extraction models* verwendet. Für die Erstellung des *Matching models* kommen zusätzlich noch die reinen Datenbankeinträge der im Trainingsdatensatz enthaltenen Entitäten zum Einsatz. Der Unterschied zwischen den beiden resultierenden Modellen besteht demnach im Wesentlichen darin, dass das *Matching model* zusätzlich noch explizite Repräsentationen der Datenbasis enthält. Das Ergebnis der Konditionierung sind *Semi-Markov Conditional Random Fields (Semi-CRFs)*, eine spezielle Form eines Hidden Markov Models. Die Details dieses Prinzips fallen nicht in den Rahmen dieser Arbeit. Dazu sei an [Sarawagi und Cohen, 2005] verwie-

sen.

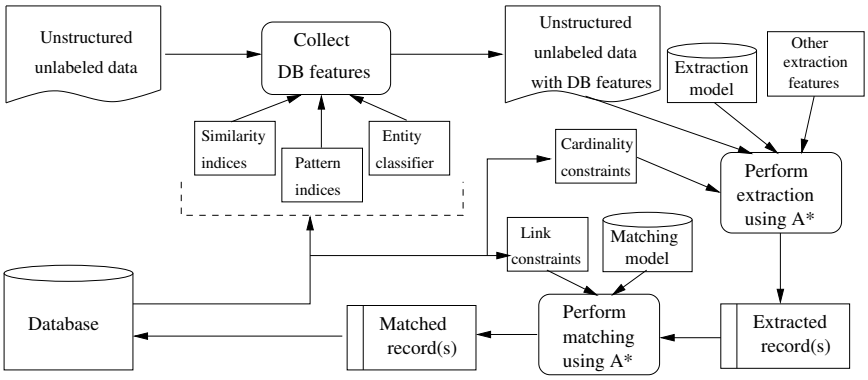


Abbildung 3.3: Anwendungsphase aus [Mansuri und Sarawagi, 2006, S. 7]

Die Anwendungsphase, dargestellt in Abbildung 3.3, verfolgt zwei Ziele. Zum Einen werden Entitäten in unstrukturierten Daten identifiziert und dort entsprechend gekennzeichnet. Zum Anderen wird die Datenbasis durch neu erkannte Entitäten erweitert. Für den ersten Aspekt, der Extraktion von Entitäten, kommen neben dem *Extraction model* und dem Trainingsdatensatz noch Kardinalitätseigenschaften (*Cardinality constraints*) der Datenstruktur zum Einsatz. Die Details der Extraktion werden später während der Beschreibung der Arbeit von Chandel et al. dargestellt. Mit den Kardinalitätseigenschaften steht eine zusätzliche Evidenz bei der Identifikation zur Verfügung. Der Datenabgleich wird zusätzlich durch Datenverknüpfungseigenschaften (*Link constrainst*) unterstützt. Zu den neu erkannten Entitäten gehören auch Variationen von bereits in der Datenbank befindlichen Entitäten. Variationen werden jeweils auf eine kanonische Form zurückgeführt. Dazu wird folgendes Beispiel aufgeführt. In unstrukturierten Daten wird ein Buch gefunden, welches bereits in der Datenbank vorhanden ist. Es wird festgestellt, dass der Autor nicht mit vollem Namen genannt wird sondern nur mit dem Initial des Vornamens und dem kompletten Nachnamen. In diesem Fall wird die *neue* Form des Autors als Variation in der Datenbank gespeichert. Eine Fremdschlüsselbeziehung stellt die Verknüpfung mit der kanonischen Form her. Durch Datenverknüpfungseigenschaften kann zum Beispiel überprüft werden, ob die Integration einer neuen Variante vollständig plausibel ist.

Zur Evaluation wurde als unstrukturierte Information ein Datensatz mit 100 Zitatseiten von Citeseer verwendet. Als strukturierte Datenquelle dient eine Publikationsdatenbank, die 45% der in den unstrukturierten Daten enthaltenen Titel sowie 75% der Autoren abdeckt. 5% der strukturierten Daten wurden für die Erstellungen des Trainingsdatensatzes verwendet. Entsprechend den Zielen dieser Entwicklung sollten sowohl die Entitäten in den unstrukturierten Daten identifiziert als auch die fehlenden Einträge in der Datenbank ergänzt werden. Die Ergebnisse von Extraktion, Datenabgleich und Datenintegration wurden jeweils mit Ergebnissen verglichen, in denen keine strukturierten Daten in die Anwendung mit einbezogen wurden. In allen drei Bereichen sind die Ergebnisse unter Verwendung der Datenbank signifikant besser als ohne Datenbank. Dennoch schwanken die absoluten Resultate je nach Datentyp, zum Beispiel Autor oder Titel. So liegt der Recall für die Extraktion von Journaltiteln bei 48%, Precision bei 56% wobei Autoren mit einem Recall von 85% und einer Precision von 74% extrahiert wurden. Die Zahlen beziehen sich auf die Ausführung des Systems unter Verwendung der strukturierten Daten.

Bisher wurde ein Aspekt dieses Systems noch gar nicht betrachtet - die eigentliche Identifikation von Entitäten, wozu ein Abgleich der strukturierten und unstrukturierten Daten er-

forderlich ist. **Chandel et al.** beschreiben die Entwicklung eines sehr effizienten Verfahrens, welches dafür zum Einsatz kommt [Chandel et al., 2006]. Es handelt sich um ein erweitertes Top-k Suchverfahren.
Identifizierung von Entitäten in unstrukturierten Texten bedeutet, dass zunächst einmal jedes Wort eine Entität repräsentieren könnte. Ein Wort wird in diesem Zusammenhang als Segment eines Textes aufgefasst. Mehrere Worte können zusammengefasst werden und bilden ebenfalls ein Segment. Jedes mögliche Segment eines Textes ist demnach ein Kandidat für eine Entität. Anders formuliert lautet das Ziel, eine optimale Segmentierung des Textes zu finden, bei der jedes Segment entweder durch seine korrekte Entität oder als sonstiger Text gekennzeichnet ist. Möchte man Entitäten anhand strukturierter Daten, in diesem Fall einer relationale Datenbank, identifizieren, so ist der naive Weg alle möglichen Segmente eines Textes zu bilden und diese an die Datenbank anzufragen. Die Zahl der resultierenden Datenbankanfragen ist exponentiell zur Größe des Textes.
Chandel et al. schlagen zunächst die Einschränkung der maximale Länge eines Segments auf einen sehr kleinen Wert vor. Dadurch steigt die Anzahl der Anfragen nur noch linear mit der Größe eines Textes. Für jedes Segement kann eine Top-k Suche, eine Suche nach den k besten Treffern, erfolgen. Da nach wie vor bei der Segmentierung überlappende Segmente entstehen, verbleiben zwei Probleme. Zum Einen werden Anfragen an die Datenbank für überlappende Segmente mehrfach ausgeführt. Zum Anderen muss aus den Ergebnissen überlappender Segmente die beste Segmentierung ausgewählt werden, sodass die korrekten Entitäten getroffen werden. Zur Optimierung der Anfragen für überlappende Segmente wurde die einfache Top-k Suche zu einer *Batch Top-k Suche*, einer gebündelten Top-k Suche, erweitert. Ergebnisse von Anfragen, die für folgende Segmente möglicherweise noch einmal benötigt werden, werden in einem Cache vorgehalten. Die Top-k Berechnung wird jedoch für jedes Segment unabhängig durchgeführt. Sie erfolgt anhand der TF-IDF-Werte, die für die Übereinstimmung mit einer Entität ermittelt wurden. Das TF-IDF-Verfahren wird unter anderem von Cohen et al. beschrieben [Cohen et al., 2003]. Weiterhin wird es in Abschnitt 4.4 noch einmal aufgegriffen. Die Auswahl der besten Segmentierung ist ein Optimierungsproblem, welches unter Anwendung der Dynamischen Programmierung gelöst wird. Es werden außerdem noch Verbesserungen dieses dynamischen Programmieransatzes beschrieben, die jedoch in erster Linie auf die Performanz des Systems abzielen. Auch die experimentellen Ergebnisse werden vordergründig in Bezug auf die Effizienz dieses Ansatzes ausgewertet. Für die vorliegende Arbeit sind hauptsächlich die Ergebnisse von Recall und Precision des Gesamtsystems von Mansuri und Sarawagi entscheidend, welche bereits ausgewertet wurden.

3.2.2 Informationsextraktion auf Basis von Ontologien

Informationsextraktionssysteme, die auf der Basis von Ontologien arbeiten, lassen sich nur schwer in die in Abschnitt 3.1 eingeführten Kategorien einordnen. Eine Möglichkeit ist es, die Instanzdaten von Ontologien als Listen zu betrachten, jedoch beziehen diese Systeme ihr umfangreiches Domänenwissen vor allem auch aus der Datenstruktur. Aus diesem Grund stellen sie auch eine starke Verwandtschaft zu dem in dieser Arbeit vorgestellten Ansatz.
An dieser Stelle werden zwei Arbeiten aus der aktuellen Forschung vorgestellt, die beide prinzipiell mit der Erkennung von Entitäten in unstrukturierten Daten zusammenhängen. Im ersten Ansatz [Cheng et al., 2003] ist dies nicht auf den ersten Blick offensichtlich, da eine Kategorisierung von unstrukturierten Daten angestrebt wird. Als Voraussetzung dafür müssen aber Entitäten in den unstrukturierten Daten identifiziert werden, welche aggregiert werden, um die Bedeutung eines Dokuments wiederzugeben. Die ursprünglich identifizierten Entitäten bleiben verborgen, da man dem Ergebnis nur die Kategorie eines Dokuments

entnehmen kann.
Die zweite Arbeit von Hassell et al. lenkt den Betrachter allein aufgrund des Titels umgehend auf das Gebiet der NER[Hassell et al., 2006]. Es wird die Problemstellung adressiert, dass insbesondere die Erkennung von Personen aufgrund häufig vorkommender Namen sehr stark mit Mehrdeutigkeiten behaftet ist. Die Einbeziehung von Ontologien soll die Erschließung eines breiteren Kontexts, um im Text identifizierte Entitäten herum, ermöglichen. Dadurch können diese Mehrdeutigkeiten aufgelöst werden.
Cheng et al. haben ein Framework zur semantischen Klassifikation von Dokumenten, das *Ontology-based Semantic Classification (OSC) framework*, entwickelt. Es identifiziert Entitäten in unstrukturierten Texten, welche anhand dieser Entitäten klassifiziert werden. Dazu werden unter anderem Ontologien als Wissensbasis verwendet.
Der Gesamtprozess betrachtet vier Facetten, welche durch unterschiedliche Systemkomponenten umgesetzt werden. Die erste Facette ist die syntaktische Analyse des unstrukturierten Texts. Dazu kommt der *Link Grammar Parser* von Sleator und Temperley (Carnegie Mellon Universität[2]) zum Einsatz [Sleator und Temperley, 1993]. Dieser Parser zerlegt einen Text in einzelne Sätze und stellt satzweise die syntaktische Struktur dar.

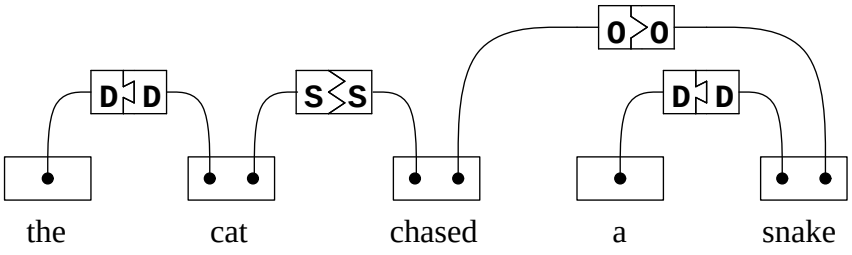


Abbildung 3.4: Ergebnis des Link Grammar Parsers aus [Sleator und Temperley, 1993, S. 2]

Dazu gehört die Einteilung in Worttypen (Substantiv, Verb, ...) und die Verbindung von Wörtern zu syntaktisch korrekten Einheiten. In dem Beispiel in Abbildung 3.4 werden unter anderem die Wörter *chased* und *snake* durch einen *O*-Operator (Prädikat-Objekt) verbunden. Weiterhin sind die Operatoren *D* (Artikel-Substantiv) und *S* (Subjekt-Prädikat) zu sehen.
Die semantische Analyse wird durch zwei weitere Facetten widergespiegelt. In einem ersten Schritt wird die rein lexikalische, kontextunabhängige Semantik analysiert während im zweiten Schritt Kontextwissen in die Analyse einbezogen wird. Als semantisches Modell für die kontextunabhängige Semantik wird die WordNet-Datenbank genutzt, welche die weltweit größte lexikalische Sammlung englischer Begriffe enthält. Eine wichtige Eigenschaft von WordNet sind die enthaltenen Synonymbeziehungen, die Begriffe zu lexikalischen Konzepten gruppieren.
Die kontextabhängige Semantik wird in einem Kontextmodell in Form einer Ontologie dargestellt. Sie ordnet den lexikalischen Konzepten, die durch WordNet dargestellt werden, eine Bedeutung bezogen auf den zu betrachtenden Kontext zu. Eine mögliches kontextabhängiges Modell wäre eine Teilmenge von WordNet, die die Bedeutung von bestimmten Begriffen für den entsprechenden Kontext einschränkt.
Eine Komponente namens *Context-Based Free Text Interpreter (CFTI)* baut mit Hilfe des *Link Grammar Parsers* die syntaktische Struktur eines Satzes auf. Der CFTI nutzt WordNet um mögliche Bedeutungen eines Satzes anhand lexikalischer Konzepte zu erkennen. Mit Hilfe des Kontextmodells werden Mehrdeutigkeiten aufgelöst und im Idealfall eine explizite Bedeutung gefunden. Die Bedeutung eines Satzes wird in Form einer Signatur gespeichert,

[2] http://www.cmu.edu

die durch die Ontologie gegeben ist.

Die letzte zu betrachtende Facette ist die Kategorisierung eines vollständigen Dokuments. Dazu wurde der *Context-Based Categorization Agent (CCA)* entwickelt. Dieser arbeitet mit einer weiteren Ontologie, die das Kontextmodell erweitert. Mit dieser Erweiterung werden, entsprechend den Nutzervorgaben, Signaturen auf Kategorien abgebildet. Der CCA ist für die Zuweisung von Kategorien auf der Instanzebene zuständig. Abschließend wird das Ergebnis der Kategorisierung eines Dokuments in einer Datenbank gespeichert und kann durch Anwendungsprogramme, wie Suchmaschinen oder grafische Benutzeroberflächen, genutzt werden.

Cheng et al. beschreiben eine Evaluierung, welche in zwei Phasen ausgeführt wurde. In der ersten Phase wurde ein Domänenexperte gebeten, 33 unstrukturierte Dokumente aus dem Bereich der künstlichen Intelligenz fünf vorgegebenen Kategorien zuzuordnen. Dabei sollte jedes Dokument mindestens einer und maximal allen fünf Kategorien zugeordnet werden. In der zweiten Phase wurde diese Kategorisierung durch das entwickelte Framework durchgeführt. Dazu wurde der Domänenexperte von einem Wissensexperten befragt, um dessen Kenntnisse in die Erstellung des Kontextmodells einfließen zu lassen.

Die Ergebnisse der Evaluierung zeigen, dass annähernd alle Kategorien des Domänenexperten getroffen wurden, was einen Recall von fast 100% bedeutet. Allerdings wurden durch die Anwendung auch viele zusätzliche Kategorien zugeordnet, weshalb sich die Precision auf sehr niedrigem Niveau bewegt. Das zeigt, dass die prinzipielle Herangehensweise dieses Ansatzes funktioniert, jedoch weitere Verbesserungen notwendig sind um auch sehr präzise Ergebnisse liefern zu können.

Hassell et al. stellen ein System zur eindeutigen Identifikation von Entitäten auf Basis einer Ontologie vor [Hassell et al., 2006]. Ziel ist es, Autoren in Dokumenten eindeutig zu identifizieren. Dazu wurde eine Ontologie mit Namen aus dem Digital Bibliography & Library Project (DBLP) erstellt, anhand derer Autoren in DBWorld-Artikeln erkannt werden sollen. Es wird deutlich, dass die Schwierigkeit darin besteht, aus mehrfach vorkommenden, identischen Namen den korrekten Autor auszuwählen und entsprechend zu kennzeichnen. Die Disambiguierung erfolgt anhand von fünf Evidenzen, die jeweils in einer Phase des Prozesses bestimmt werden. Jede Evidenz trägt zur Berechnung einer Punktzahl bei, die den Konfidenzwert der erkannten Entität repräsentiert.

In der ersten Phase wird eine reine Volltextsuche nach Namen und abgekürzten Namen durchgeführt. Namen von Autoren sind durch die Ontologie abgebildet. Bei abgekürzten Namen wird der Vorname nur durch den ersten Buchstaben gefolgt von einem Punkt dargestellt. Diese Form befindet sich nicht explizit in der Ontologie. Die Evidenz *Name* führt zu einer Menge von Kandidaten, in der zu jedem Element der Konfidenzwert berechnet wird. Der initiale Wert berechnet sich nach der Inverse Document Frequency (IDF)-Methode, die unter anderem in [Baeza-Yates und Ribeiro-Neto, 1999] beschrieben wird. Dabei werden Kandidaten, deren Name besonders häufig in identischer Form in der Ontologie auftritt schlechter bewertet als jene, deren Name weniger häufig vorhanden ist.

Die zweite Evidenz ist die *Textnähe* zu einer bestimmten Eigenschaft, die aus der Ontologie entnommen werden kann. Zu deren Bestimmung wird innerhalb eines definierten Fensters um die Textposition eines Kandidaten nach Vorkommen der entsprechenden Eigenschaft gesucht. Wird zum Beispiel in der unmittelbaren Nähe des Namens noch die zugehörige Institution gefunden erhöht sich die Bewertung des Kandidaten. An dritter Stelle wird das reine *Textvorkommen* von Merkmalen bewertet. Im Gegensatz zur vorherigen Phase spielt die Position im Text keine Rolle. Hier werden Attribute wie zum Beispiel *bevorzugte Themen* des Autors betrachtet.

In der vierten Phase der Disambiguierung wird die *Popularität* eines Kanditaten in die Be-

rechnung einbezogen. In der Ontologie wird unter anderem die Anzahl von Publikationen eines Autors gespeichert. Anhand dieses oder ähnlicher Merkmale wird dann die *Popularität* berechnet, welche entsprechend in den Konfidenzwert einfließt.
Als letztes werden noch die *Beziehungen von Entitäten* betrachtet. Durch die Ontologie lässt sich festellen, welche Autoren häufig gemeinsam publizieren oder in Komitees zusammenarbeiten. Treten Kandidaten eines *Netzwerks* gemeinsam in einem Artikel auf, erhöht sich deren Konfidenzwert gegenüber deren, die nicht in eine Beziehung zu Anderen gebracht werden können. Letztlich wird jeweils aus Kandidaten mit identischem Namen und Vorkommen im Text derjenige mit dem höchsten Konfidenzwert ausgewählt und im Dokument über die Referenz zu seiner DBLP-Seite gekennzeichnet.

Correct Disambiguation	Found Entities	Total Entities	Precision	Recall
602	620	758	97.1%	79.4%

Tabelle 3.1: Recall und Precision aus [Hassell et al., 2006, S. 11]

Zur Evaluierung des vorgestellten Ansatzes haben die Autoren einen Textkorpus von 20 DB-World-Dokumenten manuell annotiert und anschließend als Goldstandard für einen Testlauf ihres Systems verwendet. In Tabelle 3.1 sind die Ergebnisse zusammengefasst. Es wird deutlich, dass die Entitäten für den speziellen Anwendungsfall mit sehr hoher Präzision und einem gutem Recall identifiziert wurden.

3.2.3 Informationsextraktion auf Basis strukturierter Daten

Wie sich bereits am Titel dieses Abschnitts, der dem Titel der vorliegenden Arbeit entspricht ablesen lässt, werden nun die Ansätze vorgestellt, die am stärksten mit dieser Arbeit in Beziehung stehen. Es handelt sich um die Arbeiten von Bhide et al. und Chakaravarthy et al., die jeweils dem IBM India Research Lab[3] angehören [Bhide et al., 2007; Chakaravarthy et al., 2006]. Es wird jeweils das Ziel verfolgt, Instanzen von Entitäten aus strukturierten Daten in unstrukturierten Inhalten zu identifizieren.
Bhide et al. stellen *LIPTUS* vor, ein System welches von Banken verwendet wird, um eingehende Kundenkommunikation mit den entsprechenden Kontoinformationen in Beziehung zu setzen. Desweiteren wird eine Textanalyse durchgeführt, um zum Beispiel das Anliegen des Kunden herauszufinden.
Die Identifikation eines Kunden oder eines Kontos im unstrukturierten Text einer Email wird mit sehr einfachen Mitteln erreicht. In einem ersten Schritt wird der Text von überflüssigen Informationen *bereinigt*. Dazu gehört zum Beispiel die Entfernung von Werbung oder Zitaten früherer Emails wie sie oft im Fuße einer Nachricht zu finden sind. Kunden, Konten, Kreditkarten werden im Allgemeinen über Nummern identifiziert. Diese Gegebenheit macht man sich zu Nutze und extrahiert im nächsten Schritt mit Hilfe regulärer Ausdrücke alle Vorkommen von Nummern im Text. Innerhalb festgeschriebener Fenster wird im Umfeld einer Nummer nach einer kleinen Menge von Wörtern gesucht, die einen Hinweis auf die Art der Nummer liefern (zum Beispiel *Mastercard*). Anschließend werden die gefundenen Nummern entsprechend ihres Typs in der Datenbank gesucht. Wird zu einer Nummer kein Treffer erzielt wird sie verworfen. Im Idealfall liefern alle gefundenen Nummern einen zusammenhängenden Kontext. Dies ist der Fall wenn zum Beispiel eine Kreditkartennummer gefunden wurde, die einem Konto zugeordnet ist dessen Nummer ebenfalls identifiziert wurde. Können die gefundenen Nummern nicht auf diese Art und Weise konsolidiert werden wird eine weitere Analyse des Textes vorgenommen. In den unstrukturierten Daten

[3] http://www.research.ibm.com/irl

wird nun nach Kontextinformationen gesucht, die mit den strukturierten Daten übereinstimmen. Es wird zum Beispiel überprüft, ob der Absendername mit dem Namen des Kontoinhabers übereinstimmt. Darüber hinaus gibt es eine Reihe weiterer Hinweise, die Überprüft werden. Mit jeder passenden Information, die gefunden wird erhöht sich der Konfidenzwert für das entsprechende Konto oder den Kunden. Das Konto bzw. der Kunde, der am Ende der Überprüfung den höchsten Wert erreicht, wird als passende Entität angenommen. Als Resultat wird angegeben, dass die Verarbeitung von 1,2 Millionen Emails *nur einige Stunden* in Anspruch genommen hat und 98% von ihnen korrekt zugeordnet werden konnten. Weitere Ausführungen zu den Ergebnissen lassen sich der Quelle nicht entnehmen.

Neben der Identifikation des entsprechenden Kunden oder Kontos wird versucht weitere Informationen aus dem Text zu entnehmen. Explizit im Text enthaltene Informationen lassen sich je nach Anwendungsfall durch bestimmte Muster erkennen. Eine Datenaktualisierung lässt sich zum Beispiel der Signatur einer Nachricht entnehmen. Aus dem laufenden Text können Informationen entnommen werden, die bestimmte Handlungen mit dem Kunden erfordern. Dazu gehören Ereignisse, die der Kunde nennt aber auch vorkommende Fremdprodukte, die darauf hinweisen, dass der Kunde eventuelle Wechselabsichten verfolgt. Ein wichtiger Punkt ist die Messung der Kundenzufriedenheit anhand der unstrukturierten Daten. Diese erfolgt mit maschinellem Lernen, genauer mit teilüberwachtem Lernen wie es in Abschnitt 3.1.3 beschrieben wird. Dazu wird der Algorithmus mit typischen Phrasen trainiert, die eine entsprechende Einstellungen eines Kunden wiederspiegelt.

Hierbei wird eine Quote von 80% korrekt erkannten, unzufriedenen Kunden erreicht. Weitere Aussagen darüber, wie diese Ergebnisse erreicht wurden und wie die Bewertung erfolgte, werden nicht getroffen. Dieser sehr spezifische Ansatz zeigt eine Kombination aller in Abschnitt 3.1 genannten Prinzipien. Die meisten Informationen werden hier anhand von Regeln, unterstützt von kleineren Listen, extrahiert. Dabei werden sehr gute Ergebnisse erzielt. Anhand der Erweiterung mit maschinellem Lernen lässt sich ein zusätzlicher Nutzen aus den unstrukturierten Daten ziehen.

Chakaravarthy et al. haben mit dem EROCS-System eine generische Lösung für derartige Anwendungen geschaffen. Ihr Ziel ist es, Informationen in unstrukturierten Daten strukturiert verfügbar zu machen, um somit zum Beispiel die Vorteile von Anfragemechanismen nutzen zu können. EROCS verfolgt eine entscheidene Grundannahme. Für jede Entität einer Datenstruktur gibt es eine Schablone, die mit Inhalten gefüllt wird und somit den Kontext der Entität widerspiegelt. Kann eine solche Schablone mit ausreichend Daten aus unstrukturierten Inhalten gefüllt werden, dann bedeutet dies dass die entsprechende Entität darin enthalten ist. Ein wichtiger Punkt dabei ist, dass die Entität selbst nicht im Text genannt werden muss, um identifiziert zu werden.

