Extraktion und Identifikation von Entitäten in Textdaten im Umfeld der Enterprise Search

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)
in der Wissenschaftsdisziplin Informationssysteme

vorgelegt an der
Universität Potsdam
Mathematisch-Naturwissenschaftliche Fakultät

eingereicht von

Dipl.-Medieninf. Falk Brauer
geboren am 26. September 1981
in Dresden

Betreuender Hochschullehrer:
Prof. Dr. rer. nat. Felix Naumann,
Universität Potsdam, Hasso Plattner Institut

Dresden, im Oktober 2010

Online veröffentlicht auf dem
Publikationsserver der Universität Potsdam:
URL http://opus.kobv.de/ubp/volltexte/2011/5140/
URN urn:nbn:de:kobv:517-opus-51409
http://nbn-resolving.org/urn:nbn:de:kobv:517-opus-51409

Inhaltsverzeichnis

Danksagung

Hiermit möchte ich allen Menschen meinen Dank aussprechen, welche die vorliegende Arbeit ermöglicht haben.

Zuallererst danke ich Prof. Dr. Felix Naumann für die Begleitung meiner Arbeit. Er stand mir stets engagiert mit Rat und Tat zur Seite, obwohl seine Zeit oft knapp bemessen war. Besonders das Interesse an den von mir entwickelten Lösungsansätzen sowie seine Geduld, sich in diese hineinzudenken, verdienen Erwähnung. Er und Prof. Dr. Ulf Leser lehrten mich Argumente und Lösungen wissenschaftlich aufzubereiten und dabei das Wesentliche im Blick zu behalten. Allen anderen Gutachtern möchte ich für ihre Zeit danken, welche sie in das Studium meiner Arbeit investieren.

Weiterhin gilt mein Dank dem Unternehmen SAP, dessen Doktorandenprogramm die vorliegende Arbeit erst ermöglichte. Viele meiner Kollegen leisteten einen wertvollen Beitrag zu dieser Arbeit. Meinen Dank bekunden möchte ich zunächst Dr. habil. Gregor Hackenbroich, weiterhin meinen ehemaligen Kollegen Dr. Hong-Hai Do und Dr. Alexander Löser, meinen aktuellen Kollegen Dr. Adrian Mocan, Wojciech Barczynski, Marcus Schramm und Robert Rieger sowie den ehemaligen Werksstudenten Michael Huber und Felix Förster. Mit Diskussionen, Ratschlägen und der aktiven Auseinandersetzung mit meinen Themen trugen sie zum Erfolg der Arbeit bei.

Ferner möchte ich allen danken, die an meiner Ausbildung an der TU Dresden beteiligt waren. Insbesondere Prof. Dr. Andreas Schill bin ich zu Dank verpflichtet, an dessen Lehrstuhl ich unter anderem meine Diplomarbeit schrieb. Weiterhin ist mir Prof. Dr. Bernhard Ganter, mein damaliger Mathematik-Professor, in sehr positiver Erinnerung geblieben. Er lehrte mich, von konkreten Problemstellungen zu abstrahieren und Probleme in ihrer Gesamtheit zu betrachten.

Stets Hilfe, Unterstützung und offene Ohren boten mir auch meine Familie und Freunde, denen ich meinen herzlichsten Dank schulde. Besondere Erwähnung gilt hierbei Sabine, die viel Verständnis für den Zeitmangel in unserem Privatleben aufbrachte, den eine solche Arbeit mit sich bringt. Und auch meiner 6 Monate alten Tochter gilt mein Dank, denn sie wusste und weiß mich immer wieder aufzuheitern.

Zusammenfassung

Die automatische *Informationsextraktion (IE)* aus unstrukturierten Texten ermöglicht völlig neue Wege, auf relevante Informationen zuzugreifen und deren Inhalte zu analysieren, die weit über bisherige Verfahren zur Stichwort-basierten Dokumentsuche hinausgehen. Die Entwicklung von Programmen zur Extraktion von maschinenlesbaren Daten aus Texten erfordert jedoch nach wie vor die Entwicklung von domänenspezifischen Extraktionsprogrammen.

Insbesondere im Bereich der *Enterprise Search* (der Informationssuche im Unternehmensumfeld), in dem eine große Menge von heterogenen Dokumenttypen existiert, ist es oft notwendig, ad-hoc Programmmodule zur Extraktion von geschäftsrelevanten Entitäten zu entwickeln, die mit generischen Modulen in monolithischen IE-Systemen kombiniert werden. Dieser Umstand ist insbesondere kritisch, da potenziell für jeden einzelnen Anwendungsfall ein von Grund auf neues IE-System entwickelt werden muss.

Die vorliegende Dissertation untersucht die effiziente Entwicklung und Ausführung von IE-Systemen im Kontext der Enterprise Search und effektive Methoden zur Ausnutzung bekannter strukturierter Daten im Unternehmenskontext für die Extraktion und Identifikation von geschäftsrelevanten Entitäten in Dokumenten. Grundlage der Arbeit ist eine neuartige Plattform zur Komposition von IE-Systemen auf Basis der Beschreibung des Datenflusses zwischen generischen und anwendungsspezifischen IE-Modulen. Die Plattform unterstützt insbesondere die Entwicklung und Wiederverwendung von generischen IE-Modulen und zeichnet sich durch eine höhere Flexibilität und Ausdrucksmächtigkeit im Vergleich zu vorherigen Methoden aus.

Ein in der Dissertation entwickeltes Verfahren zur Dokumentverarbeitung interpretiert den Datenaustausch zwischen IE-Modulen als Datenströme und ermöglicht damit die weitgehende Parallelisierung von einzelnen Modulen. Die autonome Ausführung der Module führt zu einer wesentlichen Beschleunigung der Verarbeitung von Einzeldokumenten und verbesserten Antwortzeiten, z. B. für Extraktionsdienste. Bisherige Ansätze untersuchen lediglich die Steigerung des durchschnittlichen Dokumentendurchsatzes durch verteilte Ausführung von Instanzen eines IE-Systems.

Die Informationsextraktion im Kontext der Enterprise Search unterscheidet sich z. B. von der Extraktion aus dem *World Wide Web (WWW)* dadurch, dass in der Regel strukturierte Referenzdaten, z. B. in Form von Unternehmensdatenbanken oder Terminologien zur Verfügung stehen, die oft auch die Beziehungen von Entitäten beschreiben. Entitäten im Unternehmensumfeld haben weiterhin bestimmte Charakteristiken: Eine Klasse von relevanten Entitäten folgt bestimmten Bildungsvorschriften, die nicht immer bekannt sind, auf die aber mit Hilfe von bekannten Beispielentitäten geschlossen werden kann, so dass unbekannte Entitäten desselben Typs extrahiert werden können. Die Bezeichner der anderen Klasse von Entitäten haben eher umschreibenden Charakter. Die korrespondierenden Umschreibungen in Texten können variieren, wodurch eine Identifikation derartiger Entitäten oft erschwert wird.

Zur effizienteren Entwicklung von IE-Systemen wird in der Dissertation ein Verfahren untersucht, das alleine anhand von Beispielentitäten effektive Reguläre Ausdrücke zur Extraktion von unbekannten Entitäten erlernt und damit den manuellen Aufwand in derartigen Anwendungsfällen minimiert. Verschiedene Generalisierungs- und Spezialisierungsheuristiken erkennen Muster auf verschiedenen Abstraktionsebenen und schaffen dadurch einen Ausgleich zwischen Genauigkeit und Vollständigkeit bei der Extraktion. Bekannte Regellernverfahren im

Bereich der Informationsextraktion unterstützen die beschriebenen Problemstellungen nicht, sondern benötigen einen (annotierten) Dokumentenkorpus.

Eine Methode zur Identifikation von Entitäten, die durch Graph-strukturierte Referenzdaten vordefiniert sind, wird als dritter Schwerpunkt untersucht. Es werden Verfahren konzipiert, welche über einen exakten Zeichenkettenvergleich zwischen Text und Referenzdatensatz hinausgehen und Teilübereinstimmungen und Beziehungen zwischen Entitäten zur Identifikation und Disambiguierung heranziehen. Das in der Arbeit vorgestellte Verfahren ist bisherigen Ansätzen hinsichtlich der Genauigkeit und Vollständigkeit bei der Identifikation überlegen.

1

Informationsextraktion im Unternehmenskontext

Die vorliegende Arbeit untersucht die Extraktion und Identifikation von Entitäten im Unternehmensumfeld. Im Fokus stehen die effiziente Entwicklung und Ausführung von anwendungsspezifischen Informationsextraktionssystemen und die effektive Ausnutzung von strukturierten Referenzdaten im Unternehmensumfeld zur Extraktion und Identifikation von Entitäten in Dokumenten. Entitäten im Sinne dieser Arbeit sind gemäß [Che76] wie folgt definiert:

Definition 1.0.1 (*Entität*)

> *Eine Entität ist ein individuelles Objekt, das von Interesse für eine bestimmte (Unternehmens-)Anwendung ist. Entitäten unterscheiden sich durch ihre Eigenschaften (Attributtyp-Attributwert-Tupel). Entitäten sind durch qualifizierte Beziehungen miteinander verbunden und sind weiterhin Entitätstypen zugeordnet. Entitätstypen unterscheiden sich durch deren assoziierte Attributtypen.*

Unstrukturierte Daten in Textform repräsentieren nach wie vor einen Großteil des täglichen Datenaufkommens in Unternehmen. Deren Management und Analyse wurden bisher nicht zufriedenstellend gelöst. Es werden neue Zugriffsverfahren benötigt, um effizienter auf relevante Informationen aus Texten zuzugreifen und deren Inhalte zu analysieren, die über bisherige Verfahren zur Stichwort-basierten Dokumentsuche hinausgehen. Die Extraktion von Entitäten aus Text ist eine Schlüsseltechnologie zur Realisierung von neuartigen Zugriffsverfahren. Wir bezeichnen im Folgenden text-zentrische Anwendungen im Unternehmensumfeld als *Enterprise Search*, wobei nicht nur die Suche nach relevanten Inhalten, sondern z. B. auch die statistische Analyse von extrahierten Daten eingeschlossen wird.