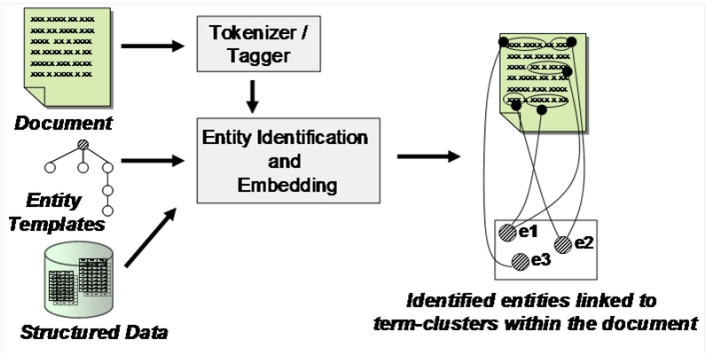


Abbildung 3.5: Systemarchitektur von EROCS aus [Chakaravarthy et al., 2006, S. 2]

Dieser Zusammenhang ist in der Systemarchitektur (Abbildung 3.5) dargestellt. Aus den strukturierten Daten werden die besagten Schablonen (*Entity Templates*) erstellt, die dann gemeinsam mit den unstrukturierten, vorverarbeiteten Dokumenten, dem System zur Verfügung gestellt werden. Die Vorverarbeitung der unstrukturierten Daten entspricht einer Filterung nach Substantiv-Phrasen, welche zum Beispiel aus einem Artikel, einem Adjektiv und einem Substantiv bestehen. Die Problembeschreibung, die hier formuliert wird ähnelt sehr der Beschreibung von Mansuri und Sarawagi. Es gilt, eine optimale Segmentierung für ein Dokument zu finden, sodass jedes Segment die am besten passende Enität kennzeichnet. Dies wird als optimale Annotation eines Dokuments bezeichent. Dabei wird jedoch die Annahme getroffen, dass jeder Satz einer Entität zugeordnet werden kann und ein Segment immer aus einem oder mehreren Sätzen besteht. Das Problem von überlappenden Segmenten wird dadurch übergangen. Innerhalb eines Segments wird für jeden Term eine Abbildung auf mögliche Entitäten vorgenommen. Die Abbildung wird mit dem leicht modifizierten TF-IDF-Prinzip bewertet. Anhand der Werte kann die optimale Annotation für das Dokument bestimmt werden.
Um die Anzahl der Datenbankanfragen zu minimieren wird ein Cache aufgebaut, der sowohl die Abbildung eines Terms auf alle möglichen Entitäten als auch die im Kontext einer Entität vorkommenden Terme enthält. Es wird eine Erweiterung vorgestellt, die es mit Dynamischer Programmierung ermöglicht, die optimale Annotation anhand der Cache-Aktualisierungen zu bestimmen.
Für die Evaluation des Systems wird ein Auszug der Internet Movie Database (IMDB) verwendet. Als unstrukturierte Daten wurden Dokumente mit Filmkritiken und -beschreibungen aus dem Internet[4] heruntergeladen. Aus diesen Dokumenten wurden die expliziten Filmnamen herausgelöscht, um die Wirksamkeit des Systems zu unterstreichen.

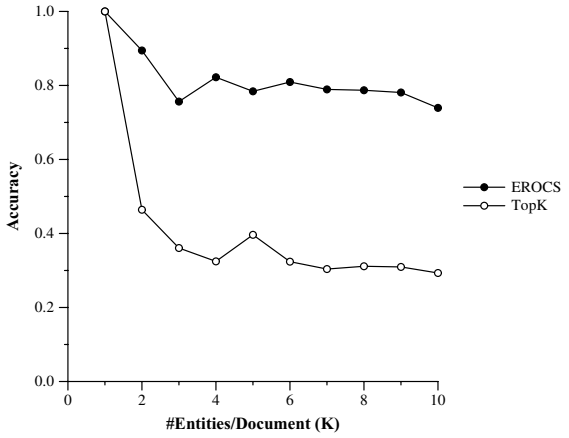


Abbildung 3.6: Ergebnisse von EROCS aus [Chakaravarthy et al., 2006, S. 8]

Die Ergebnisse der Informationsextraktion werden in Abbildung 3.6 dargestellt. Die Bewertung erfolgt in Form des F-Maßes (siehe Abschnitt 2.6). In der Abbildung ist es als *Accuracy* bezeichnet. Dieser Wert wird für verschiedene Häufigkeiten unterschiedlicher Entitäten in einem Dokument bestimmt (*#Entities/Document (K)*). Die unterschiedlichen Werte werden mit den Ergebnissen einer Top-k Suche verglichen. Über die Details der Top-k Suche finden sich allerdings keine Informationen. Die Grafik zeigt, dass beide Verfahren für Dokumente, in denen nur ein Film thematisiert wurde, ein F-Maß von 1,0 erreichen. Dies bedeutet, dass eine Entität in einem Dokument immer erkannt wird. Für zwei bis drei Entitäten pro

[4] http://www.filmsite.org

Dokument sinkt der Wert für beide Verfahren. Für die Top-k Suche wird jedoch eine rapide Verschlechterung im Vergleich zu EROCS erkennbar. Ab einer Anzahl von vier Entitäten pro Dokument konvergieren die Werte beider Verfahren. EROCS pegelt sich bei 70-80% ein während sich die Top-k Suche eher an die 30% annähert.
Die Aussage, die sich daraus schließen lässt ist, dass EROCS im Vergleich zu einer Top-k Suche bessere Ergebnisse bei der Identikation von Entitäten in Dokumenten erreicht. Da jedoch keine detaillierten Angaben zu dem Top-k Verfahren gemacht werden, ist es schwierig, dies einzuordnen. Ebenfalls ist es nicht bekannt wie sich das F-Maß zusammensetzt, also welche genauen Werte für Precision und Recall erreicht wurden.

3.3 Bewertung und Abgrenzung

Nachdem verschiedene Verfahren und Ansätze aus dem Themenbereich der Named Entity Recognition beschrieben wurden, werden diese nun nach den Gesichtspunkten der vorliegenden Arbeit bewertet. Diese Bewertung dient vor allem einer Abgrenzung der hier zu erreichenden Ziele im Vergleich zum aktuellen Stand der Technik.
Die klassischen Verfahren, die in Abschnitt 3.1 vorgestellt wurden, zeigen gute Ergebnisse bei der reinen Erkennung von Entitäten. Sie sind nicht in der Lage konkrete Instanzen zu identifizieren und zu benennen.
Die Arbeiten von Mansuri und Sarawagi sowie Chandel et al. ermöglichen die Identifikation konkreter Instanzen einer Entität, nutzen dies jedoch nur als Zwischenschritt für Aufgaben wie Datenreinigung und -vervollständigung. Bei der Identifikation zielen sie in erster Linie auf eine hohe Performanz ab. Dies führt dazu, dass sich die Ergebnisse der reinen Extraktion qualitativ kaum von Maschinenlernverfahren (Abschnitt 3.1.3) abheben, die auf die Erkennung von Namen spezialisiert sind.
Im Bereich der Informationsextraktion auf Basis von Ontologien haben Hassell et al. und Cheng et al. gezeigt, wie man strukturiertes Wissen aus Ontologien für die Verarbeitung von unstrukturierten Informationen nutzen kann. Insbesondere die Identifikation von Entitäten, entwickelt von Cheng et al., kommt den Zielen dieser Arbeit sehr nahe. Durch die starke Orientierung am Anwendungsfall ist jedoch unklar wie gut sich dieses Prinzip verallgemeinern lässt. Für den Fall, dass man es auch auf andere Anwendungsgebiete übertragen kann, entsteht ein weiteres Problem. In dieser Arbeit werden prinzipiell strukturierte Informationen aus Datenbanken verwendet, die bereits vorhanden sind. Insbesondere im kommerziellen Umfeld sind solche Daten im Gegensatz zu Ontologien oft gegeben. Um sie in einer Ontologie verfügbar zu machen entsteht zusätzlicher Aufwand der hier vermieden werden soll.
Die Nutzung von Wissen aus strukturierten Daten bei der Identifikation von Entitäten wird durch das LIPTUS-System sehr gut in einem kommerziellen Anwendungsfall dargestellt. Bhide et al. schreiben jedoch, dass aufgrund des Anwendungsfalls sehr einfache Methoden angewendet werden konnten, um das Ziel der Identifikation von Entitäten zu erreichen. Gerade die Verarbeitung von Kontonummern lässt sich nur sehr schwer in andere Anwendungsbereiche wie die Produkterkennung überführen.
Chakaravarthy et al. haben mit EROCS ein sehr mächtiges Werkzeug geschaffen, welches hervorragende Ergebnisse im Bereich der Identifikation von Entitäten liefert. Leider kommt dieses System sehr monolytisch daher und lässt sich wahrscheinlich nicht in Systeme integrieren, die für die Identifikation von Entitäten noch weiteres Kontextwissen ausnutzen. Entitäten können zwar anhand einer Schablone idenfiziert werden. Diese Schablone muss aber für jede gewünschte Entität manuell erstellt werden. Gerade im Fall von komplexen Datenstrukturen ist es wichtig, ad hoc unterschiedliche *Zielentitäten* wählen zu können. Dazu

wären eine ganze Reihe von Schablonen erforderlich, die jede erdenkliche Entität abdecken. Der Fokus der hier vorliegenden Arbeit ist, beliebige, vorhandene Datenstrukturen für die Identifikation von Entitäten in unstrukturierten Daten zu verwenden. Der Unterschied zu den Arbeiten von Hassell et al. und Chakaravarthy et al. besteht darin, dass Daten nicht verändert werden und nicht in einer andere Struktur, wie zum Beispiel eine Ontologie überführt werden. Ebenso ist der Aufwand bei der Integration der Daten auf ein Minimum beschränkt, was bedeutet, dass nur die Beschreibung der Datenstruktur erforderlich ist. Die zu erkennden Entitäten werden durch die Anwendung bestimmt und sind nicht auf bestimmte Typen eingeschränkt.
Das Wissen über strukturierte Daten wird für die Anreicherung unstrukturierter Inhalte angewendet und nicht umgekehrt. Es werden Entitäten der strukturierten in unstrukturierten Daten identifziert. Im Gegensatz zum Ansatz von Mansuri und Sarawagi werden die unstrukturierten Daten nicht für die Aktualisierung der strukturierten Daten verwendet. Die strukturierten Daten bieten Informationen sehr hoher Güte wohingegen über die unstrukturierten Daten oftmals keine Aussagen über die Qualität getroffen werden können.
Die durch den Lösungsansatz dieser Arbeit erkannten Entitäten werden mit den entsprechenden Instanzdaten verknüpft. Viele der aktuellen Lösungen sind nicht integrier- oder erweiterbar. Die hier entwickelte Lösung ist nach außen nicht abgeschlossen sondern gliedert sich leichtgewichtig in die algebraischen Prinzipien des Unstructured Information Management von SAP Research ein. Die Ergebnisse sind dementsprechend nachvollziehbar und erklärbar, sodass eine Kombination mit weiteren algebraischen Operatoren möglich ist. Dadurch kann die Qualität der Ergebnisse je nach Anwendungsfall verbessert werden, da eine Anpassung an die spezifischen Eigenschaften der gegebenen Datenstruktur möglich ist.
Zusammenfassend sind noch einmal die wichtisten Punkte in einer Übersicht aufgeführt.

Unveränderte Verwendung strukturierter Daten Die Instanzdaten und die Datenstruktur werden unverändert durch das System verwendet.

Geringer Integrationsaufwand Die strukturierten Daten werden deskriptiv an das System angebunden.

Direkte Verknüpfung von Daten Unstrukturierte Inhalte werden direkt mit strukturierten Daten verknüpft, wodurch weitere Verarbeitungsschritte auf Basis dieser Informationen möglich sind.

keine Typabhängigkeiten Jede beliebige Entität einer Datenstruktur kann für die Identifikation verwendet werden.

erweiterbarer Ansatz Die Konzepte zur Identifikation von Entitäten können aufgrund der Framework-Architektur beliebig um weitere Aspekte ergänzt werden.

Modularität Die entwickelten Komponenten sind nicht ausschließlich für die Identifikation von Entitäten in Dokumenten nutzbar. Sie gliedern sich modular in das algebraische Konzept eines Frameworks ein und sind daher beliebig mit anderen Informationsextraktionsschritten kombinierbar.

Der hier vorgestellte Ansatz geht neue Wege bei der Identifikation von Enitäten auf Basis strukturierter Daten. Einige der Konzepte stützen sich jedoch auf die Erkenntnisse dieses Kapitels. Kleine Teilaspekte der beschriebenen verwandten Arbeiten, die sich als sehr praktikabel erweisen, werden in die im folgenden Kapitel beschriebene Systemarchitektur übernommen.

Kapitel 4

Konzeption der Systemarchitektur

Ausgehend von der in Kapitel 2 vorgenommenen Anforderungsanalyse erfolgt in diesem Kapitel die Konzeption der Systemarchitektur. Als erstes werden aus den in Abschnitt 2.7 identifizierten Herausforderungen drei Kernaspekte abgeleitet. Daraus entsteht ein erster Überblick zum Aufbau der Systemarchitektur des hier entwickelten Ansatzes. Die genannten Kernaspekte bilden die Struktur für eine Aufgabenverteilung an der sich dieses Kapitel orientiert. Die einzelnen Komponenten zu den Kernaspekten werden jeweils detailliert beschrieben. Zum Schluss wird die Architektur noch einmal in einem Überblick zusammengefasst.

4.1 Identifikation von Kernaspekten

Die Kernaspekte, die der Systemarchitektur der vorliegenden Arbeit zu Grunde liegen, basieren auf den in Abschnitt 2.7 beschriebenen Herausforderungen. Es wurde dargestellt, dass ein Abgleich von unstrukturierten Daten mit einer Datenstruktur ohne die Vorverarbeitung von unstrukturierten Daten sehr aufwendig und somit nicht praktikabel ist. Jeden Eintrag einer Datenstruktur darauf hin zu überprüfen, ob er in einem Text enthalten ist, führt bei den hier gegebenen Daten (siehe Abschnitt 2.2) zu mehr als 100'000 solcher *ist-enthalten-in*-Operationen *pro* zu verarbeitendem Dokument. Um den Datenabgleich in umgekehrter Richtung durchzuführen sind zwei Schritte erforderlich. Zuerst wird der Text in einzelne Wörter zerlegt. Im zweiten Schritt wird jedes einzelne Wort des Textes in der Datenbank abgefragt.
Dabei tun sich drei Probleme auf. Erstens entsteht eine große Anzahl von Datenbankanfragen. Genauer gesagt ist sie proportional zur Größe des Textes in Wörtern. Zweitens enthält ein Text viele Wortarten wie Artikel, Präpositionen, Füllwörter und vieles mehr, die für einen Datenabgleich nicht relevant sind. Drittens besteht ein Eintrag in einer Datenbank, d.h. eine Entität, nicht immer nur aus einem einzelnen Wort. Bei den hier zu Grunde liegenden Daten handelt es sich in der Regel um kurze Phrasen. Der erste Kernaspekt, der hieraus abgeleitet wird ist:

Vorverarbeitung unstrukturierter Inhalte Die Vorverarbeitung hat das Ziel, eine Menge von Phrasen zusammenhängender Wörter zu bilden, die als Kandidaten für den Abgleich mit strukturierten Daten herangezogen werden.

Eine weitere Herausforderung bezieht sich auf die Einbindung der strukturierten Daten. Das Wissen über ihre komplexe Struktur und deren Instanzdaten wird für die Identifikation von

Entitäten in unstrukturierten Inhalten genutzt. Um einen Abgleich mit beliebigen Datenstrukturen zu ermöglichen, wird ihre Anbindung an das System über eine generische Schnittstelle realisiert. Desweiteren werden Mechanismen geschaffen, die einen performanten Zugriff auf die Datenstrukturen sicher stellen. Dies ist erforderlich da trotz der Reduzierung der unstrukturierten Inhalte sehr viele Anfragen an die strukturierten Daten notwendig sind. Der zweite Kernaspekt lautet:

Integration komplexer Datenstrukturen Die Integration hat das Ziel, eine einheitliche Zugriffsschnittstelle für beliebige Anwendungskomponenten zu schaffen und die Daten mit geeigneten Methoden performant zur Verfügung zu stellen.

Der Abgleich von unstrukturierten Inhalten mit Instanzdaten strukturierter Inhalte liefert eine Menge von Treffern, die als Kandidaten für die Verknüpfung von unstrukturierten Textsegmenten mit Entitäten in Frage kommen. Um aus dieser Kandidatenmenge den korrekten Treffer mit hoher Genauigkeit zu bestimmen, wird das Wissen über die syntaktischen Eigenschaften der unstrukturierten Phrase sowie das Strukturwissen über die strukturierten Daten an einer Stelle zusammengeführt und ausgewertet. Der dritte Kernaspekt ist:

Konsolidierung aller Informationen Das Ziel der Konsolidierung ist die Identifikation der korrekten Entität. Dies geschieht durch Zusammenführung des gewonnenen Wissens über Eigenschaften der unstrukturierten und strukturierten Daten sowie deren mögliche Verbindungen.

Es zeichnet sich ein Verarbeitungsprozess ab, der unstrukturierte Inhalte zerlegt und in kleinen Bestandteilen dieser Inhalte nach Semantiken sucht, die durch die verwendeten strukturierten Daten widergespiegelt werden. Die unstrukturierten Daten werden in eine Struktur überführt, die anfänglich auf syntaktischen Eigenschaften beruht und sukzessiv eine Abbildung aufbaut, die Texteinheiten mit Konzepten der strukturierten Daten verbindet.

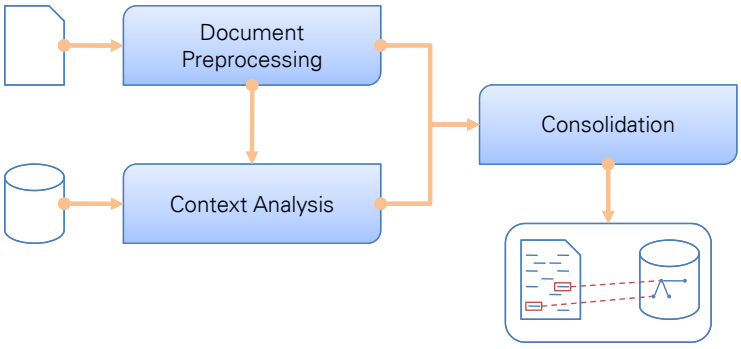


Abbildung 4.1: Überblick der Systemarchitektur

Die Systemarchitektur, die diesen Verarbeitungsprozess umsetzt, wird in Abbildung 4.1 dargestellt. Sie lässt sich in drei Bereiche einteilen, eine *Dokumentenvorverarbeitung*, eine *Kontextanalyse* und eine *Konsolidierung*. Während der *Dokumentenvorverarbeitung* wird ohne das Kontextwissen der strukturierten Daten versucht, einen unstrukturierten Text in eine Menge sinnvoller Phrasen zu zerlegen, welche anschließend mit strukturierten Daten abgeglichen werden. Die *Kontextanalyse* dient dem Abgleich der Phrasen mit den strukturierten Daten, welche zuvor in das System integriert werden. Sie liefert eine Menge von Kandidaten, die als Entitäten in Frage kommen. Die *Konsolidierung* führt die zuvor gewonnenen

Informationen zusammen und identifiziert die korrekten Entitäten aus der Kandidatenmenge. Die nächsten Abschnitte beschreiben, welche Systemkomponenten im Einzelnen für die Umsetzung der verschiedenen Bereiche erforderlich sind. Anhand des nachfolgend aufgeführten Beispieltexts, der einem typischen Forumbeitrag aus dem SDN entspricht, werden die Konzepte der einzelnen Komponenten veranschaulicht.

> *Hi all,*
> *I am a member of the SAP Netweaver Partner Program and I have a specific integration scenario of Netweaver and SAP MI.*
> *I want to setup an XI configuration for payload messages, but I don't know how to deal with that.*
> *Is there anywhere a tutorial like „Installing XI step by step" or can anybody help me with that?*
> *Cheers,*
> *Tom.*

4.2 Dokumentenvorverarbeitung

Die Darstellung des naiven Ansatzes aus Abschnitt 2.7 hat gezeigt, dass es nicht zielführend ist, Anfragen mit wahllosen Textinhalten an eine Datenstruktur zu stellen. Vielmehr muss ein Text geeignet zerlegt werden, um kleine Segmente zu erhalten, die mit strukturierten Daten abgeglichen werden können. Dies erfolgt in der Dokumentenvorverarbeitung, welche die erste Verarbeitungsphase bei der Identifikation von Entitäten in unstrukturierten Daten ist. Sie arbeitet ohne Kontextwissen und ordnet sich daher in den Bereich des Natural Language Processing (NLP), der Verarbeitung natürlichsprachiger Eigenschaften von Dokumenten ein. Baeza-Yates und Ribeiro-Neto beschreiben die Vorgehensweise für eine Dokumentenvorverarbeitung zu Zwecken der Informationsgewinnung [Baeza-Yates und Ribeiro-Neto, 1999, Kap. 7]. Einige der von ihnen vorgeschlagenen Verarbeitungsschritte werden für diese Arbeit übernommen.

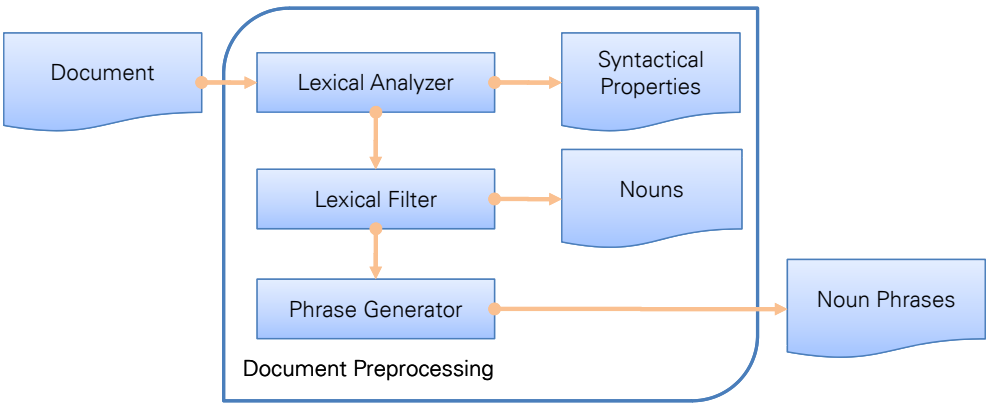


Abbildung 4.2: Architektur der Dokumentenvorverarbeitung

Die ausgewählten Verarbeitungschritte finden sich in Abbildung 4.2 wieder, welche eine detaillierte Struktur der Dokumentenvorverarbeitung darstellt. Die lexikalische Analyse extrahiert die syntaktische Struktur eines Dokuments. Anhand der syntaktischen Eigenschaften der einzelnen Wörter werden bestimmte Wortarten, in diesem Fall Nomen herausgefiltert. Aus konsekutiv auftretenden Wörtern der gefilterten Menge werden Phrasen gebildet.

Die drei Teilkomponenten, die für die Dokumentenvorverarbeitung entwickelt werden, sind im Folgenden detailliert dargestellt.

4.2.1 Lexikalische Analyse

Die erste Teilkomponente der Dokumentenvorverarbeitung ist die lexikalische Analyse. Für die Zerlegung eines Textes gibt es verschiedene Varianten. Die Einfachste ist, einen Text anhand von Leerzeichen in einzelne Wörter aufzuteilen. Diese Art der Zerlegung gibt keinerlei Aufschluss über die syntaktischen Eigenschaften eines Wortes. Da in der Dokumentenvorverarbeitung neben der Zerlegung von Inhalten auch irrelevante Wörter aufgrund ihrer syntaktischen Eigenschaften herausgefilter werden, müssen diese syntaktischen Eigenschaften von Wörtern ebenfalls erkannt werden. Cheng et al. nutzen für ihren Ansatz, der in Abschnitt 3.2.2 vorgestellt wird, den *Link Grammar Parser* von Sleator und Temperley [Cheng et al., 2003; Sleator und Temperley, 1993]. Für die vorliegende Arbeit ist der Einsatz eines solchen Parsers nicht erforderlich. Dessen Kernfunktion ist die Erkennung von Relationen zwischen Wörtern im Text.
Es wird hier lediglich die Identifikation der Wortart benötigt, da syntaktische Relationen keine Rückschlüsse auf mögliche Entitäten zulassen. Diese werden anhand der Wortart ausgewählt. Dafür reicht die Verwendung eines einfachen Part-of-Speech Taggers aus. Dieser weist jedem Wort eines beliebigen Textes seine Wortart zu. Part-of-Speech Tagger gibt es als frei verfügbare Programme, die auch über eine Schnittstelle in andere Programme integriert werden können. Diese Part-of-Speech Tagger arbeiten unterschiedlich und erzielen je nach verwendeten Daten unterschiedlich zufriedenstellende Ergebnisse in Bezug auf Performanz und Genauigkeit. Daher sollte je nach Anwendungsfall gesondert entschieden werden, welcher konkrete Part-of-Speech Tagger verwendet wird.

PRP	VB	TO	NN	DT	NN	NN	IN	NN	NN
I	want	to	setup	an	XI	configuration	for	payload	messages
CC	PRP	VB	RB	VB	WRB	TO	VB	IN	DT
but	I	do	n't	know	how	to	deal	with	that

Legende:
PRP - Personal Pronoun; VB - Verb; TO - to; NN - Noun; DT - Determiner
IN - Preposition; CC - Coordinating conjunction; RB - Adverb; WRB - Wh-adverb

Tabelle 4.1: Beispiel einer Lexikalischen Analyse

Tabelle 4.1 zeigt das Ergebnis einer Lexikalischen Analyse des Satzes *I want to setup an XI configuration for payload messages, but I don't know how to deal with that*. Der Satz stammt aus dem Beispiel, das in Abschnitt 4.1 eingeführt wurde. Die erste und dritte Zeile der Tabelle (grau hinterlegt) beinhalten jeweils die Wortart des darunter liegenden Wortes. Im Kasten unter der Tabelle werden die Symbole für die Wortarten erklärt.

4.2.2 Lexikalische Filterung

Um eine Reduzierung der Suchmenge zu erreichen, muss nach der Zerlegung eines Dokuments eine Filterung von irrelevanten Wörtern vorgenommen werden. Im Bereich der Informationsgewinnung ist es üblich, wie von Baeza-Yates und Ribeiro-Neto vorgeschlagen, so genannte Stoppwörter aus Dokumenten herauszufiltern. Zu Stoppwörtern gehören Präpositionen, Konjunktionen und Artikel. Sie enthalten keine relevanten Informationen zur Semantik eines Dokuments.

Die vorliegende Arbeit behandelt die Identifikation von Entitäten in unstrukturierten Dokumenten. Entitäten im Sinne von Datenbankentitäten werden in der Regel durch Nomen oder Nominalphrasen, d.h. Wortgruppen aus Nomen bestehend, repräsentiert. Daher ist es sinnvoller ein unstrukturiertes Dokument nicht nur von Stoppwörtern zu bereinigen, sondern ausschließlich die im Text enthaltenen Nomen zu entnehmen.

Wort	setup	XI	configuration	payload	messages
Startpunkt	139	148	151	169	177
Endpunkt	144	150	164	176	185

Tabelle 4.2: Beispiel einer Lexikalischen Filterung

Tabelle 4.2 zeigt das Ergebnis einer solchen Filterung. Es wurden nur die Wörter des Satzes aus Tabelle 4.1 übernommen, die als Nomen (*NN*) gekennzeichnet wurden. Zusätzlich wird die Start- und Endposition der Wortes im Text erfasst, da diese sich nach der Filterung nicht mehr aus dem Textfluss ergeben.

4.2.3 Phrasenbildung

Wie schon erwähnt bestehen Datenbankentitäten in der Regel nicht nur aus einzelnen Wörtern, sondern überwiegend aus Phrasen. Aus diesem Grund werden die gefilterten Wörter aus Tabelle 4.2 zu Phrasen zusammengefasst, sofern sie direkt aufeinander folgend im Dokument auftreten. Dies lässt sich aufgrund ihrer Positionsangaben bestimmen. Ein ähnliches Vorgehen wird von Chakaravarthy et al. für das in Abschnitt 3.2.3 vorgestellte System beschrieben. Dadurch entsteht eine Menge von Wörtern bzw. Phrasen, die den Eigenschaften von Instanzdaten einer Datenstruktur entsprechen.

Wort	setup	XI configuration	payload messages
Startpunkt	139	148	169
Endpunkt	144	164	185

Tabelle 4.3: Beispiel einer Phrasenbildung

Mit der Erstellung dieser Phrasen ist die Dokumentenvorverarbeitung abgeschlossen. Das Ergebnis ist ein Dokument, das in eine Menge von Nominalphrasen zerlegt wurde, wie es in Tabelle 4.3 dargestellt ist. Diese Menge wird in den weiteren Verarbeitungsphasen des Systems für die Identifikation von Entitäten verwendet.

4.3 Kontextanalyse

Die Kontextanalyse verfolgt im Wesentlichen zwei Ziele. Zum Einen werden die strukturierten Daten an das System angebunden. Zum Anderen wird über den Abgleich mit diesen Daten jeder mögliche Kontext erfasst, in dem ein Kandidat auftreten kann. Die Kandidatenmenge für den Abgleich wird durch die Dokumentenvorverarbeitung zur Verfügung gestellt.

4.3.1 Einbindung strukturierter Daten

Für einen Datenabgleich von unstrukturierten und strukturierten Daten müssen die strukturierten Datenquellen an das System angebunden werden. Diese Anbindung erfolgt unabhängig von der konkreten Datenstruktur, sodass diese je nach Anwendungsfall beliebig ausgetauscht werden kann. Dazu wird eine Abstraktionsschicht entwickelt, welche zwischen

dem System und der Datenstruktur vermittelt. Die konkrete Aufgabe besteht darin, strukturierte Daten über eine deskriptive Schnittstelle an das System anzubinden und dem System einen generischen Zugriff auf die Instanz- und Strukturdaten der Datenquelle zur Verfügung zu stellen.

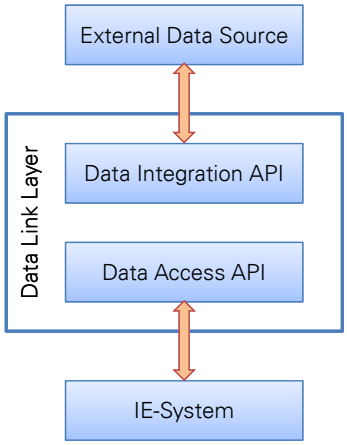


Abbildung 4.3: Aufbau der Datenanbindungsschicht

Abbildung 4.3 zeigt den Aufbau der Datenanbindungsschicht (*Data Link Layer*). Sie stellt zwei Schnittstellen zur Verfügung.

Data Integration API Die Datenintegrationsschnittstelle stellt die Verbindung zwischen der Anwendung und einer beliebigen strukturierten Datenquelle her. Das können sowohl Datenbanken als auch XML-Dateien, CSV-Dateien oder Indizes sein. Dazu wird eine Beschreibung der Datenstruktur mit allen wichtigen Eigenschaften zur Verfügung gestellt. Dies kann zum Beispiel in Form einer Text- oder XML-Datei erfolgen. Insbesondere Datenbanken wie MySQL bieten die Möglichkeit, Strukturinformationen über eine Programmschnittstelle abzufragen. In diesem Fall benötigt das System nur die Verbindungseinstellung sowie die zu verwendenen Strukturen. Weitere Eigenschaften wie Attributnamen, Datentypen, Primärschlüssel oder Fremdschlüssel werden selbstständig durch das System erfragt.

Data Access API Die Datenzugriffsschnittstelle realisiert den einheitlichen Zugriff einer Anwendung auf die Datenstruktur. Dazu werden dem Informationsextraktion (IE)-System die Strukturdaten zur Verfügung gestellt. Die Anwendung bestimmt jeweils die anzufragende Struktur und übermittelt die Anfrageparameter. Die Datenanbindungsschicht ist für die Weiterleitung der Anfrage, zum Beispiel in Form eines SQL-Befehls zuständig. Außerdem überträgt sie die Antwort der Anfrage an das System.

Bei der Implementierung dieser Schicht werden Methoden wie die Indizierung angewendet, um einen performanten Zugriff auf die Instanz- und Strukturdaten zu gewährleisten.

4.3.2 Datenabgleich

Die von der Dokumentenvorverarbeitung erzeugten Nominalphrasen weisen ähnliche Eigenschaften wie die Instanzdaten einer Datenstruktur auf, unterscheiden sich jedoch inhaltlich von ihnen. Bei einem exakten Abgleich mit den strukturierten Daten werden daher nur wenig Übereinstimmungen erzielt. Aus diesem Grund wird ein Ähnlichkeitsvergleich angestrebt. Cohen et al. beschreiben eine Reihe von Möglichkeiten für Ähnlichkeitsvergleiche

von Zeichenketten [Cohen et al., 2003]. Diese Methoden arbeiten auf Zeichenebene und eignen sich um gleiche Wörter zu erkennen, die sich durch Tipp- oder Rechtschreibfehler unterscheiden. Solche Vergleiche auf Datenbanken durchzuführen ist sehr rechenintensiv und wird für diese Arbeit nicht berücksichtigt. Die Integration dieser Funktion kann in einer späteren Erweiterung in Betracht gezogen werden.

Für die hier zu Grunde liegende Problemklasse ist es wahrscheinlicher, dass die Phrasen des unstrukturierten Dokuments nicht vollständig oder nicht in der gleichen Form durch die strukturierten Daten wiedergegeben werden. Derartige Unterschiede können durch Ähnlichkeitsmaße auf Zeichenkettenbasis nicht erkannt werden. Aus diesem Grund wird ein sehr einfach gehaltener Algorithmus verwendet, der Varianten von Phrasen generiert. Stichprobenartige Untersuchungen der vorliegenden Daten haben gezeigt, dass es ausreichend ist, reine Teilphrasen zu bilden, die keine andere Reihenfolge der Inhalte darstellen und auch keine Lücken zwischen einzelnen Wörtern aufweisen.

SAP	**Netweaver**	**Partner**	**Program**
SAP			
	Netweaver		
		Partner	
			Program
SAP	Netweaver		
	Netweaver	Partner	
		Partner	Program
SAP	Netweaver	Partner	
	Netweaver	Partner	Program
SAP	Netweaver	Partner	Program

Tabelle 4.4: Beispiel der Variantenbildung

Tabelle 4.4 zeigt alle resultierenden Varianten der Phrase *SAP Netweaver Partner Program*, die auch dem Beispiel aus Abschnitt 4.1 entnommen ist.

Bei einem Datenabgleich werden die Varianten der Phrasen aus dem unstrukturierten Dokument in der Datenstruktur über die im vorherigen Abschnitt beschriebene Schnittstelle abgefragt. Anhand einer Konfiguration wird der Komponente für den Datenabgleich bekannt gegeben, welche Teile, zum Beispiel welche Tabellen der Datenstruktur jeweils angefragt werden. Damit erfolgt eine erste Verbindung zwischen unstrukturierten und strukturierten Daten.

Eine wichtige Aufgabe während des Datenabgleichs ist die Auflösung von Fremdschlüsselbeziehungen. Abbildung 2.4 auf Seite 9 zeigt, dass eine SAP-Komponente anhand eines bestimmten Terms identifiziert werden kann. Ein Term wiederum kann in verschiedenen Varianten, zum Beispiel in Form einer Abkürzung, auftreten. SAP-Komponenten stellen so gesehen die *Zielentität* dar, die hier erkannt werden soll. Aus diesem Grund wird die Verbindung einer Abkürzung über dessen zugeordneten Term bis hin zur Komponente vollständig aufgelöst. Die Strukturinformationen der Datenquelle, die über die Datenanbindungsschicht angefragt werden können, werden verwendet um weitere Abfragen durchzuführen. Diese führen über Fremdschlüsselbeziehungen zu dem gewünschten Ergebnis. Eine derartige Vorgehensweise lässt sich auch auf andere Datenquellen übertragen. Fremdschlüsselbeziehungen sind die Basis strukturierter Daten, wie zum Beispiel relationaler Datenbanken.

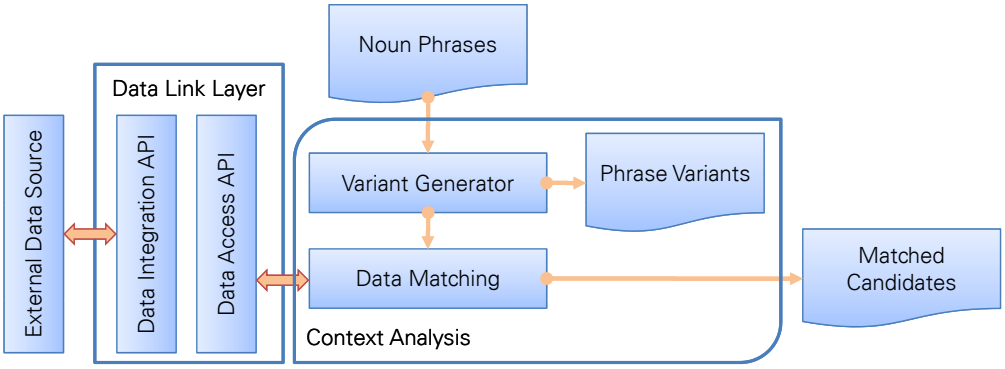


Abbildung 4.4: Architektur der Kontextanalyse

4.3.3 Überblick

Abbildung 4.4 zeigt noch einmal die entwickelten Komponenten der Kontextanalyse sowie deren Schnittstellen. Im Ergebnis der Kontextanalyse wird jeder Kandidat aus der Menge der Phrasenvarianten übernommen, der einen oder mehrere Treffer in der Datenstruktur verzeichnen kann. Darüber hinaus wird gespeichert, welche Art von Information er enthält (Komponentenname, Abkürzung, Term, ...) und welcher *Zielentität* diese Information zugeordnet werden kann.

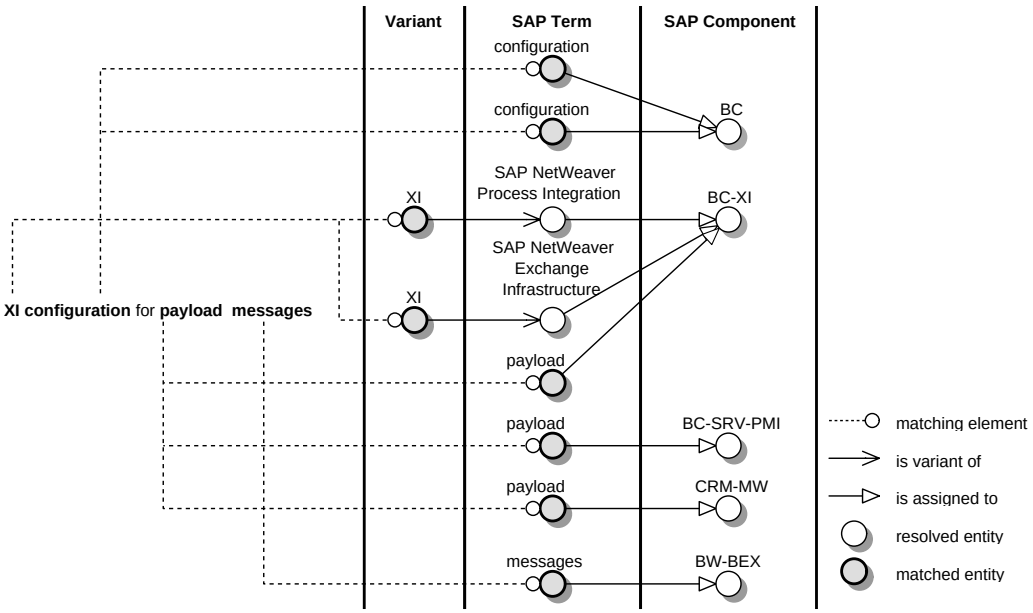


Abbildung 4.5: Ergebnis der Kontextanalyse

Abbildung 4.5 zeigt dieses Ergebnis am Beispiel des Textausschnitts *XI configuration for payload messages*. Aufgrund der Variantenbildung ist es möglich, dass eine Phrase zu mehreren Treffern führt, die unterschiedlichen Kontexten entstammen. Die Teilphrasen *configure*, *messages* und *payload* werden jeweils ein oder mehrmals als Entsprechung für einen Term gefunden. *XI* wird zweimal als Variante eines Terms, genauer gesagt als Abkürzung für einen solchen, gefunden. Durch Auflösung der Fremdschlüsselbeziehungen lassen sich daraus sechs verschiedene Kontexte, im Sinne von Komponenten für diese Phrase ableiten. An dieser Stelle werden die Ergebnisse ohne eine Wertung als Kandidaten abgespeichert. Die

Auswahl der korrekten Zuordnung wird während der weiteren Verarbeitung vorgenommen.

4.4 Konsolidierung

Die Konsolidierung führt alle bisher gewonnenen Informationen zusammen und wählt aus der Menge der möglichen Zuordnungen zwischen unstrukturierten und strukturierten Daten jeweils für eine Phrase die plausibelste Entität aus. Dabei stehen neben den Informationen, die durch die Kontextanalyse zu jedem Kandidaten gespeichert wurden, sowohl die Eigenschaften der unstrukturierten Daten als auch die Strukturinformationen der externen Datenquellen zur Verfügung.
Nachdem die Dokumentenvorverarbeitung und die Kontextanalyse wichtige Voraussetzungen, wie die Reduzierung der Suchmenge und Bestimmung von Kandidaten von Entitäten in unstrukturierten Inhalten zur Verfügung gestellt haben, erfolgt durch die Konsolidierungskomponente die eigentliche Identifikation der korrekten Entitäten.
Da das Problem darin besteht, aus einer Menge von Treffern den Korrekten auszuwählen, werden diese geeignet bewertet. Auf Grundlage der Bewertung erfolgt die Aufstellung einer Rangfolge, aus der der beste Treffer ausgewählt wird. In den verwandten Arbeiten, wie zum Beispiel aus den Abschnitten 3.2.1 und 3.2.3, wird dies deshalb als Rangordnungs-Problem bezeichnet.
In den genannten Arbeiten wird dieses Problem durch stochastische Verfahren gelöst, wie es auch bei anderen Systemen zu beobachten ist. Ein prominenter Vertreter dieser Verfahren ist TF-IDF. Es dient im ursprünglichen Sinn für die Gewichtung eines Terms in einem Dokument. TF-IDF bringt zum Ausdruck, wie stark ein Term den Inhalt eines Dokuments wiederspiegelt. Der Term entstammt in diesem Fall in der Regel aus einer Menge von Termen innerhalb einer Suchanfrage. Es ist ohne Weiteres auch für die Gewichtung eines Terms in einer Datenbank nutzbar. Die Suchanfrage ist in dem Fall die Phrase aus dem unstrukturierten Inhalt. Das Dokument spiegelt den Datensatz wieder, in dem die Phrase gefunden wurde. Die Bezeichnung *Dokumentenkorpus* entspricht daher im Folgenden der *Datenbank* und das *Dokument* ist als *Datenbankeintrag* bzw. *Datensatz* aufzufassen.
TF-IDF (Gleichung 4.3) beinhaltet zwei Faktoren. Term Frequency (TF) ist die Intra-Dokument-Charakterisierung, ausgedrückt durch die Häufigkeit des Terms t im Datensatz d im Vergleich zur Maximalhäufigkeit über alle Terme t_i.

$$TF_{t,d} = \frac{|t(d)|}{max(|t_i(d)|)} \tag{4.1}$$

Die Inverse Document Frequency (IDF) beschreibt die Inter-Dokument-Charakterisierung, ausgedrückt durch den Logarithmus über die Anzahl N aller Datensätze in der Datenbank geteilt durch die Anzahl der Datensätze n_t, in denen t vorkommt.

$$IDF_t = \log \frac{N}{n_t} \tag{4.2}$$

$$TF_{t,d}IDF_t = \frac{|t(d)|}{max(|t_i(d)|)} \cdot \log \frac{N}{n_t} \tag{4.3}$$

Für die Implementierung einer TF-IDF basierten Rangordnung für diese Arbeit ergeben sich verschiedene Probleme.

Kein Einfluss durch Strukturkenntnis Bei TF-IDF handelt es sich um ein Verfahren, das die strukturierten Daten anhand von Statistiken der Instanzdaten bewertet. In dieser Ar-

beit fließt aber insbesondere das Wissen über die Struktur der Daten in die Bewertung mit ein.

Keine Eignung für exakten Vergleich Das Verfahren ist vorwiegend für eine *ist-enthalten-in*-Beziehung zwischen unstrukturierten und strukturierten Daten ausgelegt. Hier wird jedoch ein exakter Abgleich zwischen einem Teil der unstrukturierten Daten und einem Datenbankeintrag durchgeführt. Die TF-Komponente ist in diesem Fall bedeutungslos, da sie immer 1 ergibt. Die Bewertung ist somit nur abhängig von IDF.

Keine Auflösung von Mehrdeutigkeiten Die IDF-Komponente bewertet mehrdeutige Terme in jedem Fall schlechter als Eindeutige. Dies ist prinzipiell erwünscht, jedoch ist eine Auflösung von Mehrdeutigkeiten dabei nicht möglich.

Keine Unterscheidung gleicher Terme Mit dem vorherigen Punkt geht einher, dass keine Unterscheidung syntaktisch gleicher Terme möglich ist, obwohl sich verschiedene Semantiken in ihnen widerspiegeln. Das Verfahren ist eher dazu geeignet eine Rangfolge von Entitäten über ein gesamtes Dokument zu bestimmen. Für die reine Identifikation von Entitäten innerhalb einer Phrase ist es nicht zweckdienlich.

Aus den genannten Nachteilen werden im Folgenden einige Regeln abgeleitet, die für die Konsolidierung zum Einsatz kommen. Sie berücksichtigen sowohl die Fragestellung wie gut ein Kandidat die unstrukturierten Daten wiederspiegelt, als auch wie sehr das Vertrauen in den Kandidaten aufgrund der Eigenschaften der strukturierten Daten ist. Diese Regeln werden zu einem Konsolidierungsalgorithmus zusammengefasst, der die Auswertung der bis dahin gesammelten Informationen steuert.

4.4.1 Längste Übereinstimmung

In Abschnitt 4.3.2 wurde beschrieben, wie aus einer, dem unstrukturierten Dokument entnommenen Phrase verschiedene Varianten gebildet werden. Für diese Varianten werden mit Hilfe der Kontextanalyse Kandidaten für entsprechende Entitäten gefunden. An dieser Stelle werden die verschiedenen Varianten wieder zusammengeführt. Dabei werden die Treffer ausgewählt, die die längste Übereinstimmung (in Anzahl der Wörter) mit der ursprünglichen Phrase aus der Dokumentenvorverarbeitung aufweisen. Die erste Konsolidierungsregel lautet:

> ***Je Phrase werden nur die Kandidaten in die Ergebnismenge übernommen, die die längste Übereinstimmung mit der ursprünglichen Phrase aufweisen.***

Abbildung 4.6 zeigt die Kandidatenmenge für die Beispielphrase *SAP MI*. Aus dieser Phrase wurden drei Varianten gebildet, die in den unterschiedlichen Datenbanktabellen Treffer erzielen. Insgesamt können dieser Phrase neun verschiedene *Zielentitäten* zugewiesen werden. Die Abbildung zeigt außerdem wie die Regel der längsten Übereinstimmung zum Tragen kommt. Die Variante *SAP MI* stellt in dem Fall die längste Übereinstimmung mit der Ursprungsphrase dar, da sie dieser selbst entspricht. Für die weitere Verarbeitung werden nur die Treffer in die Zielmenge übernommen, die durch diese Variante erzielt wurden. Alle Weiteren werden verworfen. In diesem konkreten Beispiel ist die Identifikation der korrekten Entität damit bereits abgeschlossen, da nur ein Kandidat übrig bleibt. Im Folgenden werden weitere Beispiele gezeigt in denen es einer weiteren Verarbeitung bedarf, da auch nach Anwendung dieser Regel noch mehrere Kandidaten in der Ergebnismenge vorhanden sein können.