Die Entwicklung von Systemen zur Extraktion von maschinenlesbaren Daten aus Texten (die *Informationsextraktion*) setzt Methoden voraus, die Domänenwissen hinsichtlich der Entitäten oder Dokumente in bestimmter Art und Weise ausnutzen, um relevante Daten aus Texten zu extrahieren. In vielen Anwendungsfällen kann Domänenwissen zum Beispiel in Form von Extraktionsregeln formalisiert und durch generische Extraktionssoftware ausgenutzt werden. Für viele Anwendungsfälle sind aber auch speziell entwickelte Extraktionsmethoden notwendig, die häufig mit generischen Methoden kombiniert werden müssen.

Besonders im Unternehmensumfeld, in denen eine große Menge von heterogenen Dokumenttypen existiert, ist es oft notwendig, ad-hoc Programmkomponenten, z. B. zur Extraktion von

Dokumentstrukturen zu entwickeln, die mit generischen Methoden zur Informationsextraktion in monolithischen Informationsextraktion (IE)-Systemen kombiniert werden. Einige wenige in der Literatur beschriebene Ansätze [SDNR07, BFBS10] unterstützen eine Komposition von generischen und anwendungsspezifischen IE-Komponenten und stellen damit ein mächtiges Werkzeug bei der Entwicklung von anwendungsspezifischen IE-Systemen dar.

Eine graphische oder deklarative Komposition von IE-Komponenten verbessert durch die leichtere Wiederverwendung von Komponenten die Effizienz in der Entwicklung von IE-Programmen, erschwert aber die (programmatische) Optimierung von IE-Systemen hinsichtlich der Dokumentverarbeitungszeiten, z. B. durch eine Parallelisierung von Datenverarbeitungsoperationen. Die Parallelisierung von visuell oder deklarativ aus IE-Modulen komponierten IE-Systemen wirft ungelöste Forschungsfragen auf, die wir im Rahmen dieser Arbeit untersuchen.

Zur Entwicklung von anwendungsspezifischen IE-Systemen im Kontext der *Enterprise Search* ist eine Art von Domänenwissen von besonderem Interesse: strukturierte Daten, die Bezeichner von im Unternehmenskontext relevanten Entitäten und deren Beziehungen definieren. Nach der Einschätzung von *Agrawal et al.* [AAB+08] ist die effektive Nutzung von vorhandenen strukturierten Daten zur Analyse von unstrukturierten Daten eine der aktuellen Herausforderungen im Forschungsgebiet der Informationssysteme.

Bezeichner von Entitäten im Unternehmensumfeld (z. B. Produktnamen) folgen oft Mustern, die durch Bildungsvorschriften beschrieben werden können. Nicht immer sind diese Bildungsvorschriften bekannt. Eine Methode zur Rekonstruktion von Bildungsvorschriften aus gegebenen Beispielinstanzen kann den Entwicklungsaufwand für anwendungsspezifische IE-Systeme wesentlich reduzieren und bildet einen zweiten Schwerpunkt der vorliegenden Arbeit.

Die Identifikation von Entitäten, die durch Graph-strukturierte Referenzdaten (relationale Daten, XML, Ontologien) vordefiniert sind, wird als dritter Schwerpunkt untersucht. Im Speziellen untersuchen wir Verfahren, die über einen exakten Zeichenkettenvergleich zwischen Text und Referenzdatensatz hinaus gehen und Teilübereinstimmungen und Beziehungen zwischen Entitäten zur Identifikation und Disambiguierung von Entitäten heranziehen.

Im verbleibenden Kap. 1 führen wir die Problemstellungen ein und motivieren offene Forschungsprobleme. In Kap. 1.1 werden Hintergrund und Problemstellungen, die in dieser Arbeit untersucht werden, eingeführt und der Aufbau der Arbeit beschrieben. Zur weiteren Motivation stellen wir in Kap. 1.2 Anwendungsfälle vor, die im Kontext dieser Arbeit gelöst wurden, und die Relevanz der hier untersuchten Forschungsprobleme untermauern.

1.1 Einleitung

Im Folgenden werden Forschungsgebiete, die im Kontext der vorliegenden Arbeit relevant sind, eingeführt. Im Weiteren werden Charakteristiken von Entitäten im Unternehmensumfeld analysiert und die Beiträge der Arbeit hinsichtlich der Extraktion und Identifikation von Entitäten erläutert.

1.1.1 Einordnung und Hintergrund

Seit den frühen Jahren der Informationstechnologie ist die strukturierte Speicherung von Daten und der gezielte Datenzugriff ein wichtiges Anliegen in Forschung und Industrie [BW64]. Die Entwicklung von Relationalen Datenbank Management Systemen (RDBMS) in den 1970ern (z. B. System R [ABC+76]) verstärkte diesen Trend und führte zu einem in der Industrie etablierten Standard für die Speicherung und den Zugriff auf strukturierte Daten. Neuere (semi-)strukturierte Datenrepräsentation sind Extensible Markup Language (XML) oder Ontologien (z. B. im RDF-Format), die in den letzten Jahren im Unternehmenskontext an Relevanz gewonnen haben.

Eine strukturierte Speicherung von Daten erlaubt eine strukturierte Suche, Abfrage und Aggregation von Daten und wird daher im großem Maße zur Datenorganisation im Unternehmensumfeld genutzt. Trotz strukturierter Speicherung sind Datenquellen in Unternehmen höchst heterogen. Weiterhin werden in Unternehmen nach wie vor große Datenmengen in Textform erhoben, deren Abfrage und Aggregation nicht ohne weiteres möglich ist.

Die Forschungsgebiete, die sich mit Herausforderungen der Integration und Abfrage von heterogenen und zum Teil unstrukturierten Datenbeständen widmen und im Kontext dieser Arbeit eine Rolle spielen, sind die *Informationsintegration*, die *Informationsextraktion* und das *Information Retrieval*. Sie werden im Folgenden eingeführt:

Informationsintegration. Der starke Anstieg von strukturierten Informationsquellen in Unternehmen im Zuge der Digitalisierung von Geschäftsprozessen und die Bemühungen um eine holisistische Sicht über unternehmensinterne Prozesse führte zu Lösungen zur Stammdatenverwaltung, um system- und anwendungsübergreifende Konsistenz sicherzustellen. Die Stammdatenverwaltung erfordert, wie auch andere Anwendungen, eine Integration und Föderation von heterogenen, verteilten (meist strukturierten) Datenbeständen.

Im Forschungsgebiet der Informationsintegration wurde seit den 1990er Jahren eine große Anzahl von Ansätzen zur Integration strukturierter Datenbestände [WVV+01, LN06] entwickelt. Teilgebiete des Forschungsgebiets sind die Abbildung und Integration von Datenschemata [RB01], die Duplikaterkennung [EIV07, LCW07, CKM07] und die Datenfusion [BN08].

Die Arbeiten in diesem Bereich widmen sich im Allgemeinen der Integration von strukturierten Datenquellen und ermöglichen eine (semi-)automatische und/ oder deklarative Integration [BN05] heterogener Datenquellen. Eine nahtlose Integration von unstrukturierten und strukturierten Daten ist ein nahe liegender nächster Schritt in der Informationsintegration, um eine holisistische Sicht über unternehmensinterne Prozesse sicherzustellen.

Informationsextraktion. Die Extraktion von strukturierten Daten mit wohldefinierter Semantik aus Textdaten wurde mit der Reihe der *Message Understanding Conferences (MUC)* in der Forschung populär. Ziel der *Named Entity Recognition*, als Teilgebiet der *Informationsextraktion (IE)*, ist es, Entitätstypen, wie z. B. Personen oder Organisationen aus Texten zu extrahieren. Das Teilgebiet der *Fact Extraction* adressiert die Extraktion von (meist binären) Beziehungen zwischen Entitäten in Texten. In beiden Bereichen der Informationsextraktion wurden verschiedenste Verfahren, die im Allgemeinen in maschinelle Lernverfahren und Regel-basierte Techniken unterschieden werden können, entwickelt (siehe z. B. [Sar08] für einen Überblick).

Die verschiedenen Verfahren zur Informationsextraktion sind häufig speziell für bestimmte Texttypen (z. B. semi-strukturierter Text oder Fließtext) und Entitäts- oder Beziehungstypen konzipiert und nicht ohne weiteres in andere Anwendungsfälle adaptierbar [Gri97]. Ein Ansatz, der eine Komposition von generischen Ansätzen mit anwendungsspezifischen Programmcode ermöglicht, bietet eine gute Basis zur Entwicklung von IE-Systemen für verschiedene Domänen und Anwendungsfälle im Unternehmenskontext und wird daher in dieser Arbeit verfolgt.

Information Retrieval. Der Forschungsbereich des *Information Retrieval* adressiert im Allgemeinen die Suche nach bekannten Informationen, die in der Regel in unstrukturierter Form vorliegen. Die gebräuchlichste Form ist die Schlüsselwort-basierte Dokumentsuche.

Ansätze im Bereich des Information Retrieval entstanden bereits in den 1960er Jahren [Luh59]. Mit dem in den 1990er Jahren aufkommenden *World Wide Web (WWW)* wurde das *Ranking* von Dokumenten entsprechend ihrer Relevanz zu einem wichtigen Forschungsfeld und führte zur Entwicklung von relativ generischen Bewertungsverfahren, wie z. B. *TF-IDF*, *BM25* und Probabilistischen Rankingverfahren [MRS08], aber auch zu Web-spezifischen Verfahren, wie z. B. *PageRank* [Whi08]. Spezielle Verfahren zur Bewertung der Relevanz von Dokumenten im

Unternehmensumfeld wurden in der Literatur bisher nicht beschrieben, so dass mangels Alternativen zumeist generische Dokumentbewertungsverfahren wie *TF-IDF* [RW99] für Probleme in der *Enterprise Search* verwendet werden.

Die vorliegende Arbeit adressiert die Schnittstellen von Informationsintegration, Information Retrieval und Informationsextraktion für Anwendungen im Bereich der *Enterprise Search*.