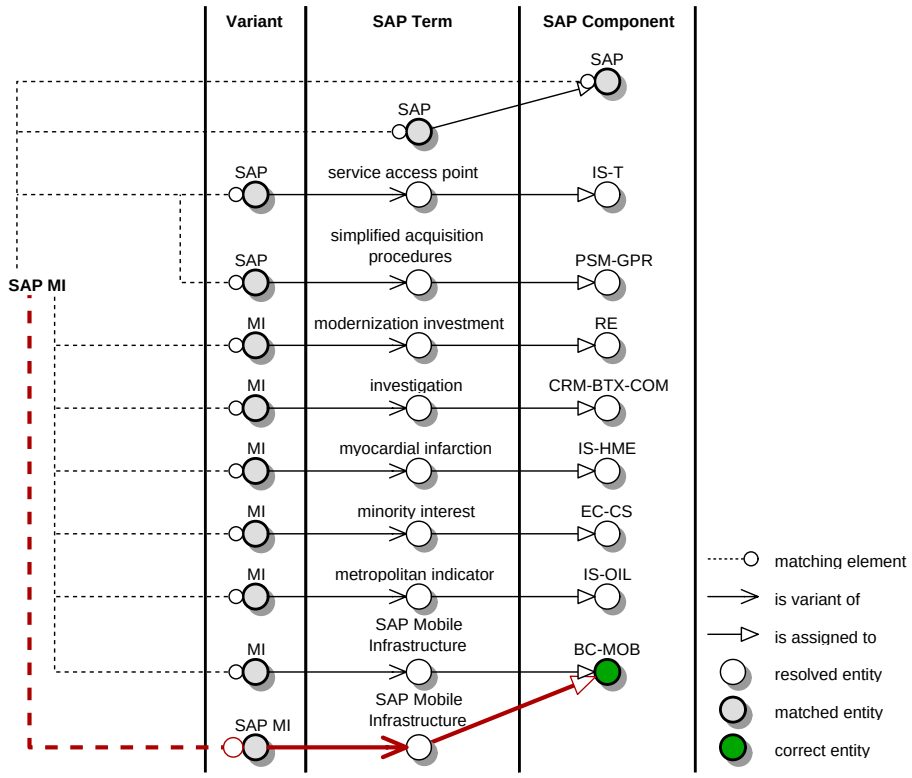


Abbildung 4.6: Ergebnis der längsten Übereinstimmung

4.4.2 Bester Datenbanktreffer

Mit der Regel der längsten Übereinstimmung werden Charakteristiken der unstrukturierten Daten genutzt, um irrelevante Treffer aus der Ergebnismenge zu entfernen. Es gibt nach wie vor die Möglichkeit, dass nach diesem Schritt weitere Kandidaten verbleiben, die zu unterschiedlichen *Zielentitäten* aufgelöst werden. Es wird daher eine weitere Regel eingeführt, die die Eigenschaften der strukturierten Daten berücksichtigt. Jeder Teil der Datenstruktur, zum Beispiel Datenbanktabellen, wird bei der Einbindung in das System mit einem Vertrauensfaktor versehen, der Auskunft darüber gibt, wie wertvoll ein Treffer in dieser konkreten Struktur ist. Die zweite Konsolidierungsregel lautet:

> ***Jedem Kandidaten der Ergebnismenge wird der Vertrauensfaktor der Datenstruktur zugewiesen in der sein Treffer erzielt wurde. Die Kandidaten mit dem höchsten Wert verbleiben in der Ergebnismenge, alle restlichen werden entfernt.***

In Abbildung 4.7 ist das Beispiel der Phrase *SAP NetWeaver Partner Program* zu sehen. Auf die Gesamtmenge der Kandidaten wurde bereits die Regel der längsten Übereinstimmung angewendet. Dadurch werden alle Kandidaten der Varianten *SAP* und *NetWeaver* aus der Ergebnismenge entfernt. *SAP NetWeaver* bietet die längste Übereinstimmung der Phrase aus den unstrukturierten Daten mit Einträgen in der Datenstruktur. Mit dieser Variante werden jedoch noch sechs Treffer in der Datenstruktur erzielt, weshalb hier die Regel des besten Datenbanktreffers angewendet wird. Fünf Kandidaten erhalten den Vertrauensfaktor eines *SAP-Terms* und einer erhält den einer *SAP-Komponente*. Die Vertrauensfaktoren der Datenstrukturen sind so gewählt, dass ein direkter Treffer auf eine *SAP-Komponente*

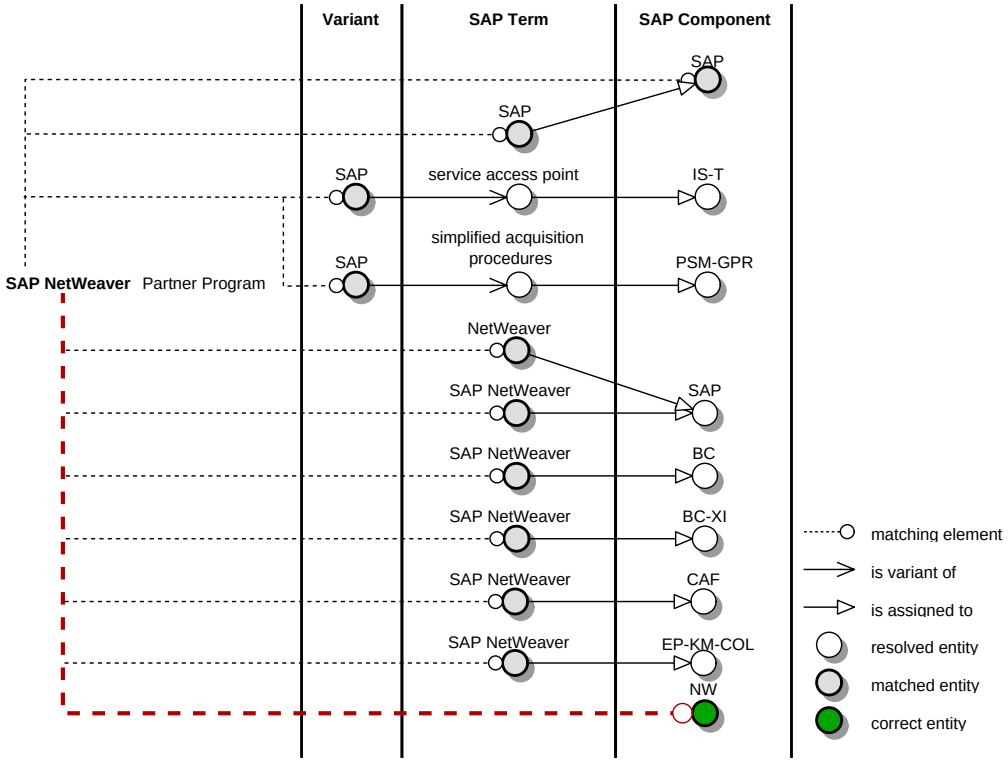


Abbildung 4.7: Ergebnis des besten Datenbanktreffers

höher bewertet wird, als ein Treffer auf einen *SAP-Term*. Der Kandidat, der die Teilphrase *SAP NetWeaver* der Komponente *NW* zuweist wird als korrekt ausgewählt. Mit dieser Regel werden vor allem Mehrdeutigkeiten aufgelöst. Begriffen, die in gleicher Form in unterschiedlichen Teilen der Datenstruktur vorkommen werden damit Wertigkeiten verliehen, welche es ermöglichen irrelevante Treffer bei der Suche nach der am Besten bewerteten Entität herauszufiltern.

4.4.3 Geringste Streuung

Auch wenn die Ergebnismenge mit der Anwendung der zuvor beschriebenen Regeln in einer Vielzahl der Fälle bereits auf einen Kandidaten je Phrase reduziert wird, so gibt es immer noch die Möglichkeit, dass Varianten gleicher Länge mit einem Treffer in der gleichen Datenstruktur auf mehrere verschiedene Entitäten zurückgeführt werden können. Dies ist insbesondere bei Phrasen der Fall, in denen nur Varianten einen Treffer erzielen, die aus einem Wort bestehen bzw. wenn die gesamte Phrase nur ein einzelnes Wort ist. Unter diesen Umständen führen die ersten zwei Konsolidierungsregeln zu keiner signifikanten Minimierung der Ergebnismenge.
Die Regel der geringsten Streuung sorgt dafür, dass eine Ergebnismenge mit einer hohen Mehrdeutigkeit zumindest auf Varianten reduziert wird, deren Kanditaten mehrmals zu der gleichen *Zielentität* aufgelöst werden können. Eine geeignete Methode zur Berechnung der Streuung einer Variante ist die Bestimmung ihrer Entropie. Die Entropie wird im eigentlichen Sinne genutzt, um den Informationsgehalt einer Quelle pro Zeichen zu beschreiben. Je mehr unterschiedliche Worte zum Beispiel in einem Satz vorhanden sind, desto höher ist sein Informationsgehalt pro Wort, seine Entropie. Besteht ein Satz nur aus der Aufreihung von zehn mal demselben Wort, so ist sein Informationsgehalt pro Wort sehr gering, da die

Information mit nur einem Wort dargestellt werden kann. Dieser Sachverhalt wird auf die unterschiedlichen Treffer pro Variante einer Phrase übertragen.
In Gleichung 4.4 wird die Berechnung der Entropie H für eine Variante X dargestellt. Über alle enthaltenen Zielentitäten $z \in Z$ wird die Summe ihrer bedingten Auftrittswahrscheinlichkeit P berechnet. $P(X = z)$ ist die Wahrscheinlichkeit, dass die Variante X der Entität z entspricht. Durch den Logarithmus wird eine Normalverteilung der Werte erreicht, wodurch unterschiedlich große Mengen eindeutig miteinander verglichen werden können.

$$H(X) = -\sum_{z \in Z} P(X = z) \cdot \log_2(P(X = z)) \tag{4.4}$$

Am Ergebnis kann man die Streuung der unterschiedlichen *Zielentitäten* ablesen. Je kleiner der Wert, also je geringer der Informationsgehalt, desto geringer ist die Streuung, da weniger unterschiedliche *Zielentitäten* enthalten sind. Die dritte Konsolidierungsregel lautet demnach:

> ***Pro Phrase wird für jede Variante die Entropie über die Treffer ihrer Kandidaten berechnet. Es werden nur die Varianten übernommen, die einen bestimmten Schwellwert der Entropie nicht überschreiten. Die verbleibenen Kandidaten werden aus der Ergebnismenge entfernt.***

Der Schwellwert kann je nach Benutzervorgabe angepasst werden. Dies hat die Auswirkung, dass Varianten bis zu einer bestimmten Streuung immer noch als korrekt aufgefasst werden. Wird dieser Wert besonders hoch eingestellt, so sollte der resultierende Algorithmus (Abschnitt 4.4.4) angepasst werden. Dieser sollte dann nicht alle Treffer übernehmen, die durch die Variante erreicht wurden sondern nur diejenigen mit der höchsten Auftrittswahrscheinlichkeit. Für die vorliegende Arbeit wird der Wert so gewählt, dass nur Varianten zugelassen werden, die überhaupt keine Streuuung verursachen, sondern deren Treffer auf ein und dieselbe Entität zurückzuführen sind. In diesem Fall ist die Entropie $H(X) = 0$.

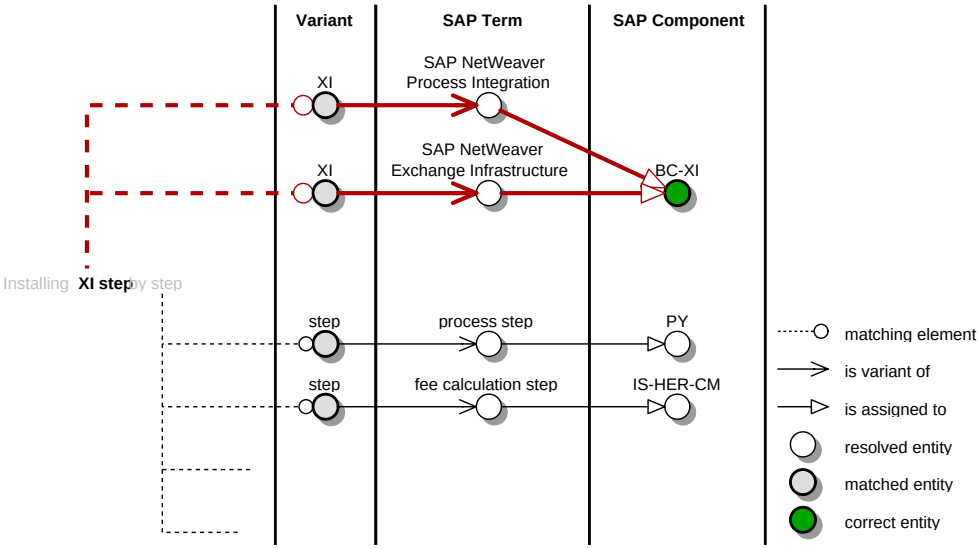


Abbildung 4.8: Ergebnis der geringsten Streuung

In Abbildung 4.8 wird das Beispiel des Textsegments *Installing XI step by step* dargestellt. *XI step* und *step* wurden als Nominalphrase aus dem Text entnommen. Da jeweils nur für die Teilphrasen *XI* und *step* Treffer in den strukturierten Daten erzielen, wird mit der

Regel der längsten Übereinstimmung keine Reduzierung der Ergebnismenge erreicht. Die Anwendung der zweiten Regel (bester Datenbanktreffer) wird durch die ins Leere laufenden, gestrichelten Linien angedeutet. Die Entropie der Variante $X = XI$ berechnet sich wie folgt. Z ist nur einelementig mit $z_1 = BC - XI$. $P(X = z_1)$ ergibt daher 1.

$$H(X) = -(1 \cdot \log_2 1) = 0 \tag{4.5}$$

Für die Variante $X = step$ ergeben sich folgende Werte. Z besteht aus $z_1 = PY$ mit $P(X = z_1) = 0,5$ und $z_2 = IS - HER - CM$ mit $P(X = z_2) = 0,5$.

$$H(X) = -2 \cdot (0,5 \cdot \log_2 0,5) = -2 \cdot (0,5 \cdot -1) = 1 \tag{4.6}$$

Entsprechend der Regel verbleiben nur die Kandidaten in der Ergebnismenge, die zur Entität *BC-XI* aufgelöst werden können. Dies führt dazu, dass für diese Variante die Aussage getroffen werden kann, dass sie über mehrere Pfade durch die strukturierten Daten zu einer Entität führt. Die Kandidaten, die mit *step* erzielt wurden, werden verworfen. Dabei wird deutlich, dass mitunter alle Kandidaten einer Phrase verworfen werden. In dem Beispiel bleiben für die erste Phrase noch die Treffer von *XI* erhalten. In der zweiten Phrase, die nur aus *step* besteht, werden alle Kandidaten verworfen. Das ist erwünscht, da innerhalb dieser Phrase keine Entität eindeutig, im Sinne der hier aufgestellten Regeln identifiziert werden konnte.

4.4.4 Überblick

Abbildung 4.9 stellt noch einmal einen Überblick zur Konsolidierung dar.

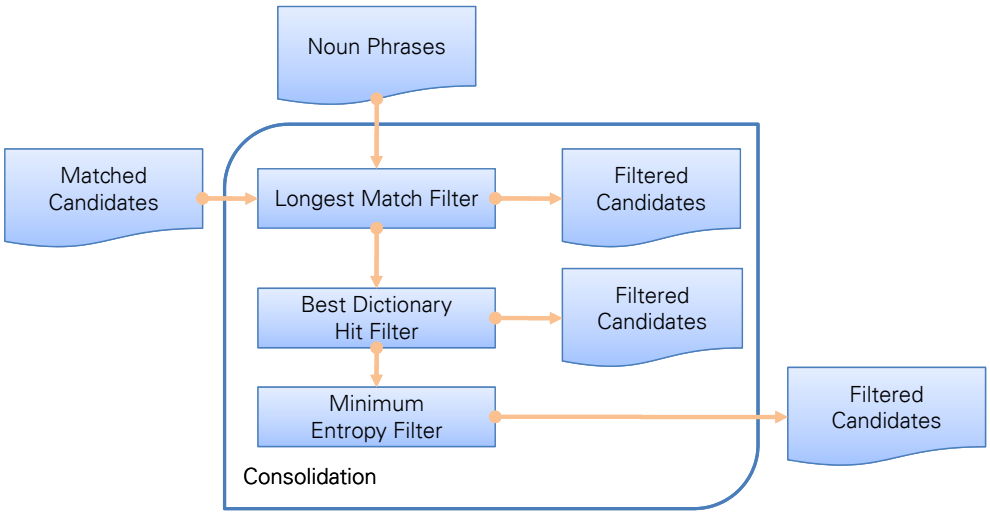


Abbildung 4.9: Architektur der Konsolidierung

Die bisher in diesem Abschnitt beschriebenen Konsolidierungsregeln werden nun noch einmal zusammenfassend in einem Konsolidierungsalgorithmus beschrieben. Der Algorithmus arbeitet prinzipiell auf Ebene einer Nominalphrase, weshalb keine Kenntnisse über angrenzende Inhalte des unstrukturierten Dokuments vorhanden sind. Als Erweiterung dieser Arbeit ist die Ausweitung der Konsolidierung über Phrasengrenzen hinweg denkbar. Zur Umsetzung der hier geforderten Zielstellung hat sich die soeben beschriebene Vorgehensweise jedoch als vielversprechender erwiesen.

Algorithmus 1 Konsolidierung

```
for each noun group do
  if result set > 1 then
    choose longest matched variant
  end if
  if result set > 1 then
    for each candidate do
      assign trust of hit dictionary
    end for
    choose highest ranked candidate
  end if
  if result set > 1 then
    for each variant do
      calculate entropy
      if entropy > 0 then
        remove variant from result set
      end if
    end for
  end if
end for
```

4.5 Zusammenfassung

In diesem Kapitel wurden die Konzepte der Systemarchitektur umfassend beschrieben. Ausgehend von den Herausforderungen, die in der Anforderungsanalyse in Kapitel 2 geschildert sind, wurden hier als erstes die Kernaspekte für die Entwicklung eines Systems zur Identifikation von Entitäten in unstrukturierten Daten ermittelt. Dies sind im einzelnen:

- Vorverarbeitung unstrukturierter Inhalte
- Integration komplexer Datenstrukturen
- Identifikation der korrekten Entitäten

Der Gesamtprozess wurde, wie in Abbildung 4.10 auf der nächsten Seite dargestellt, in drei Phasen eingeteilt, die jeweils durch entsprechende Systemkomponenten umgesetzt werden. Die Dokumentenvorverarbeitung reduziert als erstes die unstrukturierten Inhalte auf ihre signifikanten Bestandteile. Dies sind in diesem Fall Nominalphrasen. In der Kontextanalyse werden unterschiedlichen Varianten der Phrasen erstellt, um einen Ähnlichkeitsvergleich zu ermöglichen. Diese Varianten werden in den strukturierten Daten abgefragt und anhand von Fremdschlüsselbeziehungen zu Entitäten aufgelöst, welche fortan als Kandidaten weiterverarbeitet werden.
Um dies zu ermöglichen wurde eine Datenintegrationsschicht entwickelt, welche die generische Integration beliebiger strukturierter Datenquellen ermöglicht und den Informationsaustausch mit der Datenquelle und dem System sicherstellt. Die Kandidatenmenge wird letztendlich anhand von Konsolidierungsregeln in eine Ergebnismenge überführt, die die korrekte Zuordnung von Entitäten zu Segmenten, in dem Fall Phrasen, unstrukturierter Dokumente enthält.
Die Beschreibung der detaillierten Implementierung der hier beschriebenen Konzepte sowie die Auswertung der Ergebnisse werden in den nächsten Kapiteln vorgenommen.

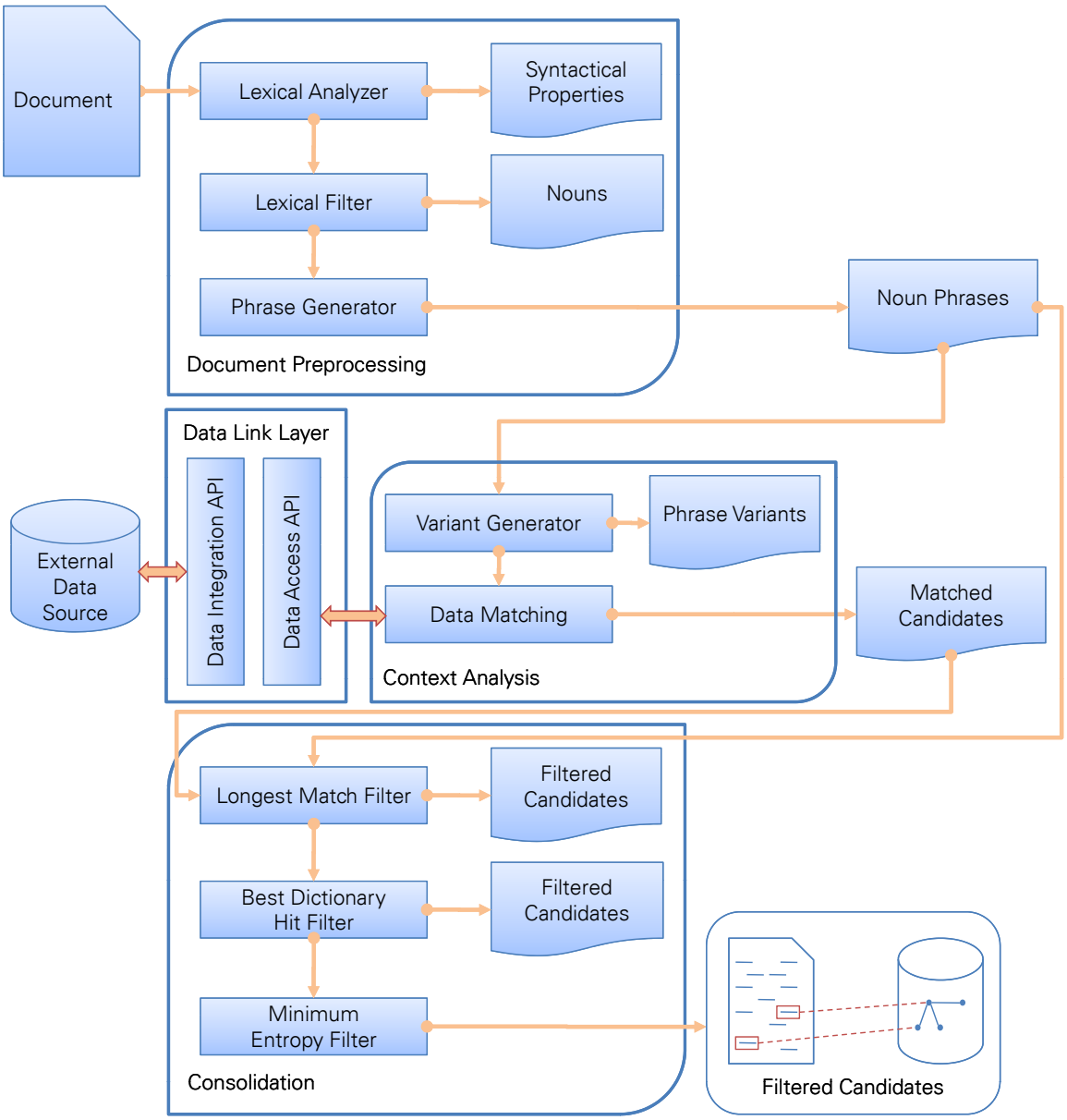


Abbildung 4.10: Gesamtarchitektur

Kapitel 5

Beschreibung der Implementierung

In Kapitel 4 wurde die Konzeption einer Systemumgebung für die Umsetzung der Anforderungen aus Kapitel 2 vorgenommen. In diesem Kapitel wird nun die prototypische Implementierung der konzipierten Systemkomponenten dargestellt. Dazu wird als erstes ein kurzer Überblick über die Systemumgebung gegeben, in der die Implementierung erfolgt. Anschließend wird analysiert, wie sich die konzeptionell entwickelten Komponenten in das SAP Research UIM-Framework integrieren lassen. Daraus geht unter anderem hervor, dass konzeptionelle Erweiterungen am Framework selbst vorgenommen werden müssen. Die Umsetzung dieser Erweiterungen sowie die konkrete Implementierung der unterschiedlichen Systemkomponenten werden detailliert erläutert. Abschließend werden die wichtigsten Punkte noch einmal zusammengefasst.

5.1 Systemumgebung

Die Implementierung erfolgt auf der Sun Java 6 Plattform, welche als Laufzeitumgebung eine Java Virtual Machine verwendet. Als Datenbanksystem dient der MySQL Server von Sun Microsystems in der Version 5.0. Die Eclipse-Distribution Europa (Version 3.3) wurde als Entwicklungsumgebung ausgewählt.

5.2 Einordnung in das SAP Research UIM-Framework

Die im vorherigen Kapitel vorgestellte Konzeption einer Systemarchitektur für die Identifikation von Entitäten in unstrukturierten Daten erfolgte vordergründig ohne Bezug auf eine vorgegebene Umgebung. Die Anforderungsanalyse macht deutlich, dass die Implementierung als Erweiterung des SAP Research UIM-Frameworks (siehe Abschnitt 2.3) realisiert wird. Daher wird die Konzeption aus Kapitel 4 in diesem Abschnitt auf das Framework übertragen.
Im Wesentlichen wird eine Verschmelzung der Architekturkonzepte vorgenommen, die in den Abbildungen 2.5 auf Seite 11 und 4.10 auf der vorherigen Seite zu finden sind. Es wird dargestellt in welcher Form die einzelnen Teilkomponenten der Systemkonzeption in das Framework integriert werden.
Die folgenden Teilkomponenten mit ihren entsprechenden Aufgaben wurden für die Systemarchitektur der vorliegenden Arbeit konzipiert:

Lexical Analyzer - Identifikation von Worttypen aus unstrukturierten Inhalten

Lexical Filter - Filter für bestimmte Worttypen

Phrase Generator - Phrasenbildung anhand der Position im Text

Data Integration - Einbindung von externen strukturierten Daten

Data Access - Zugriffsoperationen auf externe strukturierte Daten

Variant Generator - Bildung von Varianten

Data Matching - Abgleich von unstrukturierten und strukturierten Daten

Longest Match Filter - Filterung der längsten Übereinstimmung aus Ergebnisdaten

Best Dictionary Hit Filter - Filterung des besten Datenbanktreffers

Minimum Entropy Filter - Filterung nach minimaler Streuung (Entropie)

Das Framework ist so aufgebaut, dass sich zusätzliche Operatoren entsprechend ihres Typs ohne Eingriff in die Architektur integrieren lassen. Die unterschiedlichen Typen von Operatoren wurden in Abschnitt 2.3 beschrieben. Für die Komponenten, die sich nicht als Typ eines Operators klassifizieren lassen, müssen Erweiterungen am Framework vorgenommen werden. Im Folgenden werden sowohl Komponenten identifiziert, die sich als Operator integrieren lassen, als auch solche, die eine Erweiterung der Konzepte des Frameworks erfordern.
Mit Ausnahme von *Data Integration* und *Data Access* lassen sich alle Komponenten als Basisoperatoren (siehe Abschnitt 2.3) umsetzen bzw. in einen solchen Operator integrieren. Basisoperatoren bieten die Funktion, Eigenschaften von unstrukturierten Inhalten zu extrahieren. Anhand einer gegebenen Eingabe (Annotation) wird eine Menge von Annotationen erzeugt, die das Ergebnis des Operators darstellt.
Vor der Implementierung dieser Basisoperatoren werden die genannten Ausnahmen betrachtet. Diese Komponenten nehmen eine spezielle Rolle ein. Sie stellen die Einbindung von und den Zugriff auf externe Datenstrukturen sicher. Das Framework bietet in der Datenintegrationsschicht bisher nur Importkomponenten für unstrukturierte Daten. Für die Einbindung strukturierter Daten gibt es im Framework bisher noch keine Vorkehrungen. Daher wird im Abschnitt 5.3 zunächst eine Erweiterung der Datenintegrationsschicht des Frameworks beschrieben. Diese wird anschließend von den spezifischen Operatoren zur Identifikation von Entitäten genutzt.
In Abbildung 5.1 wird die erweiterte Architektur des Frameworks dargestellt. Die Erweiterungen sind im Bild rot markiert. Der obere Teil des Bildes zeigt die erweiterte Datenintegrationsschicht, welche nun beliebige strukturierte Daten für die Verarbeitung zur Verfügung stellt. In der Ausführungsschicht (mittig in der Abbildung) entstehen die beiden neuen Operatoren (*Lexical Analyser* und *Entity Identifier*), welche in Abschnitt 5.4 beschrieben werden. Die Deskriptoren werden um einen neuen Typ erweitert, der für die Beschreibung von strukturierten Datenquellen zuständig ist. Die genaue Beschreibung dessen ist Teil des folgenden Abschnitts.

5.3 Implementierung der Datenanbindung

Wie bereits erwähnt, existiert für das Framework bisher keine Komponente zur Einbindung strukturierter Daten. Dies ist jedoch essentiell für die Implementierung eines Operators zur Identifikation von Entitäten in unstrukturierten Inhalten. Die Konzepte der Datenanbindungsschicht aus Abschnitt 4.3.1 müssen demnach auf die Datenintegrationsschicht des Frameworks übertragen werden. Diese wird um einige Schnittstellen erweitert, die jeweils für

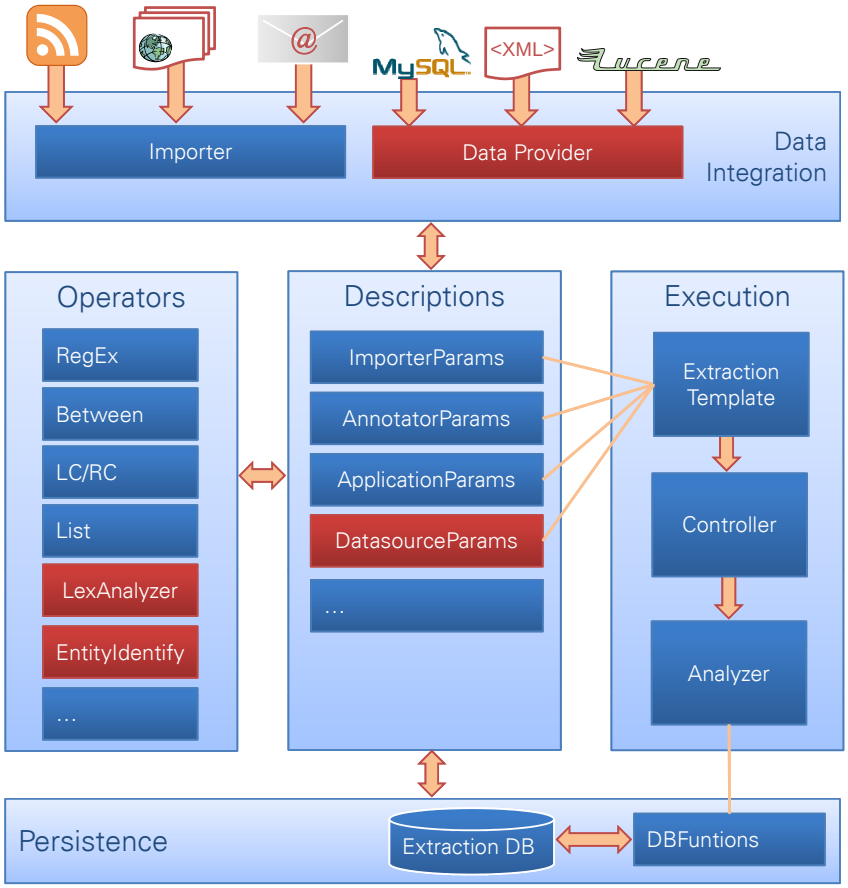


Abbildung 5.1: erweiterte Framework-Architektur

die entsprechenden Datentypen (relationale Datenbank, XML-Datei) implementiert werden müssen.
Abbildung 5.2 zeigt die UML-Struktur der Interfaces. `IDataSource` und `IDataProvider` sind die Basis-Schnittstellen. Das Interface `IDataSource` definiert die Beschreibung von grundlegenden Eigenschaften einer Datenquelle. In erster Linie muss für jede Datenquelle im System eine eindeutige ID zu deren Identifizierung vergeben werden. Außerdem müssen die enthaltenen Datenfelder sowie Primärschlüsselfelder und Standard-Suchfelder definiert werden. Das Standard-Suchfeld bringt den Vorteil, dass bei einer Anfrage der Datenquelle nicht angegeben werden muss, auf welches Feld sich die Anfrage bezieht. Wird zum Beispiel eine Adressdatenbank eingebunden, die mit einem Extraktor zur Identifikation von Namen verbunden wird, muss der Extraktor nicht angeben, dass er das Namensfeld anfragt, wenn dieses als Standard-Suchfeld definiert wurde. Eine konkrete Implementierung des Interfaces beinhaltet darüber hinaus typspezifische Eigenschaften. Dazu gehören vor allem technische Informationen, die zum Einlesen der Daten benötigt werden. Die Gesamtheit der Klassen, die das Interface `IDataSource` implementieren, ist in der Architekturabbildung (5.1) als *DatasourceParams* (Deskriptoren zur Beschreibung strukturierter Datenquellen) bezeichnet.
`IDataProvider` ist die übergeordnete Schnittstelle für den Datenzugriff. Sie beinhaltet Methoden für die Initialisierung und die Beendigung von Verbindungen zur Datenquelle. Dies erfolgt je nach Typ der Datenquelle unterschiedlich. Die Initialisierung kann zum Beispiel das Öffnen einer Datei, den Aufbau der Verbindung zu einer Datenbank oder das Laden eines In-

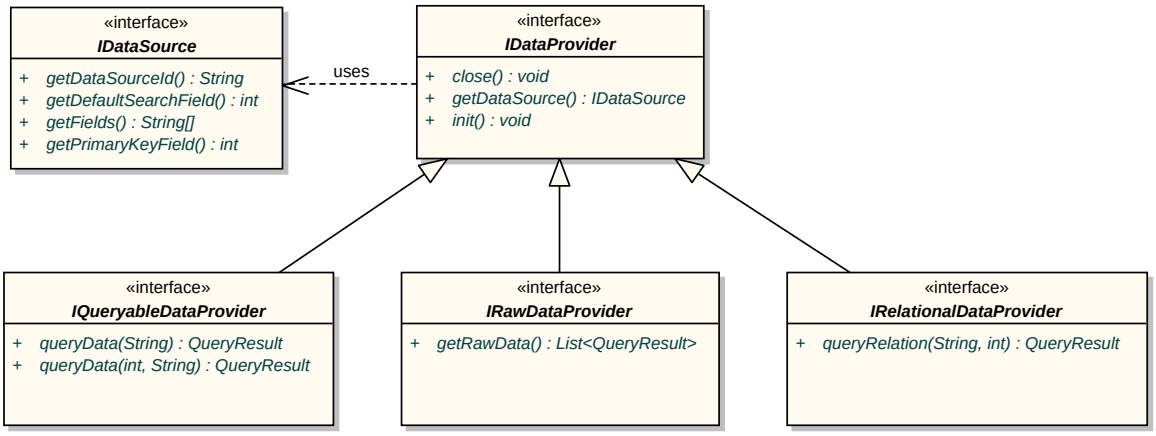


Abbildung 5.2: Interface-Struktur der Datenanbindung

dizes in den Hauptspeicher beinhalten. Die Beendigung erfolgt dann auf entgegengesetzte Art und Weise.
Der eigentliche Zugriff auf die Daten wird durch die von `IDataProvider` abgeleiteten Interfaces geregelt. `IQueryableDataProvider` wird für Datentypen implementiert, auf die ein abfragebasierter Zugriff erfolgt. Für Datenquellen, die den kompletten Inhalt ganzer Listen oder Tabellen zur Verfügung stellen wird das Interface `IFullDataProvider` implementiert. Dies erlaubt zum Beispiel die Verwendung von externen Daten für listenbasierte Operatoren. Bisher werden einfache Listenoperatoren statisch mit ihren Daten instanziiert und hielten diese permanent im Speicher. Die Erweiterung durch dieses Interface stellt demnach nicht nur zusätzliche Funktionalitäten für die in dieser Arbeit entwickelten Operatoren zur Verfügung. Sie bietet zusätzlichen Komfort für bereits vorhandene Komponenten. Desweiteren ist der direkte Zugriff auf die Gesamtdaten einer Quelle von Bedeutung, wenn Daten vor ihrer Verwendung indiziert werden sollen.
Das Interface `IRelationalDataProvider` stellt eine Methode zur Verfügung, die es erlaubt Relationen zwischen Datenquellen abzufragen. Bei der Beschreibung der konkreten Klassen, die verschiedene der genannten Interfaces implementieren, werden die hier genannten Konzepte deutlich. Grundsätzlich muss für jeden Typ externer Datenquellen sowohl `IDataSource` für die Beschreibung der Datenquelle als auch `IDataProvider` für den Zugriff auf die Datenquelle implementiert werden. Die folgenden Abschnitte beschreiben eine konkrete Implementierung für relationale Datenbanken. Dies könnte aber ebenso für XML-Dateien erfolgen, welche auch strukturierte Daten zur Verfügung stellen.

5.3.1 Beschreibung externer Datenquellen

Die Beschreibung externer Datenquellen eines bestimmten Typs wird wie bereits dargestellt durch die Implementierung des Interfaces `IDataSource` ermöglicht. Für diese Arbeit ist es erforderlich eine solche Implementierung für relationale Datenbanken durchzuführen. Abbildung 5.3 zeigt die Klasse `DbDataSource`, die das Interface `IDataSource` implementiert. Zusätzlich zu den zuvor beschriebenen, generischen Methoden werden weitere Methoden implementiert, die die konkreten Eigenschaften von Datenbanken wiedergeben. Dazu gehören der Datenbanktreiber, die URL und die konkrete Datenbanktabelle sowie der Benutzername und das Passwort für den Zugriff auf die Daten.
`DbRelation` ist eine spezielle Datenquelle, die eine Relation zwischen Datenquellen, in dem Fall Datenbanktabellen beschreibt. Sie erbt alle Methoden der Oberklasse `DbDataSource`, da

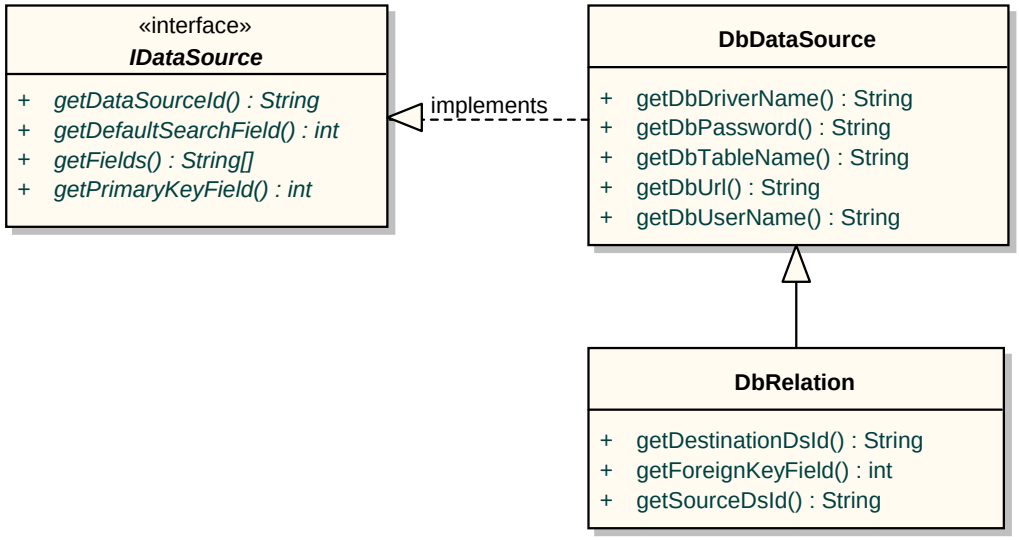


Abbildung 5.3: Implementierung der Datenquellenbeschreibung

die entsprechenden Daten auch für den Zugriff auf eine Relation notwendig sind. Desweiteren beschreibt die Klasse, welche Datenquellen sie verbindet und welches Fremdschlüsselattribut der Verbindungspunkt zwischen den Datenquellen ist. Auf diese Art und Weise wird der Aufwand bei der Auflösung von Beziehungen zwischen Datenquellen von der Anwendung an die Datenanbindungsschicht übertragen. Dies hat den Vorteil, dass die Anwendung keine Kenntnis darüber benötigt, welche Art von Datenquelle zu Grunde liegt. Diese Semantik wird bei der Konfiguration der Anwendung sowie der Beschreibung von Datenquellen und Relationen bestimmt.

5.3.2 Zugriff auf externe Datenquellen

Klassen, die den Zugriff auf eine externe Datenquelle steuern, müssen prinzipiell das Interface `IDataProvider` implementieren. Allein dadurch ist es einer Anwendung jedoch nicht möglich, Daten aus der Quelle abzufragen. Je nachdem welche Zugriffsfunktionen für den entsprechenden Datentyp zur Verfügung gestellt werden, müssen die dazu gehörigen abgeleiteten Interfaces implementiert werden. Für die Einbindung relationaler Datenbanken ist es erforderlich, Funktionen zum Anfragen von Daten und Relationen zu ermöglichen. Daher werden grundsätzlich die Interfaces `IQueryDataProvider` und `IRelationalDataprovider` implementiert. Im weiteren Verlauf der Arbeit wird eine Indizierung externer Daten implementiert, welche, wie im vorherigen Abschnitt beschrieben, komplette Tabelleninhalte benötigt. Aus diesem Grund wird das Interface `IFullDataProvider` implementiert.
Abbildung 5.4 zeigt die Details der Klasse `DbDataProvider` sowie die Verbindung zu den Beschreibungsklassen für die externe Datenstruktur. Die Anfragemethoden realisieren eine Umsetzung der parametrisierten Anfragen einer Anwendung auf datenbanktypische Anweisungen in SQL-Form.

5.3.3 Indizierung externer Datenquellen

Da für einen Datenabgleich von unstrukturierten und strukturierten Daten eine sehr hohe Anzahl von Anfragen an die strukturierten Daten erforderlich sind, sieht die Systemkonzeption für die Datenanbindung Methoden für den performanten Zugriff auf externe Datenquellen

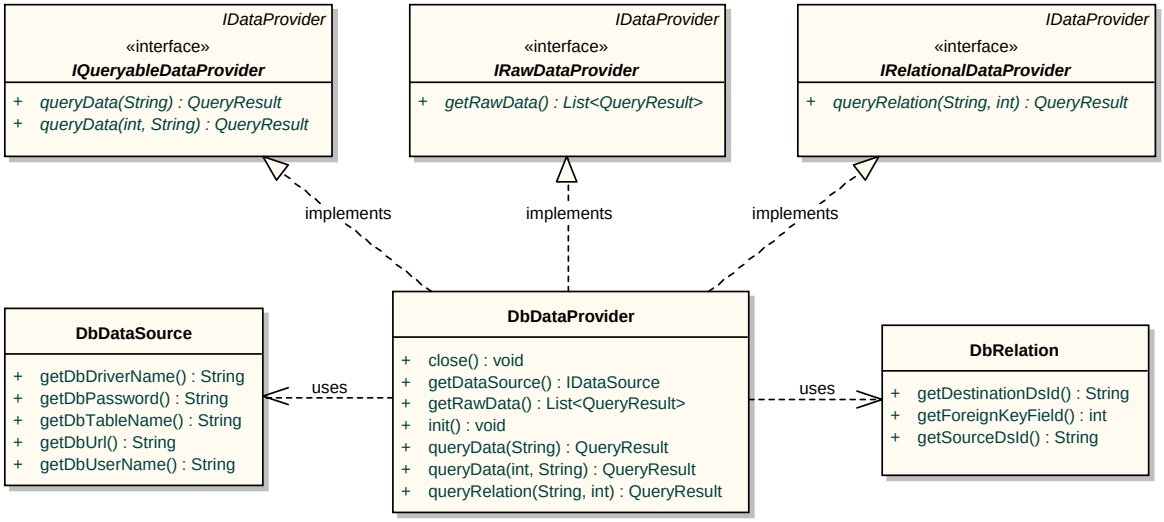


Abbildung 5.4: Implementierung der Datenintegration

vor. Dies wird durch eine Komponente zur Indizierung von Daten implementiert. Mit Hilfe eines Indizes über Inhalte von ID- und Suchfeldern, die im Arbeitsspeicher gehalten werden kann ein sehr schneller Zugriff auf externe Datenquellen ermöglicht werden. Dies ist selbst dann möglich, wenn die Daten wie im Fall von XML auf Dateiebene gespeichert sind. Datenbanksysteme indizieren intern Daten auf ähnliche Art und Weise, um Anfragezeiten zu optimieren. Doch auch bei derartigen Datenquellen ist eine Indizierung auf Anwendungsebene sinnvoll. Die Java-Plattform kommuniziert mit externen Datenbanken über Netzwerkprotokolle, wodurch ein zusätzlicher Zeitaufwand entsteht. Wird ein Datenindex durch die Anwendung im Hauptspeicher gehalten wird dieser Aufwand vermieden.

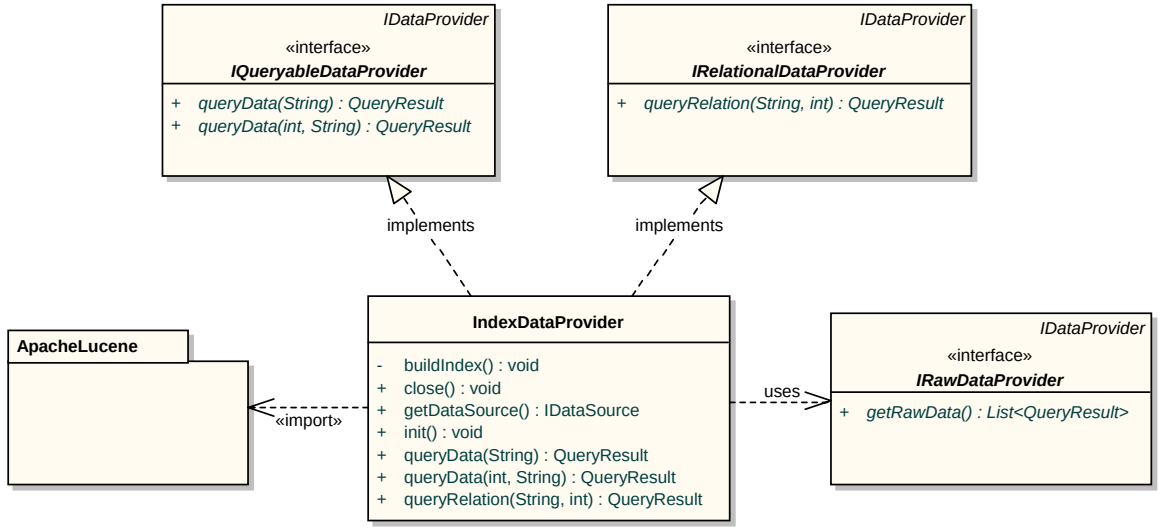


Abbildung 5.5: Implementierung der Datenindizierung

Die Abbildung 5.5 zeigt die Umsetzung einer Indizierung externer Daten. Der Aufbau von Indizes wird durch die Klasse `IndexDataProvider` implementententiert. Der *DataProvider* der zu verwendenen Datenquelle muss zumindest das Interface `IRawDataProvider` implementieren. Die Index-Klasse implementiert das Interface `IQueryableDataProvider` wodurch ein

anfragebasierter Zugriff auf die Daten möglich ist. Darüber hinaus implementiert die Klasse `IRelationalDataProvider`. Dessen Methoden stehen jedoch nur zur Verfügung, wenn die zugrunde liegende Datenquelle auch durch einen `IRelationalDataProvider` zur Verfügung gestellt wird. Die Indizierung kann demnach für beliebige Datenquellen verwendet werden und muss nicht an einen bestimmten Typ angepasst werden.

Die Kernfunktionalität der Indizierung wird durch die Integration des Apache Lucene Projekts[1] übernommen. Dieses wird über eine Bibliothek in das Programm eingebunden und über dessen API angesprochen. Es sorgt für die Erstellung des Indizes sowie dessen Speicherung und das Vorhalten im Arbeitsspeicher. Apache Lucene ist das am Weitesten verbreitete Open-Source-Framework für die Indizierung von Inhalten. Es bietet eine sehr gute Performanz und wird daher den Anforderungen für die Verwendung in diesem Prototypen gerecht.

5.3.4 Gesamtbild der Datenanbindung

Damit ist die Implementierung der Datenanbindung abgeschlossen. Abbildung 5.6 auf der nächsten Seite zeigt das Gesamtbild dieser Schicht. Die entsprechenden Klassen können anhand ihrer Schnittstellen durch die Systemkomponenten der Informationsextraktion genutzt werden.

5.4 Implementierung von Operatoren

Die Erweiterung der Datenintegration des Frameworks ist die Voraussetzung für die im Folgenden beschriebenen Operatoren zur Identifikation von Entitäten in unstrukturierten Daten. Im ersten Schritt werden die in Kapitel 4 entworfenen Komponenten zu Operatoren zusammengefasst. Wie in Abschnitt 2.3 beschrieben, führen Operatoren algebraische Operationen auf einer Eingabe bzw. einer Menge von Eingaben durch. Da die Operatoren entsprechend der definierten Algebra beliebig miteinander kombinier sind, muss die Ausgabe eines Operators immer als Eingabe eines weiteren Operators geeignet sein. Sie haben keine Kenntnis über die Semantik der vor- oder nachgeschalteten Operatoren. Die Semantik entsteht durch die Ausführung eines Extraktionsplans, der festlegt in welcher Reihenfolge und auf Basis welcher Eingaben die Operatoren ausgeführt werden. Die Ein- und Ausgabewerte der verschiedenen Komponenten aus der Systemspezifikation weisen zum Teil sehr spezielle Eigenschaften auf. So wird zum Beispiel für die Phrasenbildung eine Menge von Eingaben benötigt, die jeweils aus einem einzelnen Wort, seiner Position im Text und der syntaktischen Klasse (Substantiv, Verb, ...) des Wortes besteht. Standardmäßig arbeiten die Operatoren nur anhand von Dokumenten bzw. Dokumentsegmenten und deren Positionsangaben.

Aus den genannten Gründen werden die Komponenten in Form von zwei Operatoren implementiert. Der erste Operator führt die Dokumentenvorverarbeitung (Abschnitt 4.2) durch. Dieser wird als *Lexical Analyzer* bezeichnet. Der zweite Operator führt die weiteren Schritte von der Variantenbildung über den Datenabgleich bis hin zur Konsolidierung durch. Er wird als *Entity Identifier* bezeichnet, da er die eigentliche Identifikation von Entitäten mit Kontextwissen durchführt. Beide werden in den nächsten Abschnitten detailliert beschrieben.