Enterprise Search. Der Begriff *Enterprise Search* ist (angelehnt an [BBF+09]) in dieser Arbeit wie folgt definiert:

Definition 1.1.1 (*Enterprise Search*)

> *Ziel der* Enterprise Search *ist es, Inhalte effizienter zu finden, überhaupt zu finden oder Inhalte zu neuen Informationen aufzubereiten und zu aggregieren.* Enterprise Search *unterstützt insbesondere Mitarbeitende und Kunden eines Unternehmens, Informationen und Wissensbestände im Unternehmenskontext zu durchsuchen. Die Qualität einer Enterprise-Search-Lösung bemisst sich an der Abdeckung der Quellen, einer Unterstützung zur Formulierung komplexer Anfragen, den Möglichkeiten hinsichtlich der Ergebnisnavigation und insbesondere an der Relevanz der Suchergebnisse, die möglichst unabhängig von der Qualität der Frageformulierung sein sollte.*

Die Spezifika der Dokumentsuche innerhalb von Unternehmen werden durch die meisten Ansätze im Bereich des Information Retrieval nur unzureichend adressiert [Haw04, MM04, BC04], da z. B. der Informationsbedarf im Kontext der *Enterprise Search* im Vergleich zur Suche im WWW häufig spezifischer und die Anzahl der relevanten Dokumente äußerst gering ist [BBF+09].

Die Verwendung von Verfahren aus dem Bereich der Informationsextraktion in der *Enterprise Search* versprechen neben der Verbesserung der Relevanz von Suchergebnissen einen strukturierten Zugriff auf Dokumentinhalte, eine verbesserte Navigation in Suchergebnissen und die (statistische) Analyse von semantisch bedeutsamen Dokumentinhalten. Die Entwicklung von IE-Systemen für spezifische Anwendungsfälle ist jedoch häufig äußerst aufwändig.

Im Unternehmensumfeld besteht im Gegensatz zum WWW jedoch ein gemeinsames Verständnis über die Semantik von Begrifflichkeiten, welche als Domänenwissen z. B. in Unternehmensdatenbanken in strukturierter Form vorliegt. Entitäten, die in den maschinell einfach zu verarbeitenden strukturierten Daten gespeichert sind, treten in ähnlicher Form auch in geschäftsrelevanten Dokumenten auf. Demzufolge verspricht die Informationsintegration zwischen Dokumenten und strukturierten Unternehmensdaten eine effizientere Entwicklung von IE-Systemen für Unternehmensanwendungen.

1.1.2 Strukturierte Referenzdaten in Unternehmen

Innerhalb von Unternehmen sind oft *numerische Daten*, wie z. B. Umsatz- oder Verkaufzahlen, Stückzahlen, Kosten etc. von Bedeutung und werden zumeist in strukturierter Form verwaltet. Strukturiert gespeicherte numerische Daten ohne Kontext zu anderen textuellen Daten werden in dieser Arbeit nicht weiter betrachtet, da sie für die Textanalyse keinen oder nur geringen Mehrwert bieten. Strukturierte, textuelle Daten in Unternehmen folgen Merkmalen, die für den Bereich der *Enterprise Search* charakteristisch und für eine effektive Informationsextraktion in diesem Kontext entscheidend sind.

Domänenspezifische Syntax von Entitäten. Zur eindeutigen Identifikation bestimmter Typen von Entitäten benutzen Unternehmen häufig eine *domänenspezifische Syntax*, die durch eine Art von Bildungsvorschrift festgelegt wird. Beispiele für solche Entitäten sind Produktbezeichnungen (z. B. *„ABC-Spax-S Senkkopf, PZ-Kreuzschlitz“* oder *„HP Pavilion dv6-1200“*), die internationale Standardbuchnummer ISBN (z. B. *„978-3-86680-192-9“*) oder Bank-Identifikationscodes gemäß

SWIFT-BIC (Bsp.: „*RBOSGGSX*“). Zum Teil sind Identifikatoren für bestimmte Entitäten im Unternehmenskontext auch gesetzlich vorgeschrieben, wie z. B. für Rechnungsnummern[1] (Bsp.: „*RN.12-10-2008#4*“). Bekannte Bezeichner in *domänenspezifischer Syntax* können in der Regel mit einen Wörterbuch-basierten Ansatz (z. B. [AC75]) sehr gut in Dokumenten lokalisiert und zugeordnet werden, da zumeist keine Mehrdeutigkeiten bestehen.

Häufig sind jedoch nicht alle Bezeichner einer bestimmten Klasse von Entitäten im Vorhinein bekannt, wie z. B. Rechnungsnummern für zukünftige Rechnungen oder alle (auch zukünftige) Produktnamen eines Zulieferers. In den meisten Fällen können Bildungsvorschriften in einfache Reguläre Ausdrücke abgebildet (z. B. eine zur Illustration vereinfachte Regel[2] für ISBNs: [0-9]{1,3}-[0-9]-[0-9]$^+$-[0-9]$^+$-[0-9]) und zur Extraktion verwendet werden, so dass auch unbekannte Entitäten eines bestimmten Typs extrahiert werden können.

Eine manuelle Erstellung von derartigen Extraktionsregeln ist jedoch in vielen Fällen aufwändig. Eine generische Methode zur automatischen Inferenz von Bildungsvorschriften aus einem Satz von bekannten Bezeichnern verspricht eine enorme Reduktion des manuellen Aufwands. Eine hohe Vollständigkeit und Genauigkeit bei der Extraktion ist für die praktische Anwendbarkeit eines solchen Verfahrens entscheidend und wirft eine Reihe von nicht trivialen Problemen auf, die im Rahmen dieser Arbeit untersucht werden.

Nutzung von Fachsprache. Mitarbeiter innerhalb eines Unternehmenskontexts referenzieren Entitäten neben der Verwendung von eindeutigen Identifikatoren häufig durch *Fachsprache* [Hah83], welche in den meisten Fällen in strukturierter Form in bestimmter Art und Weise definiert ist. Fachsprache wird z. B. in Terminologien, Thesauri oder Glossaren gepflegt, tritt aber auch als Beschreibung von Datenbankinhalten [IN07] auf.

Fachsprache beinhaltet unter anderem *Fachwörter*, *Eigennamen* und *Akronyme*. Potentielle Vorkommen von vordefinierten Konzepten in Texten können oft nicht ohne weiteres einer in den strukturierten Daten vordefinierten Semantik zugeordnet werden. So kann das Akronym „*RFC*“ für „*Request for Comments*“ „*Request for Change*“ oder „*Remote Function Call*“ im IT-Bereich stehen[3]. Die korrekte Semantik ergibt sich zumeist erst aus dem Kontext der Verwendung und kann z. B. auf Basis von anderen Wörtern in einem Text erschlossen werden.

Fachsprache kann aber auch Umschreibungen beinhalten, die aus einer Ordnung von allgemein geläufigen Begrifflichkeiten gebildet wird, wie z. B. die Produktnamen „*Regler für die Bremskraft von Anhängern*“ oder „*Schnittstelle zum Tanklagerverwaltungssystem*“[4]. Solche Umschreibungen sind im Kontext eines Unternehmens häufig feste Produktbezeichnungen, die weder einer speziellen Syntax folgen, noch spezielle Eigennamen oder ähnliches beinhalten.

Die heterogene Verwendung von Fachvokabular stellt bei der Identifikation von Entitäten in Texten ein weiteres komplexes Problem dar. Erschwert wird die Identifikation von Entitäten z. B. durch die Diversität von Beschreibungsmöglichkeiten, die es für das selbe Konzept gibt. Das Konzept „*Regler für die Bremskraft von Anhängern*“ kann auch durch die Varianten „*Bremskraftregler von Anhängern*“ oder „*Regler für Bremsen*“ in einem Text wie einer informellen Bestellung referenziert werden. Insbesondere potenziert sich das Problem von Ambiguitäten wenn Bestandteile der Bezeichner im normalen Sprachgebrauch in der entsprechenden Domäne häufig auftreten (z. B. in den oben genannten Bezeichnern „*Regler*“ oder „*Schnittstelle*“).

Beziehungen zwischen Entitäten. Die im Rahmen der Informationsintegration in Unternehmen häufig verwendeten Systeme zur Stammdatenverwaltung sind nur ein Indiz für die intensiven Bemühungen, eine möglichst allumfassende Sicht auf Entitäten im Unternehmenskontext zu gewinnen. Es wird versucht, alle Beziehungen zwischen unternehmensre-

[1] siehe §14 Abs. 4 UStG

[2] In Syntax der Java API für Reguläre Ausdrücke – siehe: `http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html`, Stand 01.05.2010

[3] siehe `http://help.sap.com/saphelp_glossary/en/index.htm`, Stand 01.02.2010

[4] Bezeichnung der SAP Komponente mit dem Identifikator IS-OIL-DS-TAS

levanten Entitäten zu erfassen. Die Bemühungen um eine ganzheitliche Sicht führen dazu, dass Datenbanken betriebswirtschaftlicher Software (z. B. von *Enterprise Resource Planing*-Systemen) zum Teil mehrere zehntausend Datenbanktabellen [BH08] definieren, die über Primär-/Fremdschlüsselbeziehungen miteinander verbunden sind.

Auch Terminologien und andere strukturierte Datenquellen definieren häufig neben einfachen Begriffsklärungen auch Beziehungen zwischen den Entitäten und erfassen zusätzliche Metadaten. Wir bezeichnen im Kontext dieser Arbeit Datenbanken und andere Terminologien, die mannigfaltige Beziehungen zwischen Entitäten beschreiben, als *Graph-strukturierte Referenzdatensätze* (formale Definition in Kap. 2.3).

Beziehungen zwischen Entitäten und zusätzliche Metadaten können zur Identifikation und Disambiguierung von Entitäten in Text herangezogen werden. Einige Entitäten, die im Unternehmenskontext vorkommen, sind abstrakte Aggregationen von anderen Entitäten, denen oft kein sinnvoller textueller Inhalt, abgesehen von künstlichen Identifikatoren, zugeordnet werden kann. Solche abstrakten Entitäten, wie z. B. Geschäftstransaktionen stellen häufig n-äre Beziehungen der beteiligten Entitäten dar.

Wenn in einem Text die von einer abstrakteren Entität subsumierten Entitäten identifiziert werden, kann unter bestimmten Umständen davon ausgegangen werden, dass ein Text implizit die abstraktere Entität referenziert. Auf diese Art und Weise können z. B. Geschäftstransaktionen in Texten indirekt nachgewiesen werden, auch wenn ihr Identifikator im Text nicht erwähnt wurde.