[1] http://lucene.apache.org/

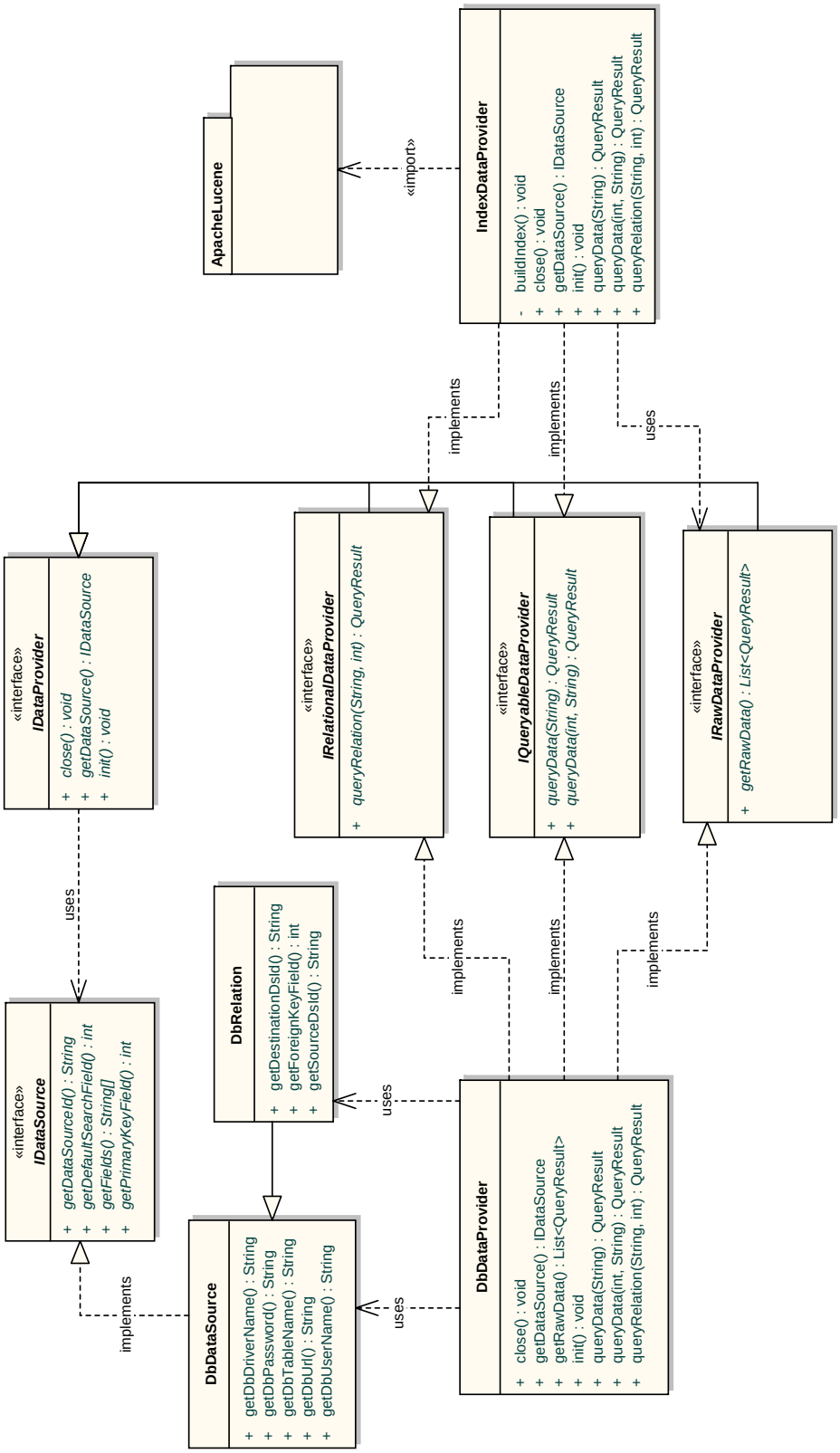


Abbildung 5.6: Übersicht der Datenanbindung

5.4.1 Lexical Analyzer

Der *Lexical Analyzer* stellt die Dokumentenvorverarbeitung auf rein syntaktischer Ebene dar. Die Ausgabe des Operators kann nicht nur für die Identifikation von Entitäten verwendet werden, sondern bietet die Grundlage für weitere Extraktionsschritte. Aus diesem Grund stellt er allgemein eine sinnvolle Ergänzung für das Framework dar.

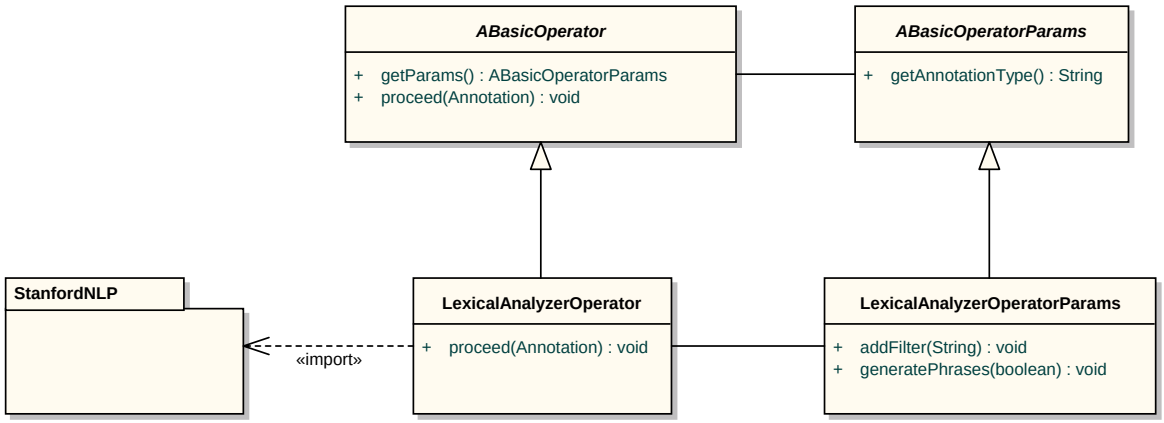


Abbildung 5.7: Implementierung der Lexikalischen Analyse

Abbildung 5.7 zeigt einen Ausschnitt des Frameworks, in dem die Integration der Klasse `LexicalAnalyzerOperator` dargestellt wird. Sie erbt wie alle Operatoren von einer abstrakten Operator-Klasse. In der `proceed`-Methode findet die eigentliche Verarbeitung statt. Als Eingabe erhält die Methode unstrukturierte Inhalte beliebiger Größe. Die Ausgabe kann unterschiedlich konfiguriert werden. Die Konfiguration erfolgt anhand eines Deskriptors, welcher durch die Klasse `LexicalAnalyzerOperatorParams` implementiert wird. Auch der Deskriptor wird von einer abstrakten Oberklasse abgeleitet. Er stellt unter anderem zwei Optionen zur Verfügung. Es ist möglich, nur eine Filterung anhand des syntaktischen Typs vorzunehmen und die gefilterten Wörter auszugeben. Die zweite Möglichkeit entspricht der vollständigen Dokumentenvorverarbeitung aus Abschnitt 4.2. Sie besteht darin, die gefilterten Inhalte vor der Ausgabe in Phrasen zu überführen. Die Filterung wird anhand der Worttypen, die mit der Methode `addFilter` erfasst wurden, vorgenommen. Der Parameter der Methode `generatePhrases` bestimmt, ob eine Phrasenbildung vorgenommen wird oder nicht.

Zur Bestimmung der Worttypen wird die NLP-Bibliothek[2] der Stanford Universität eingebunden. Sie bietet vergleichsweise gute Ergebnisse im Gegensatz zu anderen APIs für natürlichsprachige Verarbeitung von Texten. Sie zeichnet sich vor allem dadurch aus, dass sie auch in Textfragmenten, die keine vollständigen Sätze darstellen, die Wortart zuverlässig bestimmen kann. Ein Part-of-Speech Tagger wie er unter anderem auch von LingPipe[3] zur Verfügung gestellt wird, benötigt in der Regel vollständige Sätze um Wortarten bestimmen zu können. Die hier verwendete OpenSource-Bibliothek hat nicht diese Einschränkung. Der Part-of-Speech Tagger kommt daher dem Inxight-System am Nähesten, welches im SAP-TREX zum Einsatz kommt. Dieses System dient in erster Linie der Suche und Klassifikation großer Dokumente. Es beinhaltet einen Part-of-Speech Tagger, der jedoch nicht über eine Java-Schnittstelle verfügt und deshalb hier nicht zum Einsatz kommen kann.

[2] http://nlp.stanford.edu
[3] http://alias-i.com/lingpipe

5.4.2 Entity Identifier

Wie bereits erwähnt, übernimmt der *Entity Identifier* die Verarbeitungsschritte der Kontextanalyse und der Konsolodierung aus Abschnitt 4.2. Eine Aufteilung der enthaltenen Komponenten auf mehrere Operatoren ist nicht möglich, da die verschiedenen Operatoren in dem Fall eine sehr starke Abhängigkeit hätten. Eine Filterung nach dem *besten Datenbanktreffer* kann zum Beispiel nur durchgeführt werden, wenn vorher ein Datenabgleich erfolgte, der die entsprechenden Eingabewerte zur Verfügung stellt. Aus Gründen der Erweiterbarkeit gibt es die Möglichkeit, zu definieren ob und bis zu welchem Schritt eine Konsolidierung durchgeführt wird. Dies wird benötigt, wenn zukünftig weitere Konsolidierungsalgorithmen implementiert werden, die die Ergebnisse des Datenabgleichs nutzen.

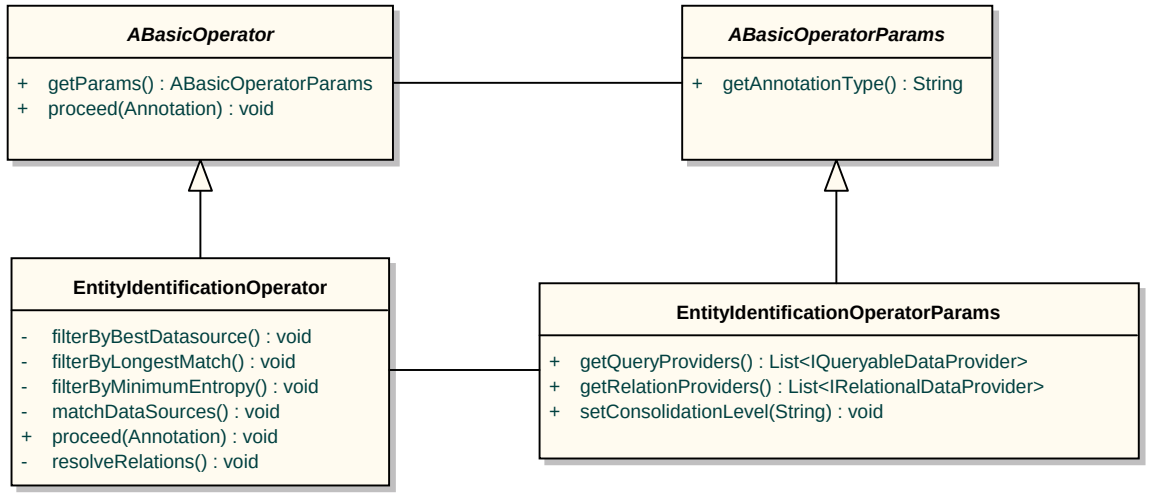


Abbildung 5.8: Implementierung des Entity Identifier

Die Klasse `EntityIdentificationOperator` wird in Abbildung 5.8 in der gleichen Form wie im vorherigen Abschnitt anhand eines entsprechenden Ausschnitts des Frameworks dargestellt. Der Deskriptor des Operators, `EntityIdentificationOperatorParams`, ist in dem Fall mit einer (oder mehreren) in Abschnitt 5.3 beschriebenen Datenquelle verbunden. Die *Data Provider* dieser Datenquellen müssen die Interfaces `IQueryableDataProvider` und `IRelationalDataProvider` implementieren.
Die implementierten Operatoren lassen sich in einen beliebigen Informationsextraktionsfluss integrieren. Ein Beispiel dafür wird in Kapitel 6 vorgestellt.

5.5 Zusammenfassung

Die in Kapitel 4 erarbeiteten Konzepte wurden in Form einer Implementierung umgesetzt. Dazu wurde die konzipierte Systemarchitektur auf die Prinzipien des in Abschnitt 2.3 vorgestellten Frameworks zur Informationsextraktion übertragen. Die Implementierung erfolgt in Form zusätzlicher Operatoren. Das Framework verfügt nun über einen Operator zur Lexikalischen Analyse (*Lexical Analyzer*) von unstrukturieren Inhalten. Außerdem steht ein Operator zur Identifikation von Entitäten (*Entity Identifier*) zur Verfügung, welcher externe strukturierte Daten nutzt.
Die Komponenten, welche die Anbindung der externen Datenstrukturen ermöglichen, konnten nicht ohne die konzeptionelle Erweiterung des Frameworks integriert werden. Die Erweiterungen beinhalten die Einführung einer Datenanbindungsschicht, welche in die beste-

hende Datenintegrationsschicht des Frameworks eingefügt wurde. Es ist nun möglich nicht mehr nur unstrukturierte Daten zur Verarbeitung zu importieren, sondern auch strukturierte Daten für verschiedene Zwecke innerhalb des Frameworks zu nutzen. Entsprechende Komponenten, die diese Datenintegration umsetzen wurden implementiert.

Kapitel 6

Evaluation

In Kapitel 4 der vorliegenden Arbeit erfolgte die Konzeption einer Systemarchitektur zur Identifikation von Entitäten in unstrukturierten Daten. Die Implementierung, die im vorherigen Kapitel beschrieben wurde, setzt die Konzeption prototypisch um. In dem aktuellen Kapitel werden sowohl die Konzepte der Systemarchitektur als auch die Implementierung des Prototyps an den Anforderungen gemessen, die in Kapitel 2 definiert wurden.
Zuerst wird überprüft, ob und inwiefern Konzeption und Implementierung die funktionalen Anforderungen abgedecken. In der Anforderungsanalyse wurden neben funktionalen Anforderungen auch qualitative Anforderungen definiert, an denen die Ergebnisse der Anwendungsausführung gemessen werden. Zu diesem Zweck wird der Prototyp einem Testlauf mit Referenzdaten unterzogen, dessen Ergebnisse auf Grundlage der Anforderungen aus Kapitel 2 ausgewertet werden. Ausgehend von dieser Auswertung werden Vorschläge gemacht, wie die Resultate durch Erweiterungen verbessert werden können. Die Evaluation wird abschließend noch einmal zusammengefasst.

6.1 Bewertung der funktionalen Anforderungen

In der Anforderungsanalyse wurde deutlich, dass diese Arbeit in erster Linie an der Qualität der Extraktionsergebnisse gemessen wird, als weniger an funktionalen Aspekten einer konkreten Implementierung. Dennoch gilt es, Voraussetzungen einzuhalten, die sich vordergründig auf die Integration der Anwendung in das bestehende SAP Research UIM-Framework und auf die Verwendung strukturierter Daten beziehen.
In den funktionalen Anforderungen wurden neben der *Entwicklung eines Operators* zur Identifikation von Entitäten sowohl die *Verwendung strukturierten Daten* als auch eine *Abstraktion der konkreten Datenstruktur* gefordert. Damit wird erreicht, dass sich beliebige strukturierte Daten mit der entwickelten Anwendung verwenden lassen. Der entwickelte *Operator* muss sich *nahtlos* in die Struktur des *Frameworks integrieren* ohne die Konzepte grundsätzlich zu verändern. Grundlegende *Veränderungen am Framework* dürfen nur *für* die *Integration strukturierter Datenquellen* vorgenommen werden. Letzteres war bis dato noch nicht möglich. Die genannten Anforderungen spiegeln sich teilweise in der Systemkonzeption wider. Teilweise können sie aufgrund ihrer konkreten Beschaffenheit nur durch die Implementierung umgesetzt werden.
Die Konzeption ist bewusst unabhängig von der Struktur des Frameworks erarbeitet um unter anderem sichtbar zu machen, inwiefern Erweiterungen an der Architektur des Frameworks erforderlich sind, um die Ziele dieser Arbeit zu realisieren. In der Konzeption wurde unter anderem ein Integrationsprinzip für strukturierte Daten entwickelt. Damit wird nicht nur ermöglicht, Daten unterschiedlicher Struktur, d.h. mit unterschiedlichem Schema, son-

dern auch unterschiedlichen Typs zu integrieren. Das umfasst neben relationalen Datenbanken auch datei- und indexbasierte Daten. Eine derartige Infrastruktur fehlte dem Frameworks bisher, weshalb dies eine Erweiterung der Frameworkarchitektur erfordert. Dieser Erweiterung wird in der Implementierung des Prototyps vorgenommen, womit den Anforderungen nach einer Erweiterung der Framework-Architektur zur Integration externer Datenstrukturen sowie der Abstraktion von einer konkreten Datenstruktur bereits nachgekommen wird.
Die Anforderung, einen Operator zu entwickeln, der Entitäten in unstrukturierten Daten identifiziert und dafür strukturierter Daten verwendet, wurde ebenfalls in der Konzeption berücksichtigt. Die genauen Methoden und Algorithmen wurden dort definiert und in eine komponentenbasierte Architektur integriert. Die Anforderung, den Operator nahtlos in das Framework zu integrieren, konnte jedoch erst in der Implementierung erreicht werden. Zu diesem Zweck wurden die konzipierten Komponenten zu zwei Operatoren zusammengefasst. Sie agieren entsprechend der Prinzipien des Frameworks und verwenden dabei generische Ein- und Ausgabeparameter. Die Operatoren können beliebig mit anderen, bereits vorhandenen Operatoren des Frameworks kombiniert werden.
Damit gelten die funktionalen Anforderungen der Arbeit als erfüllt. Die nachfolgende Anwendung eines Testszenarios wird zeigen, dass das System unter reellen Voraussetzungen einsetzbar ist. Die qualitativen Auswertung der Ergebnisse wird im weiteren Verlauf des Kapitels vorgenommen.

6.2 Testszenario

In diesem Abschnitt wird ein Testszenario beschrieben, in welchem die entwickelte Anwendung zu Testzwecken ausgeführt wird. Dies beinhaltet zuerst einen kurzen Überblick zur Systemumgebung, die für das Szenario zur Verfügung steht. Als nächstes wird ein Referenzdatensatz erstellt, der für die Ausführung verwendet wird. Zuletzt wird die Ausführung anhand eines Informationsextraktionsplans beschrieben, welcher das Szenario auf die Anwendung überträgt.

6.2.1 Systemumgebung

Die Anwendung wird auf einem handelsüblichen, IBM kompatiblen PC ausgeführt. Er ist mit einem Intel Pentium IV HT Prozessor mit 3.2GHz und einem Arbeitsspeicher von 2GB RAM ausgestattet. Als Betriebssystem kommt Microsoft Windows XP Professional mit Service Pack 2 zum Einsatz. Die Softwareumgebung entspricht im wesentlichen der Implementierungsumgebung, welche in Abschnitt 5.1 beschrieben wurde. Der Unterschied besteht darin, dass die Datenbank aus Gründen der Performanzoptimierung auf eine externe Maschine ausgelagert wurde. Dabei handelt es sich um das gleiche Modell, was auch für die Anwendungsausführung verwendet wird.

6.2.2 Referenzdaten

Das Ziel des Testszenarios ist es, die Ergebnisse der Anwendungsausführung zu messen. Dabei kommen insbesondere die in Abschnitt 2.6 vorgestellten Metriken Precision und Recall zum Einsatz. Da es sich dabei um Metriken handelt, die ein Verhältnis zwischen Ist und Soll zum Ausdruck bringen, müssen als erstes Referenzdaten geschaffen werden, die das erwünschte Ergebnis widerspiegeln.
Die Erstellung eines Referenzdatensatzes beinhaltet die Auswahl eines Dokumentenkorpus, der für die Identifikation von Entitäten auf Basis strukturierter Daten verwendet wird. In diesem Dokumentenkorpus müssen alle Entitäten gekennzeichnet werden, von denen

erwartet wird, dass sie von der Anwendung erkannt werden. Die Kennzeichnung wird von einem Domänenexperten vorgenommen.

Dabei tut sich ein Problem auf. Die unstrukturierten Daten, die für diese Evaluation verwendet werden, stellen Forumbeiträge des SAP Developer Network dar. In diesen Beiträgen wird versucht, Instanzen von SAP-Komponenten zu identifizieren. Die Identifikation erfolgt anhand von Komponentenbezeichnung selbst, aber auch auf Basis von Produktnamen, Termen, Abkürzung und veralteten Schreibweisen. Die genauen Zusammenhänge der Daten wurden in Abschnitt 2.2 dargestellt. Der Domänenexperten hat implizites Wissen dieser strukturierten Daten, um die Kennzeichnung in den Referenzdaten vorzunehmen. Er nimmt somit eine Auflösung von Mehrdeutigkeiten vor, wie sie auch von der Anwendung mit explizitem Wissen durchgeführt wird. Der entscheidene Punkt ist, dass der Domänenexperte nicht der Autor eines Beitrags ist und somit nicht dessen genaue Intention kennt. Auf diese Art und Weise ist es möglich, dass der Experte eine andere Entität im Text identifiziert, als vom Autor gemeint. Die Anwendung kann auch nicht die Intention des Autor kennen, jedoch handelt sie nach vorgegebenen, rationalen Regeln, wodurch eine Abweichung messbar wird. Der Domänenexperte handelt subjektiv, weshalb in unterschiedlichen Situtationen unterschiedlich geartete Abweichungen entstehen können.

Dies ist ein typisches Problem bei der Erstellung von Referenzdatensätzen zur Evaluation elektronischer Systeme. Komplette Textdokumente, in diesem Fall Forumbeiträge, enthalten sehr viel Inhalt, bei dessen Erzeugung der Autor zu ausführlichen Umschreibungen neigt. Aus diesem Grund ist es für den Domänenexperten besonders schwierig, punktgenau Entitäten zu identifizieren. In dieser Arbeit wird versucht, diesen Fakt abzuschwächen. Forumbeiträge sind Hypertextdokumente. Sie gelten im Sinne der semantischen Informationsverarbeitung als unstrukturiert. Sie weisen jedoch eine Struktur auf, die dem Layout und der Verknüpfung von Dokumenten dient. Die Verknüpfungselemente werden in Form von Anker-Tags erzeugt, die eine kurze textuelle Beschreibung des Verknüpfungsziels enthalten. Beim Einfügen eines solchen Link-Ankers ist der Autor dazu bestrebt, kurz und knapp wiederzugeben, was den Leser erwartet, wenn er dieser Verknüpfung folgt.

Beobachtungen der Foruminhalte haben gezeigt, dass im Inhalt eines Anker-Tags mit sehr hoher Wahrscheinlichkeit auf eine Komponentenbezeichnung bzw. einen komponentenbezogenen Term zurückgegriffen wird, wenn das Zieldokument von der entsprechenden Komponente handelt. Bei der Auflösung von Mehrdeutigkeiten, kann der Domänenexperte neben den Informationen, die sich im Text befinden, zusätzlich den Inhalt des Zieldokuments hinzuziehen. Aus diesem Grund eignen sich die Inhalte der Anker-Tags im Vergleich zu ganzen Dokumenten besser als Referenzdaten. Die Informationen werden wie bereits dargestellt wesentlich konkreter ausgedrückt, was die Arbeit des Domänenexperten erleichtert. Außerdem enthalten sie potenziell mehr Entitäten pro Dateneinheit, was die Auswertung der Ergebnisse wesentlich vereinfacht.

Der erste Schritt zur Erstellung des Referenzdatensatzes ist die Sammlung von Anker-Tag-Inhalten aus SDN-Forumseiten. Diese können sehr einfach über einen regulären Ausdruck aus dem Quelltext eines HTML-Dokuments herausgefiltert werden. Die Filterung muss nicht manuell erfolgen, sondern kann mit Hilfe des Frameworks durchgeführt werden. Ein entsprechender Operator steht zu diesem Zweck zur Verfügung. Der genaue Extraktionsplan wird im folgenden Abschnitt (6.2.3) dargestellt.

Für die Erstellung des Referenzdatensatzes wurde nur der Teil des Extraktionsplans ausgeführt, der als Ergebnis die Inhalte von Anker-Tags aus HTML-Dokumenten extrahiert. Ein Dokument ist in diesem Fall eine Seite des SDN-Forums, welche mehrere Beiträge (Posts) enthält, die zu einer Diskussion (Thread) zusammengefasst werden.

Tabelle 6.1 enthält eine Statistik der Referenzdaten und in Tabelle 6.2 wird beispielhaft die

Anzahl der Dokumente	133
Anzahl der Anker-Tags	1393
Anzahl identifizierter Entitäten	149

Tabelle 6.1: Statistik der Referenzdaten

Inhalt des Anker-Tags	**enthaltene Entitäten**
Alerts with variables from the messages payload (XI)	BC-XI
SAP NetWeaver	NW
Exchange Infrastructure	BC-XI
SAP Solution Manager	SV-SMG
How to integrate XI alerts into Portal UWL?	BC-XI, EP-BC-UWL
NetWeaver AS, General	NW
CRM - Web Application	SRD-CRM
Supplier Relationship Management (SRM)	SRD-SRM
Alert Management - use it in CCMS	BC-CCM
SAP Business One	SBO
SAP Master Data Management	MDM
SAP Mobile Infrastructure	BC-MOB
Customer Relationship Management	SRD-CRM
Supply Chain Management	SRD-SCM
BI Monitors in CCMS	SRD-BI, BC-CCM
Monitoring TREX with CCMS	BC-TRX, BC-CCM
CCMS Monitor Sets	BC-CCM
SAP MI in the Computing Center Management System	BC-MOB, BC-CCM
SAP APO Monitoring with CCMS	SCM-APO, BC-CCM
Working with XML in DI API	SBO

Tabelle 6.2: Beispieleintrag des Referenzdatensatzes

Kennzeichnung von Entitäten in den Referenzdaten dargestellt. Zu jedem Anker-Tag wird eine Menge von SAP-Komponenten zugeordnet. Diese werden in der Kurzschreibweise angegeben. Aus Platzgründen wird auf die vollständige Darstellung aller Daten verzichtet. Im folgenden Abschnitt wird ein Extraktionsplan beschrieben, durch dessen Ausführung versucht wird, die in diesem Abschnitt gekennzeichneten Entitäten in den Referenzdaten automatisch zu erkennen.