Die Konzeption einer Methode zur Identifikation von Entitäten unter Ausnutzung von Graph-strukturierten Referenzdatensätzen, die Entitäten und Beziehungen definieren und durch Fachsprache beschrieben werden, steht neben der Inferenz von Bildungsvorschriften im Fokus dieser Arbeit. Bei der Identifikation sind eine Reihe von Forschungsproblemen relevant, die in ihrer Gesamtheit bisher in der Literatur nicht betrachtet wurden.

1.1.3 Beitrag der Arbeit

In der vorliegenden Arbeit untersuchen wir die Extraktion und Identifikation von Entitäten im Unternehmensumfeld. In Abb. 1.1 ist ein Überblick über den in dieser Arbeit verfolgten Ansatz dargestellt. Die Grundlage dieser Arbeit ist die Informationsextraktionsplattform *RapidUIM* [BBH+09, BFBS10] (oben in Abb. 1.1), welche die (visuelle) Kompositition von generischen und anwendungsspezifischen IE-Modulen zu Extraktionsplänen unterstützt und so eine effiziente Entwicklung von spezifischen IE-Systemen ermöglicht (mittig in Abb. 1.1).

Im Speziellen untersuchen wir Methoden zur effizienten Ausführung von Extraktionsplan-basierten IE-Systemen und Verfahren zur effektiven Ausnutzung von strukturierten Referenzdaten für die Extraktion und Identifikation von Entitäten, die als generische Module in die Informationsextraktionsplattform eingebettet werden. Im Kontext von strukturierten Referenzdaten betrachten wir zum einen Methoden zur *Regelinferenz* anhand von strukturierten Beispieldaten (siehe „Regelinferenz“ in Abb. 1.1). Zum anderen diskutieren wir Verfahren für den *Abgleich* von unstrukturierten Daten mit strukturierten Referenzdaten und die Ausnutzung der Referenzdatenstruktur zur *Disambiguierung* (siehe „Abgleich“ und „Disambiguierung“ in Abb. 1.1).

Im verbleibenden Kap. 1.1 werden die hier betrachteten Problemstellungen anhand einer Anwendung zur (semi-)automatischen Rechnungsverarbeitung eingeführt. Ausschnitte von Beispielrechnungen und zugehörige strukturierte Daten eines RDBMS sind zur Illustration des Anwendungsfalls in Abb. 1.3 auf S. 9 dargestellt. Das Ziel der Anwendung ist es, aus einer digitalisierten Rechnung die Rechnungsnummer zu extrahieren (siehe Ⓐ in Abb. 1.3), die zugehörige Bestellung in einem RDBMS zu identifizieren (siehe Ⓑ in Abb. 1.3) und die einzelnen Rechnungspositionen mit der Bestellung abzugleichen – also z. B. den Text „Control & Regulation Unit“ mit dem Eintrag „Control Unit“ in der Tabelle „Product“.

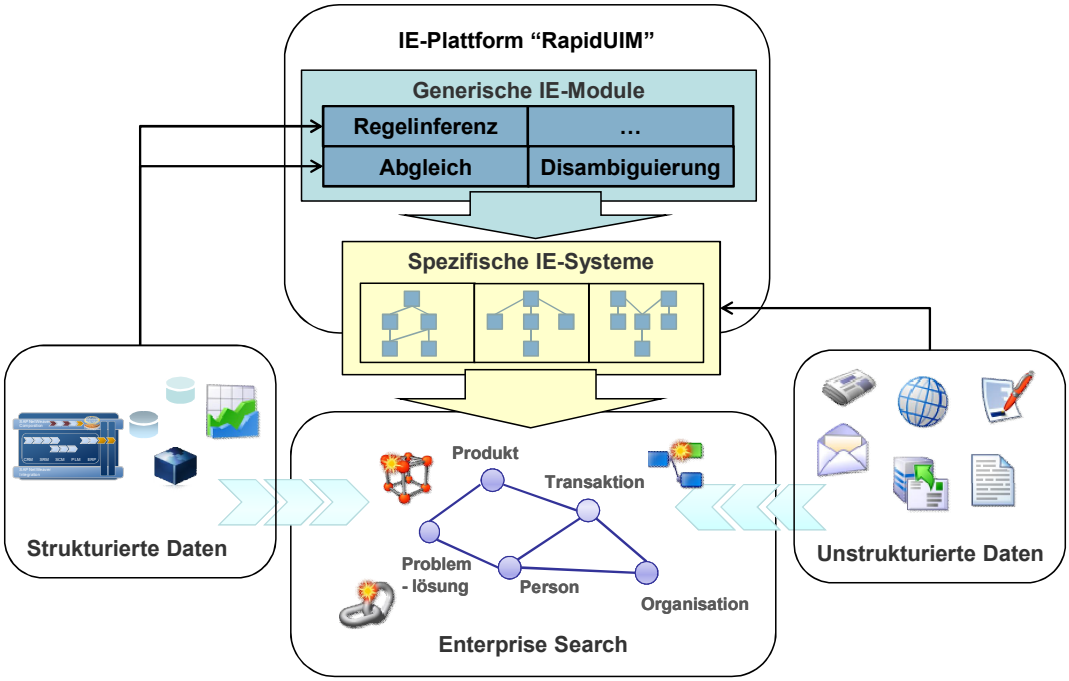


Abb. 1.1: Überblick: Extraktions- und Integrationsplattform für die Enterprise Search

Bevor wir jedoch auf die einzelnen Probleme bezüglich einer effektiven Ausnutzung von Referenzdaten eingehen, diskutieren wir die Extraktionsplattform und die effiziente Ausführung von IE-Systemen.

1.1.3.1 Laufzeitoptimierung von Extraktionsplan-basierten Systemen

Extraktionspläne beschreiben im Allgemeinen die Abhängigkeiten zwischen verschiedenen Operationen zur Informationsextraktion und definieren das Verhalten und die Extraktionslogik eines anwendungsspezifischen IE-Systems in deskriptiver Art und Weise. In Abb. 1.2 ist schematisch ein Extraktionsplan, wie er mit der hier vorgestellten *RapidUIM*-Plattform erstellt werden kann, dargestellt. Ein Extraktionsplan in *RapidUIM* ist definiert als:

Definition 1.1.2 (*Extraktionsplan in RapidUIM*)

> *Ein Extraktionsplan in RapidUIM wird als ein gerichteter azyklischer Graph $\mathcal{P} = (M, A)$, dessen Knoten M parametrisierte IE-Modulinstanzen und Kanten A eine Menge von Abhängigkeiten repräsentieren, dargestellt. Extraktionspläne enthalten mindestens ein Importmodul. Die Importmodule sind Quellknoten des Graphen und Datenquellen des IE-Programms. Alle weiteren Knoten (IE-Module) repräsentieren einzelne Operationen in der Datenverarbeitung. Kanten eines Extraktionsplans beschreiben den Datenfluss zwischen IE-Modulen zur Laufzeit.*

Andere Extraktionsplan-basierte Ansätze, die im Kontext der Projekte SystemT [KLR+08] oder CIMPLE [SDNR07] entwickelt wurden, erwarten eine deklarative Beschreibung von Extraktionsplänen in Form von an SQL oder Datalog angelehnten Sprachen, die in Ausführungspläne überführt werden. Ausführungspläne definieren ähnlich wie in Abb. 1.2 den Datenfluss zwischen IE-Modulen.

Der Extraktionsplan in Abb. 1.2 zeigt eine vereinfachte Darstellung für die Verarbeitung von Rechnungen. Es sind zwei anwendungsspezifische IE-Module zur Vorsegmentierung von Dokumenten („Absatz", „Zeile") und drei generische Module zur Extraktion und Identifikation

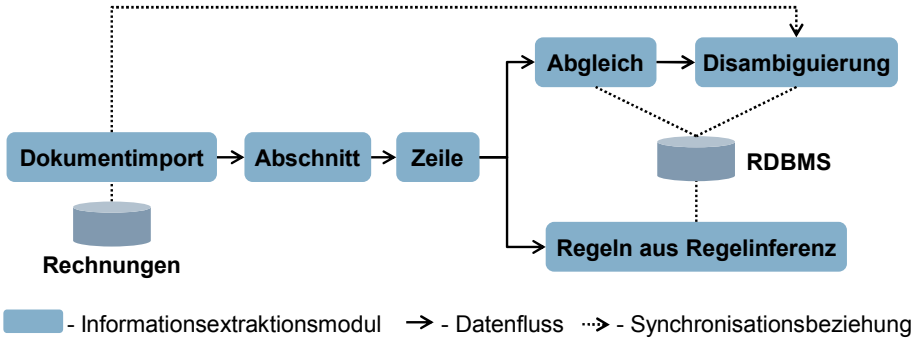


Abb. 1.2: Vereinfachter Extraktionsplan zur Rechnungsverarbeitung

von Entitäten mit Referenzdaten dargestellt, die mit einem RDBMS interagieren („Regeln aus Regelinferenz", „Abgleich" und „Disambiguierung"). Das Modul zur Regelinferenz erlernt Regeln zur Extraktion von Rechnungsnummern und das Modul „Abgleich" identifiziert Übereinstimmungen zwischen Text und RDBMS, wie z. B. Korrespondenzen von Rechnungs- und Bestellpositionen.

Ein Extraktionsplan definiert grundsätzlich die Abhängigkeiten zwischen IE-Modulen, aber auch die Abhängigkeiten zwischen Daten, die durch die Module verarbeitet werden. Zur Laufzeitverbesserung eines IE-Systems können unabhängige IE-Module wie das Modul zur Identifikation von Übereinstimmungen und das Modul zur Extraktion von Rechnungsnummern parallel ausgeführt werden.

Eine weitergehende Parallelisierung ist zur Verbesserung der Laufzeit wünschenswert. Unabhängige Daten können ebenfalls parallel verarbeitet werden. Im Beispiel können Regeln auf Zeilen angewendet werden, bevor alle Zeilen durch das Vorgängermodul zur Verfügung gestellt wurden. Weiterhin können potenziell auch mehrere Instanzen eines Moduls, z. B. zum Abgleich von Zeilen des Texts mit Bestellpositionen parallel Daten verarbeiten.

In vielen Anwendungsfällen, insbesondere wenn Beziehungen extrahiert werden sollen, bestehen komplexere, indirekte Datenabhängigkeiten. Im Falle von binären Beziehungen ist deren Extraktion häufig von der Extraktion von allen Entitäten innerhalb eines Satzes oder Abschnitts abhängig. Im hier dargestellten Anwendungsfall soll die Disambiguierung den gesamten Dokumentkontext berücksichtigen. Die im Rahmen dieser Arbeit eingeführten Synchronisationsabhängigkeiten (siehe Abb. 1.2) erlauben es, derartige Sachverhalte in Extraktionsplänen auszudrücken und den Datenaustausch zwischen IE-Modulen zur Sicherstellung von konsistenten Ergebnissen zu synchronisieren, so dass Zwischenergebnisse so früh wie möglich nebenläufig verarbeitet werden können.

Wir definieren das Ziel der Laufzeitoptimierung dementsprechend folgendermaßen:

Ziel 1 (*Laufzeitoptimierung von Extraktionsplan-basierten Systemen*)

> *Ziel der Laufzeitoptimierung ist es, gegeben ein Extraktionsplan, der Aufgaben- und Datenabhängigkeiten zwischen einzelnen IE-Modulen in einem Extraktionssystem beschreibt, mögliche unabhängige Einzeloperationen zu identifizieren und unabhängig voneinander, in asynchroner Art und Weise, nebenläufig auszuführen.*

Beitrag. Bisherige Ansätze zur Parallelisierung von IE-Systemen führen in der Regel parallele oder verteilte Instanzen eines IE-Programms aus und erhöhen so den Dokumentendurchsatz. Die Verarbeitungszeit für ein einzelnes Dokument wird dadurch nicht verbessert. Dies ist jedoch entscheidend, wenn geringe Antwortzeiten, wie z. B. für Extraktionsdienste im Unternehmenskontext relevant sind.

Wir beschreiben in der vorliegenden Arbeit ein Verfahren zur Dokumentverarbeitung, das den Datenaustausch zwischen IE-Modulen als Datenströme interpretiert und damit eine weitge-

hende Parallelisierung von Datenverarbeitungskomponenten ermöglicht, wenn Informationen zu Aufgaben- und Datenabhängigkeiten zur Verfügung stehen.

Durch die Entkopplung von IE-Modulen, die autonom ausgeführt werden, und die Kommunikation von IE-Modulen über Datenströme in gemeinsam genutzten Hauptspeicherbereichen, kann eine Parallelisierung erreicht werden, die die Dokumentverarbeitungszeiten signifikant verbessert. Es wird weiterhin ein Verfahren zur effizienten Überprüfung von Datenabhängigkeiten vorgestellt, das den *Overhead* der notwendigen Synchronisation des Datenstroms minimiert.

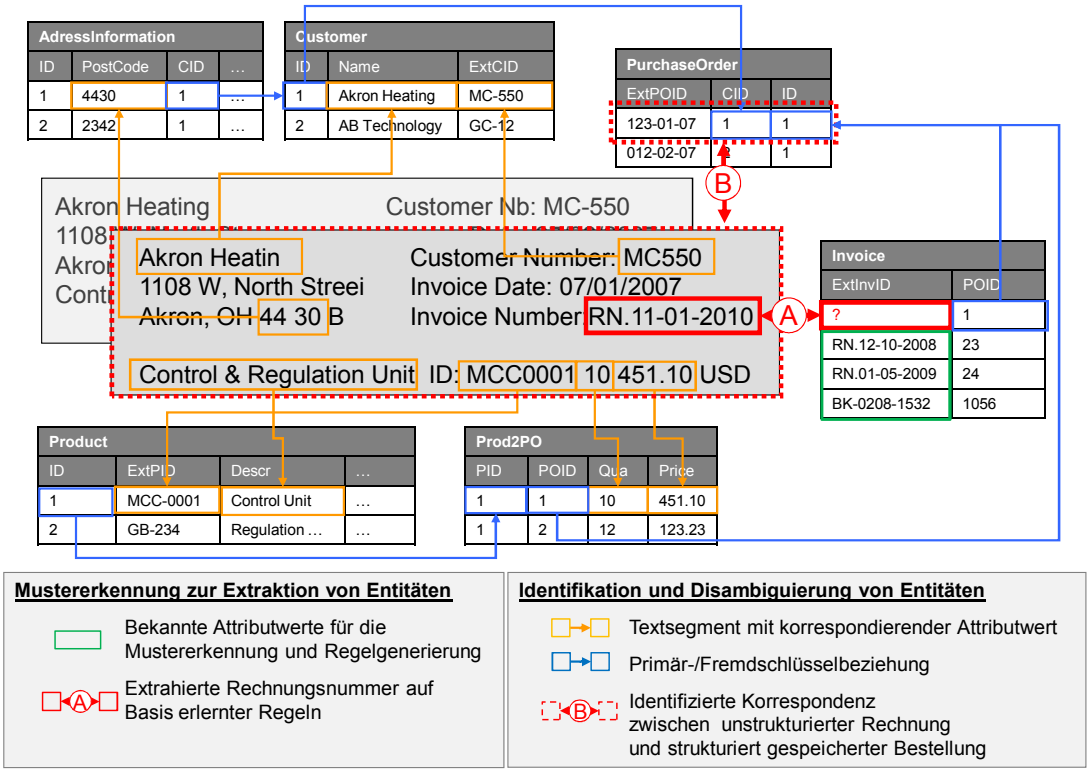


Abb. 1.3: Extraktion und Identifikation von Entitäten zur Verarbeitung von Rechnungen

1.1.3.2 Mustererkennung zur Extraktion von Entitäten

In Abb. 1.3 ist die Problemstellung der *Mustererkennung zur Extraktion von Entitäten* durch Ⓐ gekennzeichnet. Das Ziel der Mustererkennung ist es in diesem Beispiel, aus einem Satz bekannter Rechnungsnummern von verschiedenen Zulieferern, welche in der Datenbankspalte „ExtInvID“ dargestellt sind, Regeln zu erlernen, um die Rechnungsnummer „RN-11-01-2010“ aus der aktuell verarbeiteten Rechnung zu extrahieren. Allgemeiner kann das Ziel der Mustererkennung zur Extraktion von Entitäten wie folgt definiert werden:

Ziel 2 (*Inferenz von Regulären Ausdrücken zur Extraktion von Entitäten*)

> *Gegeben eine Liste von bekannten, positiven Beipielinstanzen, die eine Teilmenge einer Klasse von Entitäten beschreibt, ist das Ziel der hier untersuchten Mustererkennung, signifikante syntaktische Merkmale zu erkennen und Regeln zur Extraktion von unbekannten Entitäten der selben Klasse in Form von Regulären Ausdrücken abzuleiten.*

Insbesondere muss ein generischer Ansatz für die Inferenz von Regulären Ausdrücken folgende Problemstellungen berücksichtigen:

1. Entitäten, die einer Klasse zugerechnet werden, sind häufig syntaktisch heterogen, da z. B. mehrere Bildungsvorschriften verwendet wurden, um diese zu erzeugen. Im Beispiel in Abb. 1.3 liegt der Ursprung der Rechnungsnummern „RN.01-05-2009“ und „BK-0208-1532“ bei verschiedenen Zulieferern. Die Methode muss daher syntaktisch zusammengehörige Partitionen innerhalb einer gegebenen Liste von Bezeichnern erkennen.

2. Variable Bestandteile in der Syntax müssen identifiziert werden, so dass generierte Regeln eine hohe Vollständigkeit bei der Extraktion aufweisen. In den Rechnungsnummern im Beispiel aus Abb. 1.3 sind die fortlaufende Nummerierung sowie Monats- und Jahreszahl in „RN.**12**-**10**-**2008**“ und „RN.**01**-**05**-**2009**“ solche variablen Bestandteile.

3. Fixe Bestandteile in der Syntax der Entitäten müssen identifiziert werden, so dass die generierten Regeln eine hohe Genauigkeit bei der Extraktion aufweisen. In den Rechnungsnummern in Abb. 1.3 sind das Präfix „**RN.**“ und die Trennung durch „**-**“ in „**RN.**12-10-2008“ und „**RN.**01-05-2009“ fixe Bestandteile.

4. Muster können auf verschiedenen syntaktischen Ebenen auftreten. Zum Beispiel sind für gewisse Entitäten Merkmale auf Wortebene (z. B. charakteristische Wörter und Wortklassen) und für andere Konzepte Musterbeschreibungen auf Zeichenebene (z. B. charakteristische Zeichen und Zeichenklassen) zur Repräsentation der entsprechenden Bildungsvorschriften besser geeignet. Ein Algorithmus zur Inferenz von Regulären Ausdrücken muss derartige Charakteristiken identifizieren.

Beitrag. Im Rahmen dieser Arbeit wird eine Methode zur Inferenz von Regulären Ausdrücken anhand von Beispielentiäten untersucht, welche syntaktisch ähnliche Partitionen in einer Liste von Bezeichnern detektiert und Muster auf verschiedenen syntaktischen Ebenen identifiziert. Jede Partition wird entsprechend einer Menge von Zeichen- und Wortklassen zerlegt und in eine verallgemeinerte Musterbeschreibung überführt, die alle Entitäten einer Partition beschreibt.

Spezialisierungsregeln wenden Heuristiken an, um basierend auf der Verteilung von Instanzteilsequenzen signifikante fixe Bestandteile zu selektieren und damit speziellere Regeln aus den verallgemeinerten Merkmalen zu generieren. Auf Basis des Prinzips der „*Minimum Description Length*“ werden Teilmuster verschiedener syntaktischer Ebenen bewertet und ausgewählt, so dass ein Ausgleich zwischen generelleren und spezielleren Musterbestandteilen erfolgt. Die erzeugten Regeln schaffen somit einen Ausgleich zwischen *Vollständigkeit* und *Genauigkeit* bei der Extraktion von unbekannten Entitäten.