6.2.3 Extraktionsplan

Im SAP Research UIM-Framework erfolgt die Informationsextraktion über sogenannte Extraktionspläne. Sie sind in den Architekturabbildungen 2.5 auf Seite 11 und 5.1 auf Seite 49 als *Extraction Template* dargestellt. Für die Erstellung eines Extraktionsplans werden zunächst Annotatoren definiert, welche als parametrisierte Operatoren aufgefasst werden können. Die Menge aller definierten Annotatoren bildet dann, angeordnet in einer bestimmten Reihenfolge, den Extraktionsplan. Anhand der Reihenfolge werden die Ein- und Ausgabewerte der Annotatoren bestimmt.
Für die Identifikation von Entitäten in unstrukturierten Daten wird nun ein Extraktionsplan entsprechend dem hier beschriebenen Szenario erstellt. Prinzipiell folgt ein Extraktionsplan zur Identifikation von Entitäten in unstrukturierten Inhalten dem Ablauf der in Kapitel 4 dargestellten Systemarchitektur. Die enthaltenen *generischen* Operatoren müssen zuvor noch

in parametrisierte Annotatoren überführt werden. Anhand der konkreten Implementierung aus Kapitel 5 in folgendem Quellcodeausschnitt dargestellt.

```
1  public class ScenarioExtractionPlan {
2    public static void main(String[] args) {
3      Analyzer analyzer = new Analyzer();
4      final AnnotIdArray sdnImpIds;
5      IBasicImporter sdnImporter = AnnotatorFactory.createImporter(impParams);
6      sdnImpIds = analyzer.importDataSource(new DataSource(0,MainConst.import_annotDSName),
           sdnImporter);
7      final AnnotIdArray htmlAnchorIds;
8      IBasicAnnotator htmlAnchorAnnot = AnnotatorFactory.createAnnotator(rxrParams);
9      htmlAnchorIds = analyzer.proceed(sdnImpIds, htmlAnchorAnnot);
10     final AnnotIdArray nounPhraseIds;
11     IBasicAnnotator nounPhraseAnnot = AnnotatorFactory.createAnnotator(lexAnaParams);
12     nounPhraseIds = analyzer.proceed(htmlAnchorIds, nounPhraseAnnot);
13     IBasicAnnotator sapCompAnnot = AnnotatorFactory.createAnnotator(entIdentParams);
14     analyzer.proceed(nounPhraseIds, sapCompAnnot);
15   }
16 }
```

1. In den Zeilen 4-6 werden die Forumseiten importiert. Dazu wird ein Array definiert in dem die Ausgabeannotationen gespeichert werden. Der Importer (*sdnImporter*) wird anhand seines Deskriptors (*impParams*) instanziiert. Letztlich wird dieser an den *Analyzer* übergeben, welcher die Ergebnisse in dem zuvor definierten Array speichert.

2. In den Zeilen 7-9 werden die Anker-Tags extrahiert. Diese bekommen als Eingabeparameter die Annotationen übergeben, welche vom Importer erzeugt wurden (*sdnImpIds*).

3. In den Zeilen 10-12 werden die Nominalphrasen gebildet. Auch hier werden die Ausgaben des vorherigen Annotators als Eingabe verwendet (*htmlAnchorIds*).

4. Letztlich werden in den Zeilen 13-14 die Entitäten identifiziert. Da die Ausgabeannotationen nicht mehr für weitere Schritte benötigt werden, erfolgt auch keine Zwischenspeicherung dieser.

Da die Identifikation der Entitäten nicht über den gesamten Text der Forumseiten ausgeführt wird, wurde zusätzlich ein sogenannter *RxR Operator* in den Extraktionsplan integriert. Dieser wird mit zwei regulären Ausdrücken konfiguriert, welche die linke und die rechte Begrenzung von zu extrahierenden Inhalten repräsentieren. Auf diese Art und Weise wird zusätzlich gezeigt, dass sich die Operatoren zur Identifikation von Entitäten wirklich nahtlos in das Framework eingliedern, da sie an dieser Stelle direkt mit einem bereits vorhandenen Operator kombiniert werden.
Abbildung 6.1 stellt den detaillierten Extraktionsplan dar, der für die vorliegende Evaluation ausgeführt wird. Zentrale Bestandteile sind die vier Annotatoren *SDN Importer*, *HTML Anchor Annotator*, *Noun Phrase Annotator* und *SAP Component Annotator*. Sie werden jeweils auf Basis eines Operators und dessen Parameter generiert, welche oberhalb des Annotators abgebildet sind. Die Erzeugung von Ausgabeannotationen anhand von Eingabeannotationen wird in der Abbildung als Informationsfluss von links nach rechts dargestellt. Im Folgenden werden die Annotatoren detailliert beschrieben.

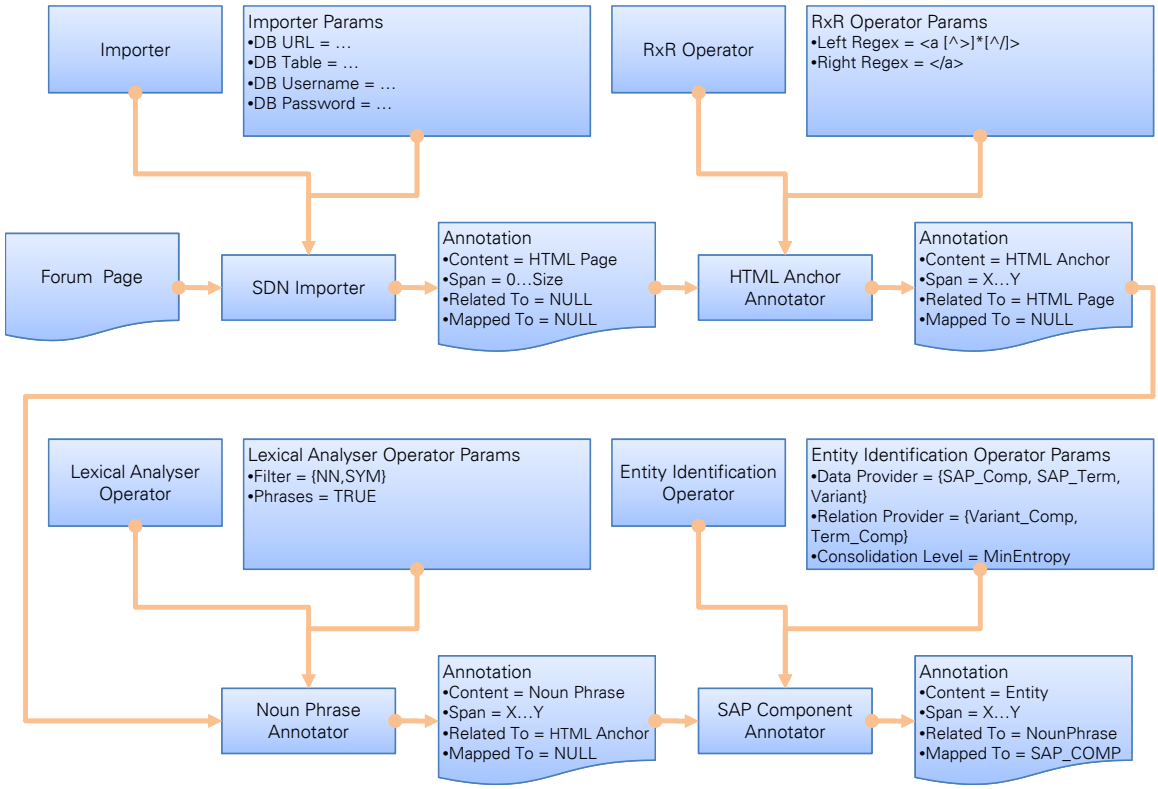


Abbildung 6.1: Extraktionsplan

SDN Importer

Der *SDN Importer* ist für die Beschaffung der unstrukturierten Inhalte in Form von Forumseiten aus dem SDN verantwortlich.

Operator Ein *Importer* ist ein spezieller Operator, der Annotationen erzeugt, selbst aber ohne Eingabe auskommt. Er ist für die Erzeugung der initialen Annotation eines Extraktionsplans zuständig. Der *SDN Importer* erzeugt diese initiale Annotation mit den Daten aus einer Datenbank. Zur Vereinfachung der Darstellung wurde dieser Inhalt als Eingabe gekennzeichnet. In der genannten Datenbank sind HTML-Dokumente aus dem SDN-Forum abgespeichert. Diese wurden zu einem früheren Zeitpunkt mit einem Webcrawler heruntergeladen und zum Zweck der wiederholten Anwendung aufbewahrt.

Parameter Die Parameter des *SDN Importers* sind die Verbindungs- und Zugriffsdaten für die Datenbank, sowie die zu verwendende Datenbanktabelle zum Abruf der HTML-Dokumente.

Annotation Die erzeugten Annotationen beinhalten das HTML-Dokument. Da das Dokument nicht zu einem übergeordneten Element gehört, wird als Bereichsangabe (Span) der Anfangs- und Endpunkt des Dokuments gespeichert und die Zugehörigkeit (Related To) mit `null` referenziert. Ebenso wird durch das Dokument kein externes Objekt abgebildet (Mapped To), weshalb dies ebenfalls mit `null` referenziert wird.

HTML Anchor Annotator

Der *HTML Anchor Annotator* ist für das Herausfiltern der Anker-Tag-Inhalte aus den Dokumenten zuständig, welche im vorherigen Schritt in den Extraktionsplan geladen wurden.

Operator Der *RxR Operator* extrahiert den Inhalt einer Ausgabe, der sich zwischen zwei gegebenen regulären Ausdrücken befindet.

Parameter Als Parameter wird der linke und rechte reguläre Ausdruck angegeben, zwischen denen sich die zu extrahierenden Inhalte befinden. Der linke reguläre Ausdruck spiegelt das Start-Tag eines HTML-Ankers wider, während der rechte reguläre Ausdruck das abschließende Tag beschreibt.

Annotation Eine Annotation besteht aus dem extrahierten Inhalt, sowie dessen Start- und Endposition im Ursprungsdokument. Dies wird durch das *Related To* Attribut referenziert. In der Annotation erfolgt ebenfalls keine Abbildung auf ein externes Objekt.

Noun Phrase Annotator

Der *Noun Phrase Annotator* hat die Aufgabe, die Inhalte der Anker-Tags zu filtern und aus den gefilterten Inhalten Phrasen zu bilden.

Operator Der *Lexical Analyser Operator* ist Teil dieser Arbeit und wurde bereits in Abschnitt 5.4.1 ausführlich beschrieben.

Parameter Die für die Filterung zu Grunde liegenden Wortarten werden anhand ihrer Abkürzungen als Parameter für den Operator übergeben. *NN* steht für Nomen und *SYM* steht für Symbole, Kurznamen und Zeichen. Diese sind für die Identifikation deshalb erforderlich, da Abkürzungen (z.B. für Produkte) durch den Part-of-Speech Tagger oft als solche gekennzeichnet werden. Außerdem wird der Operator durch das *Phrases*-Attribut angewiesen, die gefilterten Inhalte zu Phrasen zusammenzuführen.

Annotation Die Annotation dieses Annotators hat die gleichen Eigenschaften wie die des *HTML Anchor Annotator*. Es werden ebenfalls kleine Segmente des Textes extrahiert und ebenfalls keine externen Objekte referenziert.

SAP Component Annotator

Der *SAP Component Annotator* ist schließlich für die Identifikation der Entitäten im Text zuständig. Im vorliegenden Fall werden SAP Komponenten identifiziert.

Operator Auch der *Entity Identification Operator* wurde in dieser Arbeit ausführlich beschrieben. Für weitere Informationen sei deshalb an Abschnitt 5.4.2 verwiesen.

Parameter Der Annotator erhält eine Menge von *Data Providern* sowie eine Menge von *Relation Providern*, welche zum Datenabgleich und zur Auflösung von Beziehungen in den strukturierten Daten eingesetzt werden. Zusätzlich wird noch festgelegt, bis zu welchem Grad die Konsolidierung durchgeführt werden soll. In diesem Fall wird bis zur Bestimmung der minimalen Entropie konsolidiert, was der Ausführung aller vorhandenen Regeln entspricht.

Annotation Die Annotation des *SAP Component Annotator* ist die abschließende Ausgabe dieses Extraktionsplan. Sie enthält den Inhalt des HTML-Dokuments, der einer Entität entspricht. Genauer gesagt enthält sie den Inhalt eines Anker-Tags, der als Entität identifiziert wurde. Die Positionsangaben beziehen sich immer auf das ursprüngliche Dokument. Die Start- und Endpositionen der Entitäten werden demnach ebenso gespeichert. Die Referenz (Related To) bezieht sich auf die Nominalphrase. Bei dieser Annotation wird mit dem Attribut *Mapped To* die Referenz auf eine Instanz einer SAP Komponente gespeichert. Damit wird die Verbindung eines Textsegments mit einem konkreten Eintrag der Datenbank hergestellt.

6.3 Auswertung des Testszenarios

In diesem Abschnitt erfolgt die Auswertung des Testszenarios. Dazu wird als erstes das Ergebnis der Messungen zu Precision und Recall vorgestellt. Die darauf folgende Bewertung dieser Ergebnisse macht eine Ursachenanalyse erforderlich, welche anschließend vorgenommen wird. Zum Abschluss dieses Abschnitts werden Vorschläge gemacht, wie die Resultate des hier entwickelten Systems verbessert werden können.

6.3.1 Ergebnis

Das Ergebnis der Ausführung des Testszenarios wurde manuell ausgewertet und mit den Referenzdaten verglichen. Die Bedeutung der dabei zu ermittelnen Kennzahlen wurde in Abschnitt 2.6.1 dargestellt. Aus der Menge der Anker-Tag-Inhalte wurde in den Referenzdaten eine Menge von Entitäten gekennzeichnet. Über eine gekennzeichnete Entität lässt sich die Aussage treffen, ob sie bei der Ausführung des Testszenarios gefunden (True Positive) oder nicht gefunden (False Negative) wurde. Eine Textstelle, die fälschlicherweise als Entität gekennzeichnet (False Positive) wurde, kann ebenfalls konkret benannt werden. Textstellen, die korrekterweise nicht als Entität erkannt (True Negative) wurden, lassen sich jedoch nicht zählen. Aus diesem Grund handelt es sich bei TN um eine offene Menge, die sich nicht als konkreter Wert angeben lässt. Für die restlichen Kennzahlen TP, FP und FN werden die Werte in Tabelle 6.3 dargestellt.

True Positive (TP)	89
False Positive (FP)	52
False Negative (FN)	60

Tabelle 6.3: Ergebnisse des Testszenarios

Mit Hilfe dieser Kennzahlen lassen sich nun die Werte für Precision (Gleichung 6.1) und Recall (Gleichung 6.2) errechnen. Eine Bewertung der Ergebnisse wird im folgenden Abschnitt vorgenommen.

$$Precision = \frac{TP}{TP + FP} = \frac{89}{141} = 0,6312 = 63,12\% \tag{6.1}$$

$$Recall = \frac{TP}{TP + FN} = \frac{89}{149} = 0,5973 = 59,73\% \tag{6.2}$$

6.3.2 Bewertung und Ursachenanalyse

Betrachtet man die erzielten Werte für Precision und Recall und zieht die Anforderungen aus Abschnitt 2.4 hinzu, entsteht leicht der Eindruck, dass das Ergebnis des Testszenarios

diese Anforderungen nicht in vollem Umfang erfüllt. Der Recall sagt aus, dass nur knapp mehr als die Hälfte der enthaltenen Enitäten in einem unstrukturierten Dokument identifiziert werden konnten. Anhand der Precision wird deutlich, dass weniger als zwei Drittel der im Dokument identifizierten Entitäten korrekt sind. Diese Ergebnisse können als gut jedoch nicht als hervorragend bewertet werden.
Die Ergebnisse müssen jedoch differenziert betrachtet werden. Dabei spielen zwei zentrale Aspekte eine Rolle. Zum einem wird die Methode zur Erstellung der Referenzdaten hinterfragt. Zum anderen hat die Qualität der verwendeten strukturierten Daten erheblichen Einfluss auf die Ergebnisse der Informationsextraktion.

Probleme mit Referenzdaten

Wie bereits in Abschnitt 6.2.2 kurz angedeutet wurde, ist die Erstellung von Referenzdaten mit diversen Problemen behaftet. Dort wurde die Abweichung von Autorenintention und Experteneinschätzung diskutiert. Es wurde außerdem versucht diesem Problem entgegen zu wirken. Das stellt allerdings nur ein Teilproblem dar. Der vorliegende Referenzdatensatz wurde von derselben Person erstellt, die auch die Anwendung entwickelt hat. Es handelt sich demnach nicht um einen Domänenexperten im klassischen Sinn. Bei dieser Konstellation ist eine entscheidene Frage, inwiefern sich das Wissen des Domänenexperten mit dem Wissen deckt, welches der Anwendung in Form der strukturierten Daten zur Verfügung steht.
Die vorliegenden Daten enthalten mehr als 100'000 Terme, 2'500 Produktnamen und 900 Komponentenbezeichnungen. Daher muss man sich die Frage stellen, ob alle Resultate die als False Positive gekennzeichnet wurden, also fälschlicherweise ein Textsegment als Entität identifizieren, auch wirklich alle als solche zu werten. Im Sinne der strukturierten Daten kann man diese teilweise als korrekt auffassen. Das lässt sich genau so schwer in Zahlen ausdrücken wie der Fakt, dass der Domänenexperte unter Umständen Wissen zur Verfügung hat, welches nicht durch die strukturierten Daten abgebildet wird. Dazu gehört unter anderem das Wissen, was er aus dem Kontext eines Dokuments entnimmt. Die Anwendung betrachtet nur ein Textsegment für sich. Daraus ergibt sich die Frage ob die Resulte, die als False Negative gewertet wurden auch wirklich prinzipiell durch die Anwendung erreicht werden können.
In der Praxis entgegnet man diesem Problem, indem *echte* Domänenexperten hinzugezogen werden. Mehrere Experten erstellen jeweils unabhängig einen Referenzdatensatz, der sowohl das spezifische Fachwissen als auch die subjektive Komponente des Experten widerspiegelt. Aus der Menge aller Referenzdaten lassen sich unter anderem Schnittmengen und Abweichungen bestimmen. Daraus werden die maximalen Werte für Precision und Recall errechnet, die eine Anwendung unter Einbeziehung aller Expertenmeinungen erreichen kann. Für die vorliegende Arbeit steht diese Möglichkeit jedoch nicht zur Verfügung.

Probleme mit der Datenqualität

Die Darstellung der Probleme mit Referenzdaten lässt keine greifbare Aussage über Wertabweichungen zu, die dadurch verursacht werden. Anhand einer weiteren Problemklasse ist es möglich konkrete Zahlen durch konkrete Daten zu bestimmen. Bei der Bewertung von Ergebnissen gilt die Annahme, dass die Anwendung mit strukturierten Daten gearbeitet hat, welche durch die Referenzdaten repräsentiert werden. Die vorhandenen strukturierten Daten weisen jedoch sowohl Inkonsistenzen als auch Fehler auf und sind teilweise unvollständig. Dies konnte bei einer detaillierten Analyse der Testergebnisse des Anwendungsszenarios ermittelt werden.

Inkonsistenzen zeichnen sich unter anderem dadurch aus, dass Abkürzungen mehrfach vergeben werden. Ein sehr prägnantes Beispiel dafür ist, dass die Zeichenfolge *SAP* als Abkürzung für den Term *service access point* steht. Dieser Term steht in Verbindung mit der SAP-Komponente *Industry Solution for Telecommunications*. In Folge dessen wurde diese Komponente fälschlicherweise 15 Mal als Entität identifiziert. Dies hat einen negativen Einfluss auf die Precision, da sie sich mit jedem False Positive verschlechtert.

Fehler erscheinen in der Datenbank insbesondere durch falsche Zuordnung von Fremdschlüsseln. Die Abkürzung *BPM* ist auf korrekte Art und Weise mit dem Term *Business Process Management* verbunden. Dessen Fremdschlüssel für die Verbindung zu einer Komponente enthält aber die ID der Komponente *Policy Management for Financial Services* anstatt der ID für *Basic Component Business Process Management*. Auf diese Art werden sowohl Precision als auch Recall negativ beeinflusst. In 14 Fällen wurde eine falsche Entität identifiziert während die korrekte Entität nicht erkannt wurde. Für das falsche Identifizieren entsteht jeweils ein False Positive und für die eigentliche Entität, die nicht identifiziert wurde kommt noch jeweils ein False Negative hinzu.

Unvollständig kann man die Datenbank insofern betrachten, als dass keine Produktszenarien abgebildet werden, wie sie in der *SAP-Sprache* üblich sind. Kombinationen von Produkten oder Konfigurationen werden oft anhand ihrer Abkürzungen und mit einem Bindestrich verbunden dargestellt. Ist von *AII-XI-SAP R/3* die Rede, deutet dies darauf hin, dass es sich um eine weit verbreitete Konfiguration des SAP R/3 Systems im Bereich von RFID-Systemlandschaften handelt. Eine Auto-ID-Infrastruktur (AII) wird mit Hilfe der Exchange Infrastuktur (XI) an das SAP R/3 angebunden. In den Referenzdaten des Testszenarios sind 9 Vorkommen solcher Produktszenarien als Entität gekennzeichnet, welche von der Anwendung nicht erkannt werden.

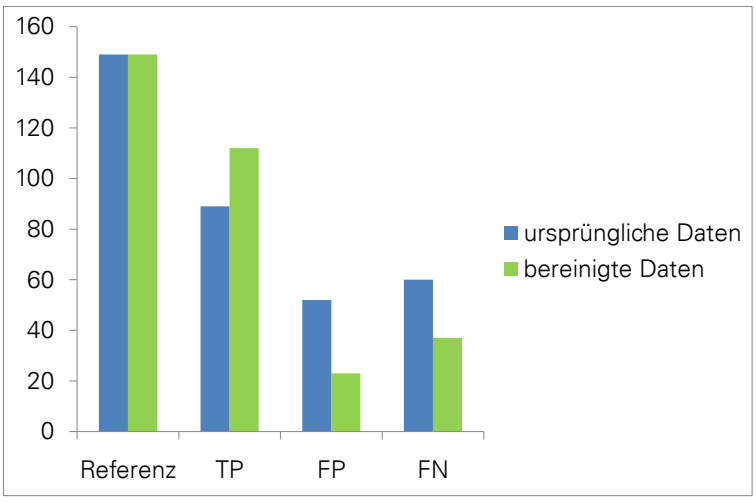


Abbildung 6.2: Bereinigtes Testergebnis

Das Ergebnis aus Abschnitt 6.3.1 wird um die zuvor genannten Werte bereinigt. Abbildung 6.2 zeigt eine Gegenüberstellung von ursprünglichen und bereinigten Daten. Auf Basis dieser Werte lassen sich auch Precision (Gleichung 6.3) und Recall (Gleichung 6.4) neu bestimmen.

$$Precision = \frac{TP}{TP + FP} = \frac{112}{135} = 0,8296 = 82,96\% \tag{6.3}$$

$$Recall = \frac{TP}{TP + FN} = \frac{112}{149} = 0,7517 = 75,17\% \tag{6.4}$$

Somit entsteht eine neue Sicht auf die Ergebnisse dieser Arbeit. Mehr als 50% der negativen Resultate (FP und FN) sind auf qualitative Eigenschaften der zugrunde liegenden Daten zurückzuführen. Zieht man diesen negativen Einfluss bei der Berechnung ab, verbessert sich der Recall um mehr als 15% und die Precision steigt um fast 20%. Dies wird zumindest als starke Annäherung an die Anforderungen gewertet.

6.3.3 Erweiterungsvorschläge

Die Ursachenanalyse des vorherigen Abschnitts hat gezeigt, dass die Resultate des Testszenarios negativ durch die Charakteristiken der strukturierten Daten beeinflusst wurden. Da es sich um Daten eines reellen Anwendungsszenarios handelt, ist es wahrscheinlich, dass in anderen praktischen Anwendungsfällen Daten mit ähnlicher Qualität vorliegen. Daher ist es nicht ausreichend, negative Resultate des Systems mit Hinweis auf die schlechte Datenqualität zu begründen. Vielmehr ist es wichtig, auf solche Erkenntnisse eingehen zu können.
Es wird nur selten möglich sein, strukturierte Daten zu verändern. Probleme, die aufgrund wirklicher *Fehler* auftreten, lassen sich daher nur schwer beheben. Der Stärke des SAP Research UIM-Frameworks liegt in seiner algebraischen Struktur. Darum ist es möglich, Probleme wie Inkonsistenzen oder Unvollständigkeiten, die im letzten Abschnitt beschrieben wurden, auszugleichen wenngleich sie nicht zu beheben sind.
An dieser Stellen werden Vorschläge unterbreitet, wie dies ermöglicht werden kann. Bei der Implementierung des *Entity Identifier* in Abschnitt 5.4.2 wurde dargestellt, dass das Level der Konsolidierung konfigurierbar ist, um die Implementierung zukünftiger Konsolidierungsregeln zu ermöglichen. Auf diese Art und Weise kann gezielt auf spezifische Eigenschaften externer Daten eingegangen werden, die spezifische Konsolidierungsregeln erfordern. Für das Testszenario wird eine einfache Regeln benötigt, die eine große Wirkung hat. So lautet die spezifische Konsolidierungsregel für das Problem mit der Abkürzung *SAP*:

> Entferne alle Kandidaten deren Inhalt der Zeichenkette `<SAP>` entspricht und die als Produktentität gekennzeichnet wurden.

Damit würden auch die Kandidaten entfernt werden, die korrekterweise als Entität *service access point* erkannt wurden, jedoch wird dies wahrscheinlich nur für einem kleinen Bruchteil der Fälle gelten. In diesem Fall erhöht dies die Precision um mehr als 7%.
Eine andere Erweiterung betrifft die Zusammenführung von Annotationen nach der Ausführung eines Annotators. Im Beispiel des Testszenarios würde dies bedeuten, zunächst in den Nominalphrasen nach Zeichenketten der Form `<[A..Z]*[-[A..Z]*]*>` zu suchen, diese an den Bindestrichen zu teilen und in den Teilphrasen nach Entitäten zu suchen. Nach dem Datenabgleich werden die Teilphrasen wieder zusammengeführt und es wird überprüft ob sich daraus eine Kombination der Form `<Komponente[-Komponente]*>` bilden lässt. In diesem Fall würde es sich um ein Produktszenario handeln, wie es im vorherigen Abschnitt beschrieben wird. Damit ließe sich der Recall um 6% erhöhen.
Aus den genannten Erweiterungsvorschlägen geht hervor, dass sowohl das Framework als auch das Ergebnis dieser Arbeit Potential für weitere Entwicklungen haben. Wenngleich an dieser Stelle keine konkrete Konzeption vorgenommen werden kann, so wird jedoch deutlich dass Möglichkeiten für derartige Entwicklungen offen stehen.