Bekannte Regellernverfahren im Bereich der Informationsextraktion unterstützen die beschriebenen Problemstellungen nicht. Weiterhin werden keine Trainingsdokumente oder negative Beispielinstanzen wie bei der Verwendung von stochastisch/ statistischen Modellen (z. B. *Hidden Markov Models* oder *Conditional Random Fields* [Sar08]) benötigt.

Die Erzeugung von Regulären Ausdrücken, an Stelle von stochastisch/ statistischen Modellen, ermöglicht eine einfache Integration mit Regel-basierten Extraktionsverfahren, die zumeist zur Implementation von IE-Szenarien im Kontext der *Enterprise Search* eingesetzt werden, und sich durch geringere Dokumentverarbeitungszeiten gegenüber stochastischen Verfahren auszeichnen [LKR+08]. Regeln ermöglichen weiterhin eine wenig aufwändige Fehlersuche und Modifikation der Extraktionsmechanismen durch Nutzer mit entsprechender Expertise [Sar08].

1.1.3.3 Identifikation und Disambiguierung von Entitäten

Die Problemstellungen der Identifikation und Disambiguierung von Entitäten ist im Beispiel aus Abb. 1.3 durch Ⓑ illustriert. Die Herausforderung besteht darin, die Bestellung („Purchase-Order“) mit dem Identifikator „123-01-07“ anhand der zu ihr in Beziehung stehenden Entitäten, die im Text auftreten, zu identifizieren.

Im Beispiel sind die Entitäten, die zum indirekten Nachweis genutzt werden: der Kunde („Customer“) mit dem Namen „Akron Heating“, das bestellte Produkt („Product“) mit der Beschreibung „Control Unit“ und dem Identifikator „MCC-0001“ sowie die Bestellmenge (‘‘Prod2PO“/„Qua“) von „10“. Eine Disambiguierung ist z. B. im Falle der Übereinstimmung zwischen „Control & Regulation Unit“ im Text und „Control Unit“ im RDBMS sowie für die Bestellmenge „10“ für den Abgleich von Bestell- und Rechnungspositionen notwendig.

Allgemeiner kann das Ziel der Identifikation und Disambiguierung von Entitäten wie folgt definiert werden:

Ziel 3 (*Identifikation und Disambiguierung von Entitäten*)

> *Ziel der Identifikation von Entitäten ist der Abgleich von Text und Referenzdatensatz auch in Situationen, wenn ein fehlerhafter Text vorliegt; Wörter, die Bestandteil von Entitäten sind, im Text verteilt auftreten und Entitäten im Text nur implizit durch in Beziehung stehende Entitäten referenziert werden. Ziel der Disambiguierung ist es, mehrdeutige, direkte und indirekte Übereinstimmungen von Text und Referenzdatensatz hinsichtlich ihrer Relevanz für den Dokumentkontext zu bewerten.*

Die Identifikation und Disambiguierung von Entitäten zielt damit insbesondere auf Referenzdaten ab, die sich durch Fachvokabular mit begrifflichen Ordnungen [Hah83] auszeichnen. Problemstellungen, die sich in diesem Kontext ergeben, sind:

1. Die Resultate der Identifikation sollen eine hohe Vollständigkeit aufweisen. Es werden z. B. fehlertolerante Verfahren zur Identifikation benötigt, um die Korrespondenz zwischen dem Produktidentifkator „MCC0001“ im Text und „MCC-0001“ im RDBMS oder „Control & Regulation Unit“ und „Control Unit“ herzustellen. Weiterhin soll die Bestellung mit der Nummer „123-01-07“ anhand benachbarter Entitäten identifiziert werden, hier z. B. durch den Kunden und die Produktliste, die mit der Bestellung in Beziehung stehen.

2. Der Ansatz soll eine hohe Genauigkeit bei der Identifikation ermöglichen. Problematisch ist, dass im RDBMS eine große Menge von Bestellungen des Kunden „Akron Heating“ für das Produkt „Control & Regulation Unit“ vorliegen können und das Zulassen von approximativen Übereinstimmungen zwischen RDBMS und Text häufig zu uneindeutigen Korrespondenzen führt. Eine geeignete Disambiguierung für indirekt und approximativ identifizierte Entitäten ist notwendig, um eine akzeptable Genauigkeit zu erreichen. Weiterhin treten in Unternehmensdaten häufig Hierarchien mit mehreren Ebenen, wie zum Beispiel Produkt-Teil-Hierarchien auf. Wenn mehrere Entitäten in einer Hierarchie approximativ oder nur zum Teil mit einem Text übereinstimmen, ist es problematisch, die relevantesten (implizit) im Text referenzierten Entitäten innerhalb der Hierarchie zu selektieren (Details in Kap. 1.2).

Beitrag. Im Rahmen dieser Arbeit wird ein Ansatz zur Identifikation und Disambiguierung von Entitäten vorgestellt, der auf einem approximativen Abgleich von Text- und Referenzdaten basiert. Auch im Dokument verteilte, partielle Übereinstimmungen mit verschiedenen Attributen einer Entität werden berücksichtigt. Wir betrachten dabei im Gegensatz zu vorherigen Verfahren nicht nur vollständige Attributinhalte oder einzelne Wörter, sondern Wortgruppen flexibler Länge zur Relvanzbewertung von Übereinstimmungen.

Basierend auf der Bewertung von partiellen Übereinstimmungen zwischen Textsegmenten und Attributen und einer Graph-basierten Aggregation dieser Werte unter Ausnutzung der vordefinierten Beziehungen, kann eine hohe Identifikationsqualität erreicht werden. Bisherige Verfahren lösen lediglich Teile der dargestellten Problemstellung und sind daher hinsichtlich der Vollständigkeit und Genauigkeit der Ergebnisse dem hier entwickelten Verfahren unterlegen.

Ergebnis der Identifikation ist eine nach *Relevanz* geordnete Liste von Teilgraphen des Referenzdatensatzes. Die Ordnung der Teilgraphen kann durch Anwendungen der *Enterprise Search*

genutzt werden, um z. B. einem Nutzer eine Liste von nach Relevanz geordneten, potenziell identifizierten Bestellungen und durch sie subsumierte Entitäten zur Verifikation vorzuschlagen. Wenn keine Mehrdeutigkeiten auftreten, besteht die Möglichkeit, den Prozess der Rechnungsverifikation komplett zu automatisieren.

1.1.4 Aufbau der Arbeit

Kapitel 1. Nachdem die Problemstellungen und Forschungsprobleme, die im Kontext dieser Arbeit untersucht werden, eingeführt wurden, werden wir im verbleibenden Kap. 1 mögliche Anwendungsszenarien analysieren. Wir stellen als Anwendungsfälle der hier vorgestellten Konzepte neuartige Systeme im Umfeld der *Enterprise Search* vor, die im Kontext dieser Arbeit entwickelt wurden. Im Speziellen gehen wir auf die (semi-)automatische Rechnungsverifikation (publiziert in [BSB+08, BLD08]), ein System zur Dokumentsuche, welches Anfragen und Dokumente semantisch anreichert (publiziert in [LBB08, MBBL09, BBH+09, BHH+10]), und die strukturierte Informationssuche in und statistische Analyse von Texten (publiziert in [BBM+09b, TBBA10]) ein.

Kapitel 2. In Kap. 2 geben wir einen Überblick zum Stand der Technik und stellen offene Forschungsprobleme heraus. Zu Beginn diskutieren wir Techniken zur Entwicklung von anwendungsspezifischen IE-Systemen und motivieren damit den Mehrwert von Extraktionsplan-basierten Techniken, insbesondere im Unternehmensumfeld. Im Weiteren werden Optimierungsmethoden für IE-Systeme vorgestellt und die Neuartigkeit einer Intra-Dokument-Parallelisierung in der Datenverarbeitung herausgestellt. Abschließend betrachten wir Verfahren zur Ausnutzung von strukturierten Referenzdaten im Kontext der Informationsextraktion und diskutieren offene Forschungsfragen. In Kap. 2 wird die Eignung verschiedener Technologien für die hier diskutierten Problemstellungen aus Anwendungsperspektive bewertet. Wir vertiefen die Abgrenzung zu verwandten Arbeiten hinsichtlich der logarithmischen Herangehensweise in späteren Kapiteln.

Kapitel 3. In Kap. 3 werden die wichtigsten Konzepte der *RapidUIM*-Plattform im Bezug auf eine effiziente Entwicklung und Ausführung von IE-Systemen vorgestellt (Teilprobleme und -lösungen veröffentlicht in [BBLM09, BBM+09b, BBM09a, BBH+09, BFBS10]). In Kap. 3.1 gehen wir auf die Architektur, Modulschnittstellen und das Meta-Datenmodell ein. Der Fokus von Kap. 3.2 liegt auf der effizienten Ausführung. Ausgehend von herkömmlichen Ansätzen zur Dokumentverarbeitung werden Konzepte erläutert, die eine umfassende Parallelisierung von IE-Modulen unterstützen. In Kap. 3.3 evaluieren wir die erzielbaren Laufzeiten im Vergleich zu einer Dokumentverarbeitung mit der populären *UIMA*-Plattform [FL04] in verschiedenen Anwendungsszenarien. Abschließend analysieren wir in Kap. 3.4 andere Ansätze zur Laufzeitoptimierung und deren Relation zu den hier entwickelten Konzepten.

Kapitel 4. Die Konzepte bezüglich der Mustererkennung zur Extraktion von Entitäten werden in Kap. 4 erläutert (Teilprobleme und -lösungen veröffentlicht in [BRBM09]). Es gliedert sich in drei Unterkapitel. In Kap. 4.1 werden die wesentlichen Konzepte erläutert. Ausgehend von der Klassifikation der Beispielinstanzen durch Zeichen- und Wortklassensequenzen sowie der Abbildung in Präfix- und Suffixautomaten werden mögliche syntaktische Muster abgeleitet. Im Weiteren erläutern wir die Identifikation von fixen Zeichenketten zur Verbesserung der Genauigkeit und gehen auf die Auswahl von geeigneten Musterbeschreibungen ein. In Kap. 4.2 wird das entwickelte Verfahren mit einem maschinellen Lernansatz verglichen, der auf Basis von annotierten Beispieldokumenten trainiert wird und zusätzlich Kontextinformationen berücksichtigt. Wir analysieren abschließend in Kap. 4.3 verwandte Arbeiten.

Kapitel 5. Die Konzepte zur Identifikation und Disambiguierung von Entitäten werden in Kap. 5 erläutert (Teilprobleme und -lösungen veröffentlicht in [BBH+09, BHH+10]). Wir beschreiben Probleme, die bei der Verwendung von naiven Ansätzen mit einen Korpus von 120.000 Dokumenten beobachtet wurden. Im Weiteren erläutern wir die Bewertung von einzelnen Übereinstimmungen von Text und Attributen der Entitäten und diskutieren die Aggregation dieser Gewichte unter der Berücksichtigung von Beziehungen zwischen Entitäten. In Kap. 5.2 vergleichen wir das hier entwickelte Verfahren zu *TF-IDF*-basierten Ansätzen, die die Struktur von Referenzdaten ausnutzen sowie in jeweils einem Szenario zu einer Web-Suchmaschine und einem maschinellen Lernansatz, welche für die hier verfolgte Problemstellung adaptiert wurden. Verwandte Arbeiten werden in Kap. 5.3 abgegrenzt.

Kapitel 6. In Kap. 6 werden die Ergebnisse der vorliegenden Arbeit zusammengefasst und ein Ausblick auf zukünftige Forschungsfragen gegeben.

1.2 Anwendungsszenarien im Bereich der Enterprise Search

Die in dieser Arbeit konzipierten Methoden sind für eine Vielzahl von Anwendungsklassen im Bereich der *Enterprise Search* einsetzbar. Im Folgenden werden als Anwendungsfälle neuartige Systeme zur Verarbeitung, Anfrage und Analyse von unstrukturierten Daten vorgestellt, die im Kontext der vorliegenden Arbeit entwickelt wurden. Sie dienen der Darstellung der Problemstellungen und Illustration des Mehrwerts der hier vorgeschlagenen Lösungen. Dokumentenkorpora aus dem Umfeld der beschriebenen Anwendungsfälle werden neben anderen Datensätzen zur Evaluation der entwickelten Konzepte im Verlauf der Arbeit wieder aufgegriffen.

1.2.1 Szenario I: Automatische Verarbeitung von Einzeldokumenten

Im Folgenden wird der Anwendungsfall der Rechnungsverifikation aus Kap. 1.1 detailliert diskutiert (publiziert in [BSB+08] und [BRBM09]). Wir betrachten zuerst das Problem der Extraktion von unbekannten Entitäten und im Anschluss die Identifikation im Kontext von Graph-strukturierten Referenzdaten.

Mustererkennung für die automatische Rechnungsverifikation

Zur Extraktion von a priori unbekannten Entitäten wie Rechnungsnummern oder Produktnamen in *Rechnungen ohne Bestellbezug* sind im Anwendungsfall der Rechnungsverifikation eine Vielzahl von speziell an den Untenehmenskontext angepassten Regeln notwendig, die fortlaufend aktualisiert werden müssen. Eine Automatisierung der Regelerstellung durch eine (periodische) Musterinferenz auf Basis manuell gepflegter, historischer, strukturierter Daten verspricht eine automatische Adaption eines IE-Systems an die Gegebenheiten des Unternehmenskontext.

Zur Illustration der konkreten Problemstellungen zeigt Tab. 1.1 Beispiele für Bezeichner im Unternehmensumfeld und mögliche manuell erstellte Reguläre Ausdrücke. Im Folgenden werden Charakteristiken der Daten, die bei einer manuellen Regelerstellung beachtet und in einen Algorithmus zur Mustererkennung abgebildet werden müssen, für die verschiedenen Beispiele diskutiert:

Rechnungsnummern. Tab. 1.1.(A) zeigt Beispiele für Rechnungsnummern von zwei Zulieferern, die in einem System für das *Supplier Relationship Management* erfasst wurden. Einfache

Tab. 1.1: Beispiele für Bezeichner in der automatischen Rechnungsverarbeitung

(A) Rechnungsnummer	**(B)** Mundpflegeprodukte	**(C)** Nootebookbezeichner
BK-0909-4301	Colgate Total Fresh	m8100y
BK-0208-1532	Colgate Total Mint	m8200n
BK-0112-013	Colgate Double Soda	d5000t ATX
RN.1210-2008	Colgate Double Frescura	d5100t ATX
RN.0105-2009	Colgate Double Fresh	m8000z
(BK-(0|1)[0-9]$^+$-[0-9]$^+$)| (RN\.[0-9]$^+$-200[0-9])	Colgate ((Total [A-Z][a-z]$^+$)| (Double [A-Z][a-z]$^+$))	(m8[0-9]$^+$(y|n|z))| (d5[0-9]$^+$t ATX)

Reguläre Ausdrücke[5] der Art [A-Z]$^+$(-|\.)[0-9]$^+$(-)[0-9]$^+$, wobei „|“ für die Alternation, „\“ ein *Escape*-Symbol für Steuerzeichen und „$^+$“ die positive Kleenesche Hülle notiert, verspricht eine hohe Vollständigkeit bei der Extraktion. Es ist jedoch bei derartigen Ausdrücken häufig von einer geringen Genauigkeit in realen Anwendungsszenarien auszugehen, da z. B. auch Zeichenketten der Art „*TEL.0172-12345*“ oder andere Bezeichner mit ähnlicher Formatierung extrahiert werden würden. Vielmehr ist es notwendig, verschiedenste Charakteristiken der gegebenen Instanzen wie fixe und variable Zeichenketten und deren syntaktische Abfolge zum Beispiel in einem Regulären Ausdruck der Art (BK(-)(0|1)[0-9]$^+$(-)[0-9]$^+$)|(RN\.[0-9]$^+$-200[0-9]) zu kodieren.

Mundpflegeprodukte. Tabelle 1.1.(B) zeigt Beispiele für Produktnamen von Mundpflegeprodukten. Die Bezeichner werden durch Konkatenation von verschiedenen Wörtern (statt Zeichen) gebildet, wobei „Colgate“ der Name des Unternehmens, das darauf folgende Wort die Produktlinie und das letztere die konkrete Produktgattung spezifiziert. Eine geeignete Extraktionsregel wie Colgate ((Total [A-Z][a-z]$^+$)|(Double [A-Z][a-z]$^+$)) umfasst dementsprechend Unternehmensname, Produktlinie und ein darauf folgendes Wort, das mit einem Großbuchstaben beginnt ([A-Z][a-z]$^+$), so dass die Extraktion von unbekannten Produktgattungen unterstützt wird. Im Falle, dass die Beispielinstanzen eine größere Heterogenität hinsichtlich der Produktlinien aufweisen würden, wäre eine Regel der Art Colgate [A-Z][a-z]$^+$ [A-Z][a-z]$^+$ unter Umständen besser geeignet. Ein Ausgleich zwischen potenzieller Vollständigkeit und Genauigkeit der Regeln anhand der Spezifika der Beispielinstanzen ist daher (auch) bei deren manuellen Entwicklung notwendig.

Nootebookbezeichner. In Tab. 1.1.(C) sind Produktnamen eines Computerherstellers dargestellt, die durch eine wesentlich höherere syntaktische Heterogenität als vorherige Beispiele gekennzeichnet sind. Hier sind komplexe Konkatenationen von fixen und variablen Buchstaben- und Nummernfolgen notwendig, um Produktnamen in einer Bildungsvorschrift zu kodieren. Das Beispiel zeigt lediglich einen kleinen Ausschnitt von hunderten Produktnamen. Dementsprechend aufwändig ist eine manuelle Mustersuche und Regelentwicklung. Eine geeignete Regel für die hier dargestellten Beispielinstanzen ist z. B. (m8[0-9]$^+$(y|n|z))|(d5[0-9]$^+$t ATX).

Ein hier untersuchtes Verfahren zur Mustererkennung soll die Regelerstellung automatisieren, so dass der manuelle Aufwand zur Implementation von Regeln für einen spezifischen Unternehmenskontext minimiert wird, aber gleichzeitig eine einfache manuelle Feinjustierung der Extraktionsmechanismen unterstützt wird.

Identifikation von Entitäten zur Rechnungsverifikation

Zur automatischen Verifikation von *Rechnungen mit Bestellbezug* ist es notwendig, in strukturierten Daten vordefinierte Entitäten in Rechnungsdokumenten zu identifizieren. Abbildung

[5] in der Syntax der Java API für Reguläre Ausdrücke, siehe `http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html`, Stand 01.05.2010

1.4 zeigt zur Illustration des Anwendungsfalls ein aktuelles System für die Rechnungsverarbeitung (als Teil eines Systems für das *Supplier Relationship Management (SRM)*[6]), das den manuellen Vergleich von strukturierten Bestellinformationen und einer textuellen Rechnung unterstützt. Das dargestellte System realisiert die Zuordnung zwischen Rechnung und Bestellung auf Basis von im Text und der Datenbank vorhandenen Bestellnummern durch einen exakten Zeichenkettenabgleich.