6.4 Belastungstest

Die Anwendung des Testszenarios aus Abschnitt 6.2 wird bewußt mit einer kleinen Menge unstrukturierter Daten durchgeführt, um eine manuelle Messung der Qualität und die Erzeugung von Referenzdaten zu erleichtern. Aus diesem Grund eignet sich das Testszenario nicht als Belastungstest zur Bestimmung der Performanz. Zu diesem Zweck wird ein zweiter Extraktionsplan ausgeführt, der hier sehr kurz beschrieben wird.
Im Wesentlichen entspricht der Extraktionsplan für den Belastungstest dem des Testszenarios. Als unstrukturierte Daten werden jedoch Inhalte von RSS-Feeds aus dem SDN-Forum verwendet. Auf diesen Inhalten wird direkt die lexikalische Filterung und Phrasenbildung vorgenommen. Anschließend erfolgt die Identifikation von Entitäten.
Die Verwendung abweichender Daten hat folgenden Grund. Die RSS-Feeds bieten im Prinzip die Rohdaten, die in das Forum gelangen. Auf diese Art und Weise lässt sich feststellen, wieviel Inhalt pro Zeiteinheit im gesamten Forum erzeugt wird. Dazu lässt sich die Leistung der Anwendung gegenüber stellen. In Abschnitt 2.2 wurde bereits beschrieben, dass täglich zwischen 3000 und 5000 Beiträge in das Forum eingestellt werden. Ein Ziel im Anschluss an diese Arbeit ist es, die Anwendung permanent auf den erzeugten Daten auszuführen. Der Belastungstest wird daher mit 50'000 Forumbeiträgen ausgeführt, welche zuvor mit einem RSS-Reader in einer Datenbank gespeichert wurden. Der Umfang entspricht dem zehnfachen der Datenmenge die maximal täglich produziert wird. Damit lassen sich verlässliche Werte für die Performanz des hier entwickelten Ansatzes bestimmen. Die Ergebnisse werden im anschließenden Abschnitt vorgestellt.

6.5 Auswertung des Belastungstests

Zum Abschluss dieses Kapitels wird der Belastungstest ausgewertet. Dazu werden die Ergebnisse dargestellt und anschließend kurz bewertet werden.
Das Gesamtergebnis des Belastungstests wird in Tabelle 6.4 dargestellt. Es wurde die Gesamtdauer des Tests gemessen. Die restlichen Werte wurden errechnet. In Tabelle 6.5 werden vier stichprobenartige Detailmessungen für die Laufzeiten der einzelnen Annotatoren dargestellt. Dazu wurden an jedem Annotator mehrere Zeitmessungen (M1-M4) für jeweils 50 Forumbeiträge vorgenommen. Die Messungen können als repräsentativ angesehen werden, da sich die Gesamtzeiten ungefähr mit dem errechneten Durchschnittswert pro Beitrag aus Tabelle 6.4 decken.

Anzahl der Forumbeiträge	50'000
Gesamtdauer des Tests (hh:mm:ss)	34:08:15
Zeitaufwand pro Beitrag	2,46 Sekunden
Datendurchsatz pro Tag	> 35'000 Beiträge

Tabelle 6.4: Ergebnisse des Belastungstests

Annotator	M1	M2	M3	M4	Durchschnitt pro Beitrag
Importer	36	31	35	29	0,655
Noun Phrases	77	79	82	89	1,635
Entity Identification	7	6	5	6	0,12
Summe	120	116	122	124	2,41

Tabelle 6.5: Laufzeit pro Annotator (s)

Wie schon in der Anforderungsanalyse dargestellt, wird der Performanz der Anwendung nicht die höchste Priorität beigemessen. Ziel ist es, die Datenmenge von ungefähr 5'000 Forumbeiträgen pro Tag durch das System verarbeiten zu können. In Tabelle 6.4 wird dargestellt, dass es möglich ist, die 7-fache Menge an Daten zu verarbeiten. Die Laufzeiten der einzelnen Annotatoren aus Tabelle 6.5 machen deutlich, dass der Hauptanteil der Verarbeitungszeit bei der Dokumentenvorverarbeitung, also bei der Generierung der Nominalphrasen aus dem Dokument heraus entsteht. Der Aufwand ist auf die Stanford-NLP-Bibliothek zurückzuführen, welche für die Identifikation der Wortarten verwendet wird. Sollte zukünftig zum Beispiel aufgrund steigender Datenmengen Optimierungsbedarf entstehen, muss bei der erneuten Auswahl einer NLP-Bibliothek auch die Performanz betrachtet werden. Für die Anwendung in der vorliegenden Arbeit besteht dieser Bedarf nicht.

6.6 Zusammenfassung

In diesem Kapitel werden die Ergebnisse der vorliegenden Arbeit mit den anfänglich in Kapitel 2 erarbeiteten Anforderungen abgeglichen. Der überwiegende Teil gilt den qualitativen Anforderungen. Die funktionalen Anforderungen konnten in einer kurzen Gegenüberstellung als erbracht gewertet werden. Zur Auswertung der qualitativen Anforderungen musste zunächst ein umfangreiches Testszenario definiert werden. Für dieses Testszenario wurde ein Referenzdatensatz erstellt und ein Extraktionsplan aufgestellt, welcher von dem hier entwickelten System ausgeführt wurde.
Die Auswertung der Ergebnisse brachte zunächst zu Tage, dass die Anforderungen nicht in vollem Umfang erfüllt werden konnten. Eine detaillierte Ursachenanalyse führt jedoch zu dem Schluss, dass unter anderem aufgrund der vorliegenden strukturierten Daten eine Reihe von Problemen verursacht wurden, die ein vollständig korrektes Ergebnis nicht möglich machen. Eine entsprechende Bereinigung der Testergebnisse konnte die durchaus guten Resultate dieser Arbeit hervorbringen. Desweiteren konnte aufgezeigt werden, wie zukünftige Entwicklungen die Resultate der Anwendung verbessern können.
Abschließend wurde für die Performanzanalyse ein Belastungstest durchgeführt, welcher auf einer leicht abweichenden Testkonfiguration basiert. Dabei wurde ersichtlich, dass das System einen um Faktor 7 höheren Datendurchsatz ermöglicht, als es derzeit erforderlich ist.

Kapitel 7

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde ein System zur Identifikation von Entitäten in Dokumenten mit Hilfe strukturierter Daten entwickelt. Eingangs der Arbeit wurden Anwendungsfälle sowie die strukturierten und unstrukturierten Daten analysiert. Die unstrukturierten Daten bestehen aus Forumbeiträgen des SAP Developer Network. In diesen Beiträgen werden Entitäten einer SAP-internen Datenbank identifiziert. Die Datenbank enthält neben Produktdaten, die in einzelne Komponenten unterteilt werden, auch eine umfassende Terminologie des SAP-Umfelds.

Ausgehend von den vorliegenden Daten und weiteren Voraussetzungen wurden in Kapitel 2 die Anforderungen für das zu entwickelnde System definiert. Die funktionalen Anforderungen beziehen sich in erster Linie auf die nahtlose Integration der Konzepte in das SAP Research UIM-Framework. Qualitativ wurde die Forderung nach einer möglichst hohen Genauigkeit bei der Identifikation von Entitäten hervorgehoben.

Im Anschluss an die Anforderungsanalyse wurden in Kapitel 3 der Stand der Technik und verwandte Arbeiten diskutiert. Dazu wurde zunächst das Themengebiet der Arbeit auf Basis der Anforderungen auf den Bereich der Named Entity Recognition eingegrenzt. Die darauf folgende Analyse verwandter Arbeiten machte deutlich, dass sich die Zielsetzungen der vorgestellten Ansätze meist in kleinen aber entscheidenden Punkten von den Anforderungen dieser Arbeit unterscheiden. Daraus ließ sich eine genaue Abgrenzung zu diesen Ansätzen ableiten.

Nachdem die Anforderungen definiert waren und eine Abgrenzung zu verwandten Arbeiten vorgenommen wurde, konnte in Kapitel 4 eine Systemarchitektur zur Identifikation von Entitäten in unstrukturierten Daten entworfen werden. Diese wurde in drei Komponenten untergliedert. Die Dokumentenvorverarbeitung dient der syntaktischen Zerlegung und Filterung unstrukturierter Dokumente mit dem Ziel die Suchmenge für Entitäten zu reduzieren. Die Kontextanalyse dient der Integration strukturierter Daten und ist für den Abgleich von strukturierten und unstrukturierten Daten zuständig. Die Konsolidierung führt die Ergebnisse der Dokumentenvorverarbeitung und der Kontextanalyse zusammen und filtert die plausibelsten Entitäten anhand von Konsolidierungsregeln aus der Menge aller möglichen Entitäten für ein Textsegment heraus.

Um die Realisierbarkeit der Konzeption unter Beweis zu stellen wurde eine prototypische Implementierung vorgenommen und in Kapitel 5 beschrieben. Eine wesentliche Anforderung dieser Arbeit bestand darin, das SAP Research UIM-Framework um Operatoren zu erweitern, die eine Identifikation von Entitäten auf Basis strukturierter Daten ermöglichen. Bei der Implementierung wurden daher die Konzepte der Systemarchitektur dieser Arbeit auf die Prinzipien des Frameworks übertragen und entsprechend umgesetzt. Dabei wurde die Architektur des Frameworks um eine Schicht zur Integration strukturierter Daten erweitert

und mit den geforderten Operatoren ausgestattet, welche mit den Datenintegrationskomponenten interagieren.
Zum Abschluss dieser Arbeit wurde in Kapitel 6 eine Evaluation auf Basis der funktionalen und qualitativen Anforderungen vorgenommen. Während die funktionalen Anforderungen durch die Implementierung als erfolgreich erfüllt bewertet werden konnten, wurden die qualitativen Ergebnisse anhand von Messungen eines Testdurchlaufs bewertet. Dazu wurde ein umfangreiches Testszenario konfiguriert und ein zugehöriger Referenzdatensatz erstellt. Die Auswertung der Ergebnisse zeigte zunächst eine Diskrepanz zwischen Anforderungen und Resultaten auf. Eine Ursachenanalyse hat jedoch ergeben, dass die Resultate insbesondere durch die verwendeten strukturierten Daten negativ beeinflusst wurden. Gemessen an diesem Fakt konnten die Resultate der vorliegenden Arbeit als positiv und die Anforderungen als erfolgreich erfüllt bewertet werden.
Mit dem Einsatz des hier entwickelten Systems lassen sich nun Aussagen über Entitäten strukturierter Daten in unstrukturierten Dokumenten treffen. Dadurch können Beziehungen über die Grenzen von Datenstrukturen hinweg bestimmt werden. Ein Anwendungsgebiet ist zum Beispiel die Verknüpfung thematisch ähnlicher Dokumente nach dem Prinzip *Frage-Antwort* oder zur einfachen Gruppierung. Ein weiteres Anwendungsgebiet ist die Aggregation von Informationen über enthaltene Entitäten in unstrukturierten Daten zu Analysezwecken. Ein großes Thema im Bereich ist Informationsgewinnung ist nach wie vor die Suche. Die Ergebnisse dieser Arbeit können verwendet werden, um Suchmaschinen zu entwickeln, die nicht auf der Indizierung von Schlagwörtern aus Texten basieren, sondern die Informationen über enthaltene Entitäten verwenden.
In dieser Arbeit wurde das Prinzip verfolgt Entitäten in kleinen, zuvor extrahierten Textsegmenten zu identifizieren. Dabei wurde das Wissen über benachbarte Segmente außer Acht gelassen. Die Einbeziehung dieses Wissens führt zu einem wesentlich höheren Konsolidierungsaufwand, verspricht aber für zukünftige Arbeiten, die sich diesem Thema widmen eine noch bessere Präzision bei der Bestimmung relevanter Entitäten in unstrukturierten Daten.

Anhang A

Abkürzungsverzeichnis

BI Business Intelligence

DBLP Digital Bibliography & Library Project

DWH Data Warehouse

FN False Negative

FP False Positive

HMM Hidden Markov Model

IDF Inverse Document Frequency

IE Informationsextraktion

IMDB Internet Movie Database

MEM Maximum Entropy Method

NER Named Entity Recognition

NERC Named Entity Recognition and Classification

NLP Natural Language Processing

OLAP Online Analytical Processing

OLIF Open Lexicon Interchange Format

SDN SAP Developer Network

TF Term Frequency

TN True Negative

TP True Positive

UIM Unstructured Information Management

Anhang B

Glossar

Anker-Tag HTML-Tag (<a> ...), für die Auszeichnung von Hyperlinks in HTML-Dokumenten verwendet

Business Intelligence Verfahren und Prozesse zur systematischen Analyse (Sammlung, Auswertung, Darstellung) von Unternehmensdaten in elektronischer Form

Citeseer Datenbank und Suchmaschine für wissenschaftliche Dokumente aus dem Bereich der Informatik

Data Warehouse zentrale Datensammlung, deren Inhalt sich aus Daten unterschiedlicher Quellen zusammensetzt

DBWorld Email-Verteiler, der eingetragene Mitglieder mit Informationen zu anstehenden Konferenzen, Seminaren und Fachtagungen aus dem Datenbankumfeld versorgt

Digital Bibliography & Library Project dt.: Digitales Bibliografie- und Bibliotheksprojekt, etablierte digitale Bibliografie von Publikationen aus dem Bereich der Informatik, führt mehr als 1Mio Artikel, siehe http://dblp.uni-trier.de

Disambiguierung Auflösung von Mehrdeutigkeiten

E-Maß Gewichtetes Mittel aus Precision und Recall, auch als Effektivitätsmaß bezeichnet, welches durch frei wählbaren Parameter einen der beiden Werte bevorzugt

Entropie bezeichnet die Informationsdichte bzw. den Informationsgehalt einer Quelle pro Zeichen

F-Maß Harmonisches Mittel aus Precision und Recall, auch als F_1-Maß bezeichnet, maximaler Wert bezeichnet optimalen Kompromiss aus beiden Werten

Factory Method Softwareentwurfsmuster aus der Kategorie der Erzeugungsmuster - beschreibt, wie ein Objekt durch Aufruf einer Methode anstatt durch direkten Aufruf eines Konstruktors erzeugt wird

False Negative dt.: falsch negativ, Klassifizierung eines Suchresultats, bezeichnet ein Resultat, welches im Sinne der Zielstellung relevant ist, jedoch *nicht* durch den Suchalgorithmus gefunden wurde

False Positive dt.: falsch positiv, Klassifizierung eines Suchresultats, bezeichnet ein Resultat, welches im Sinne der Zielstellung *nicht* relevant ist, jedoch durch den Suchalgorithmus gefunden wurde

Hidden Markov Model dt.: Verborgenes Markov-Modell, stochastisches Modell, dass aus einer Zeichensequenz auf *verborgene* Zustände schließen lässt

Internet Movie Database dt.: Internet-Filmdatenbank, Datenbank mit Informationen über alle möglichen Arten von Filmproduktionen und deren Mitwirkenden, enthält über 1Mio Einträge

Inverse Document Frequency dt.: inverse Dokumenthäufigkeit, beschreibt die allgemeine Signifikanz eines Element durch Invertierung der Anzahl seiner Vorkommen in der Gesamtmenge aller Dokumente und anschließender Logarithmierung

Konfidenzwert numerischer Wert für die Signifikanz des Ergebnisses einer Berechnung oder Ausführung eines Algorithmus, *Vertrauen in das Ergebnis*

Maximum Entropy Method dt.: Methode der maximalen Entropie, stochastisches Verfahren (aus der Signalverarbeitung), welches auf Basis von a-priori-Wahrscheinlichkeiten versucht, Signale als problemspezifische Informationen zu identifizieren

Named Entity Recognition Teilgebiet der Informationsextraktion, welches sich mit der Lokalisierung und Klassifizierung atomarer Elemente in unstrukturierten Daten beschäftigt

Named Entity Recognition and Classification siehe *Named Entity Recognition*

Natural Language Processing dt.: Verarbeitung natürlicher Sprache, Teilgebiet der Computerlinguistik, welches sich mit natürlichen Eigenschaften von Sprache beschäftigt, dient unter anderem zur Erkennung der syntaktischen Struktur von Texten

Online Analytical Processing Datenhaltungskonzept, welches komplexe Geschäftsanalysen ermöglicht, die vom Endanwender in einer mehrdimensionalen Umgebung durch Werkzeug- und IT-Unterstützung vorgenommen werden können

Open Lexicon Interchange Format XML-basierter Standard für den Austausch von lexikalischen oder terminologischen Daten, siehe http://www.olif.net

Pareto-Optimum Kriterium für eine Bewertung von Zuständen und Veränderungen, die mehrere Anforderungen betreffen. Ein Zustand ist pareto-optimal, wenn keine Veränderung möglich ist, die mindestens eine Anforderung besser erfüllt ohne sich zugleich in Bezug auf eine Andere zu verschlechtern

Parser Programm zur Zerlegung und Analyse einer Eingabe mit dem Ziel der Überführung in eine (neue) Struktur

Part-of-Speech Tagger dt.: Wortart-Markierer, Programm zur Identifikation von Wortarten, weist jedem Wort eines Textes seine Wortart zu

Precision Genauigkeit eines Rechercheverfahrens - bezeichnet den Anteil gefundener, relevanter Inhalte im Verhältnis zur Gesamtmenge gefundener Inhalte

Question Answering Systeme zur automatischen intelligenten Beantwortung von Fragen

Recall Vollständigkeit eines Rechercheverfahrens - bezeichnet den Anteil gefundener, relevanter Inhalte im Verhältnis zur Gesamtmenge relevanter Inhalte

SAP Developer Network Online Gemeinschaft für den Austausch zu SAP bezogenen Technologien

SAP Term SAP-internes Glossar, welches nahezu die gesamte Terminologie abdeckt, die mit dem SAP-Umfeld insbesondere den SAP-Produkten in Verbindung steht

Stoppwort Wort, welches im Sinne der Informationsgewinnung keine relevanten Informationen zur Semantik eines Dokuments beiträgt, Stoppwörter sind Präpositionen, Konjunktionen und Artikel

Term Frequency dt.: Häufigkeit eines Terms im Vergleich zur Maximalhäufigkeit über alle Terme innerhalb eines Dokuments

TF-IDF Gewichtung eines Terms anhand der Faktoren TF und IDF, TF bildet die Intra-Dokument-Charakterisierung (siehe Term Frequency), IDF ist die Inter-Dokument-Charakterisierung (siehe Inverse Document Frequency)

Top-k Suche Suche nach den k besten Treffern

TREX SAP NetWeaver Search and Classification (TREX), bietet SAP-Anwendungen Services für Suche und Klassifikation großer Dokumentsammlungen wie auch für die Suche und Aggregation von Business Objekten

True Negative dt.: richtig negativ, Klassifizierung eines Suchresultats, bezeichnet ein Resultat, welches im Sinne der Zielstellung *nicht* relevant ist und *nicht* durch den Suchalgorithmus gefunden wurde

True Positive dt.: richtig positiv, Klassifizierung eines Suchresultats, bezeichnet ein Resultat, welches im Sinne der Zielstellung relevant ist und durch den Suchalgorithmus gefunden wurde

Unstructured Information Management Forschungsgebiet welches sich mit der Analyse von unstrukturierten Daten beschäftigt

Visitor Softwareentwurfsmuster aus der Kategorie der Verhaltensmuster - dient zum Kapseln von Operationen, die auf Elementen einer Objektstruktur ausgeführt werden

Web of Entities globale virtuelle Ablage und Verwaltung von Informationen und Wissen über beliebige Entitäten

Webcrawler Computerprogramm zur Durchsuchung des World Wide Web, ausgehend von einer Startseite verfolgt es alle enthaltenen Verknüpfungen im Dokument, kann zur Speicherung von Web-Dokumenten genutzt werden

WordNet weltweit größte englischsprachige lexikalische Datenbank, bildet semantische Beziehungen zwischen Begriffen ab

Anhang C

Abbildungsverzeichnis

Anhang D

Tabellenverzeichnis

Anhang E

Literaturverzeichnis

[Baeza-Yates und Ribeiro-Neto 1999] BAEZA-YATES, Ricardo ; RIBEIRO-NETO, Berthier: *ACM Press books*. Bd. 1. print.: *Modern information retrieval*. England : Addison-Wesley Longman, 1999. – 513 S. – ISBN 0–201–39829–X

[Barczyñski et al. 2007] BARCZYÑSKI, Wojciech M. ; BRAUER, Falk ; LÖSER, Alexander: Algebraic information extraction of enterprise data - Methodology, operators and storage model / SAP Research. 2007. – Forschungsbericht

[Baronchelli et al. 2004] BARONCHELLI, Andrea ; CAGLIOTI, Emanuele ; LORETO, Vittorio ; PIZZI, E.: Dictionary based methods for information extraction. In: *The Computing Research Repository* (2004)

[Bhide et al. 2007] BHIDE, Manish A. ; GUPTA, Ajay ; GUPTA, Rahul ; ROY, Prasan ; MOHANIA, Mukesh K. ; ICHHAPORIA, Zenita: LIPTUS: associating structured and unstructured information in a banking environment. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*. New York, NY, USA : ACM, 2007. – ISBN 978–1–59593–686–8, S. 915–924

[Blumberg und Atre 2003] BLUMBERG, Robert ; ATRE, Shaku: The Problem with Unstructured Data. In: *Data Management Review Magazine* (2003), Nr. 2

[Chakaravarthy et al. 2006] CHAKARAVARTHY, Venkatesan T. ; GUPTA, Himanshu ; ROY, Prasan ; MOHANIA, Mukesh K.: Efficiently Linking Text Documents with Relevant Structured Information. In: *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, 2006, 667-678

[Chandel et al. 2006] CHANDEL, Amit ; NAGESH, P. C. ; SARAWAGI, Sunita: Efficient Batch Top-k Search for Dictionary-based Entity Recognition. In: *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, 2006

[Cheng et al. 2003] CHENG, Ching K. ; PAN, Xiaoshan ; KURFESS, Franz J.: Ontology-Based Semantic Classification of Unstructured Documents. In: *Adaptive Multimedia Retrieval: First International Workshop, AMR 2003, Hamburg, Germany, September 15-16, 2003, Revised Selected and Invited Papers*, 2003, 120-131

[Cohen et al. 2003] COHEN, William W. ; RAVIKUMAR, Pradeep ; FIENBERG, Stephen E.: A Comparison of String Distance Metrics for Name-Matching Tasks. In: KAMBHAMPATI,

Subbarao (Hrsg.) ; KNOBLOCK, Craig A. (Hrsg.): *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 9-10, 2003, Acapulco, Mexico*, 2003, 73-78

[Cohen und Sarawagi 2004] COHEN, William W. ; SARAWAGI, Sunita: Exploiting dictionaries in named entity extraction: combining semi-Markov extraction processes and data integration methods. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, 2004, S. 89–98

[Fink 2003] FINK, Gernot A.: *Mustererkennung mit Markov-Modellen: Theorie-Praxis-Anwendungsgebiete*. 1. Auflage. Vieweg+Teubner, 2003. – 233 S.

[Hassell et al. 2006] HASSELL, Joseph ; ALEMAN-MEZA, Boanerges ; ARPINAR, I. B.: Ontology-Driven Automatic Entity Disambiguation in Unstructured Text. In: *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, 2006, S. 44–57

[Mansuri und Sarawagi 2006] MANSURI, Imran R. ; SARAWAGI, Sunita: Integrating Unstructured Data into Relational Databases. In: *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, 2006, 29

[Nadeau und Sekine 2007] NADEAU, David ; SEKINE, Satoshi: A Survey of Named Entity Recognition and Classification. In: *Lingvisticae Investigationes* 30 (2007)

[Ramakrishnan 2008] RAMAKRISHNAN, Raghu: Web Data Management: Powering the New Web. In: *Keynote of the Australasian Database Conference*, 2008

[Ramakrishnan und Tomkins 2007] RAMAKRISHNAN, Raghu ; TOMKINS, Andrew: Toward a PeopleWeb. In: *Computer - Innovative Technology for Computer Professionals* 40 (2007), Nr. 8, S. 63–72. – ISSN 0018–9162

[Sarawagi und Cohen 2005] SARAWAGI, Sunita ; COHEN, William W.: Semi-Markov Conditional Random Fields for Information Extraction. In: *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, 2005

[Sleator und Temperley 1993] SLEATOR, D. D. ; TEMPERLEY, D.: Parsing English with a link grammar. In: *Third International Workshop on Parsing Technologies*, 1993

[Wu 1997] WU, Nailong: *The Maximum Entropy Method (Data and Knowledge in a Changing World)*. 1. Auflage. Berlin : Springer, 1997. – 327 S.

Eigenständigkeitserklärung

Hiermit erkläre ich, Marcus Schramm, die vorliegende Diplomarbeit zum Thema

Informationsextraktion auf Basis strukturierter Daten

eigenständig und ausschließlich unter Verwendung der im Quellenverzeichnis aufgeführten Literatur- und sonstigen Informationsquellen verfasst zu haben.

Dresden im Mai 2008