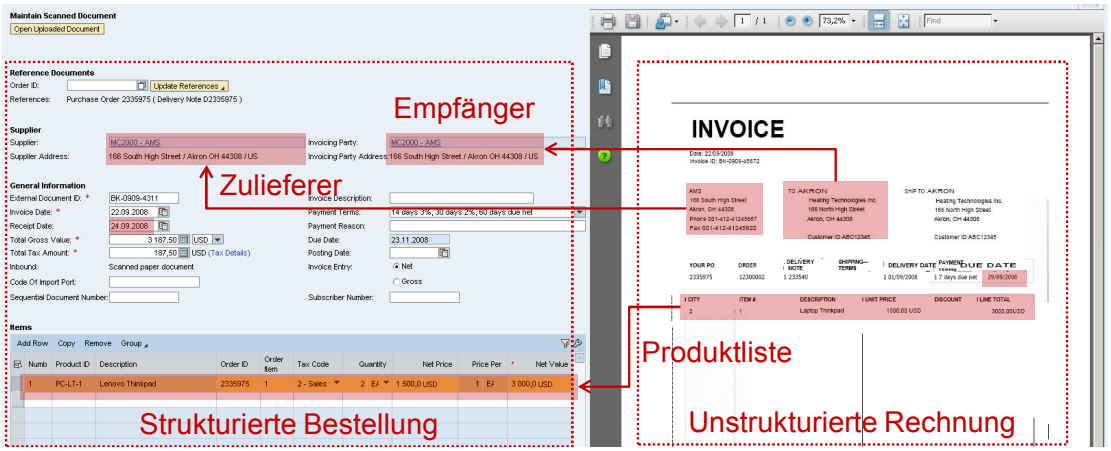


Abb. 1.4: Nutzerschnittstelle eines SRM-Systems zur Verifikation einer Rechnung anhand der zugehörigen Bestellung, die in strukturierter Form vorliegt

Die in dieser Arbeit untersuchte Identifikation und Disambiguierung von Entitäten adressiert zum einen die Zuweisung von Rechnungen und Bestellungen im Fall, dass keine Bestellnummer im Dokument detektiert werden kann, anhand von subsumierten Entitäten wie Empfänger, Zulieferer und der abstrakteren Entität „Produktliste“. Zum anderen sollen Bestellpositionen zu Textfragmenten zugewiesen werden, auch wenn sie nicht eindeutig identifiziert werden konnten, wie z. B. Paare von Produkten und Mengenangaben aus der Produktliste.

Basis der Zuordnung sind approximative Übereinstimmungen zwischen Textbestandteilen und Wortgruppen der Datenbankinhalte (z. B. von Zeilen innerhalb eines Adressblocks zu bekannten Zulieferernamen). Darauf aufbauend können Beziehungen der strukturierten Daten zur indirekten Identifikation von (abstrakteren) Entitäten verwendet werden (z. B. der Zuliefererадresse). Die Aggregation von im Text übereinstimmenden Entitäten zu abstrakteren Entitäten geht mit einer Zuordnung zu höherwertigen Dokumentstrukturen einher (z. B. eines Adressblocks zu einer Zuliefererадresse), bis schließlich das Gesamtdokument einer Bestellung zugeordnet werden kann.

Beginnend bei Zuweisungen von Dokument und wahrscheinlich zugehörigen Bestellungen können einem Nutzer Schritt für Schritt niederwertigere Textfragemente und Entitäten zur Bestätigung präsentiert werden. Die Verarbeitung von Dokumenten kann auf diese Art und Weise wesentlich im Vergleich zu einer manuellen Suche der Bestellung und händischen Verifikation beschleunigt, wenn nicht sogar in bestimmten Fällen vollkommen automatisiert werden.

Ein System zur Identifikation und Disambiguierung von Entitäten für spezielle Anwendungsfälle muss zum einen mit geringem Aufwand zu entwickeln und zum anderen effektiv sein, also eine hohe Genauigkeit und Vollständigkeit hinsichtlich der Identifikation aufweisen. Eine effiziente Ausführung bezüglich der Dokumentverarbeitungszeiten ist wünschenswert.

[6]SAP Business ByDesign SRM

1.2.2 Szenario II: Semantische Anreicherung für die unstrukturierte Dokumentsuche

Die Verarbeitung von Einzeldokumenten erfasst nur einen Teil der Anwendungsklassen, für die die in dieser Arbeit vorgeschlagenen Methoden einen Mehrwert bietet. Eine Vielzahl von Anwendungsgebieten im Bereich der *Enterprise Search* sind im Kontext der Suche von Dokumenten mit längeren, unstrukturierten Anfragen (> 5 Wörter), die komplexe Informationsbedürfnisse als z. B. im WWW adressieren, angesiedelt.

Nutzer müssen bei der Verwendung von herkömmlichen Systemen zur *Enterprise Search* häufig Anfragen iterativ umformulieren, um Schlüsselwörter zu „erraten", die ein fremder Autor in einem relevanten Dokument verwendet hat. Mehrdeutigkeiten von Wörtern im Unternehmenskontext verschlechtern häufig die Relevanz von Suchresultaten.

Strukturierte Referenzdaten können helfen, Suchanfragen wie auch Dokumente semantisch in den Unternehmenskontext einzuordnen und versprechen daher eine Verbesserung von Suchergebnissen. Im Folgenden wird ein Beispielsystem vorgestellt, das eine Dokumentsuche auf Basis einer semantischen Anreicherung von Anfragen und Dokumenten unterstützt. Teilprobleme und -lösungen des Systems wurden in [LBB08, BBH+09, MBBL09, BBLM09] publiziert.

Das SAP Community Network

Unternehmen betreiben häufig Internetforen, um die Kundenbindung und -zufriedenheit zu fördern [OVG+09]. Das *SAP Community Network (SCN)*[7] ist ein typisches Internetforum im Unternehmenskontext, in dem unternehmensinterne und -externe Nutzer Inhalte in Foren, Blogs, Wikis etc. pflegen.

Bei den angebotenen Inhalten im SCN stehen in der Regel SAP-Technologien im Mittelpunkt. Die Suche nach Inhalten wird durch eine Reihe von Problemen wie Mehrdeutigkeiten von Akronymen und technischem Vokabular erschwert. Im Fall des SCN führen diese Probleme bei der Suche nach relevanten Inhalten dazu, dass ein großer Teil der Forumfragen in gleicher oder ähnlicher Form mehrfach vorhanden ist, da relevante Inhalte nicht gefunden werden konnten (detailliert diskutiert in [MBBL09]). Im Folgenden wird zur Motivation der hier konzipierten Methoden eine Studie zu Suchintentionen im SCN und ein System zur semantischen Anreicherung von Anfragen und Dokumenten diskutiert (veröffentlicht in [LBB08]).

Zur Studie von Suchintensionen im SCN wählten wir 470.973 Suchanfragen des Jahres 2007 aus und analysierten sie hinsichtlich der Intention und angefragten Entitäten. Es wurden die populärsten 200 Anfragen (*Top200*), als auch eine Auswahl von wenig häufigen Anfragen (*Less200*) zur Analyse ausgewählt. Als Klassifikationskriterium wurden Suchintentionen nach [Bro02] verwendet. Es wird nach *Navigationsorientierten*, *Informationellen* und *Transaktionalen Anfragen* unterschieden. Die Ergebnisse der Studie sind in Tab. 1.2 dargestellt.

In *Top200* waren Anfragen bezüglich der Navigation zu Produktseiten häufig, wobei gleich häufig vollständige Produktnamen und deren z. T. mehrdeutigen Akronyme verwendet wurden. Anfragen, die eine detailliertere Auskunft zu Produkten als Ziel hatten (enthielten häufig „*What is*" oder „*How to*") sind in *Less200* bedeutender als in *Top200*. Ähnlich häufig wurde in *Top200* und *Less200* ein Informationsbedarf bezüglich der Nutzung von technischen Konzepten im Kontext von Produkten (wie z. B. „Webservice") identifiziert. In *Less200* traten Anfragen, die die Lösung eines konkreten Fehlers oder ein Konfigurationsproblem beschreiben, verhältnismäßig häufig auf. Transaktionale Anfragen (z. B. zum Download von Software) spielen eine untergeordnete Rolle.

Die Analyse der Suchintensionen zeigt, dass sowohl die Klassifikation von Inhalten (wie z. B. Homepage, Anleitung, Fehlerbeschreibung im Forum), als auch die enthaltenen Entitäten (wie z. B. Produkte, technische Konzepte, Fehler etc.) zur Verbesserung der Suche von Belang sind.

[7] siehe `http://scn.sap.com`, Stand 01.06.2010

Tab. 1.2: Überblick zu Intentionen von Suchanfragen im SCN

Intention	Anz. Top200	Anz. Less200
Navigation		
Produkt Homepage (vollst. Name)	41	9
Produkt Homepage (Akronym)	34	1
Unterseite des SCN	2	3
Information		
Beschreibung/ Anleitung zu Produkten	36	73
Allgemeiner technischer Begriff	33	27
Konfiguration und Fehlerlösung	3	20
Code-Beispiele	4	3
Transaktion		
Download	9	2
Uneindeutig		
	36	48

Verbesserung der Dokumentsuche mit RankIE

Zur Verbesserung der Suche für Anfragen, die komplexe Problemstellungen betreffen ($>$ 5 Wörter), wurde *RankIE* („***Rank**ing of Documents using **I**nformation **E**xtraction Graphs*“) [BBH+09] konzipiert, welche eine semantische Suche auf Basis von relevanten, disambiguierten Entitäten unterstützt. Anfragen mit einer Länge von mindestens fünf Wörtern umfassen im SCN ca. 15% der eine Million individuellen Suchanfragen pro Woche. Das Beispielystem *RankIE* wird im Folgenden zur Illustration einer Suche auf Basis einer semantischen Anreicherung mit den in dieser Arbeit vorgeschlagenen Methoden vorgestellt. Sie verspricht eine höhere Vollständigkeit in Suchergebnissen, da z. B. Akronyme aufgelöst werden können, und eine höhere Genauigkeit, da mehrdeutige Wörter und Wortgruppen disambiguiert werden.

Domänenwissen über Akronyme, technische Konzepte und Softwaremodule wird in der SAP Terminologie (SAPTerm) gepflegt. SAPTerm bietet eine logische Sicht auf Module, die in SAP-Systemen genutzt werden, und enthält weiterhin eine große Menge von Fachvokabular, das im Kontext der Module verwendet wird. Im weiteren Verlauf der Arbeit wird SAPTerm als laufendes Beispiel für typische Graph-strukturierte Referenzdaten im Unternehmensumfeld [Die06] herangezogen.

In SAPTerm werden 16.000 logische Softwaremodule in 32 Sprachen gepflegt. Softwaremodule sind in Hierarchien organisiert und mit Fachvokabular verknüpft. Abbildung 1.5 illustriert die Struktur von SAPTerm. Ein abstraktes Softwaremodul („Component“ in Abb. 1.5), wie z. B. „*Basis Components*“ oder „*Application Platform*“, ist Wurzelelement einer Modulhierarchie. Hierarchien spannen über maximal acht Ebenen. Zu jedem Softwaremodul werden Lang- wie auch Kurzname gespeichert („long_form“ und „short_form“ in Abb. 1.5), wie z. B. die Bezeichnung „*Basis Components*“ und der entsprechende Kurzname „*BC*“.

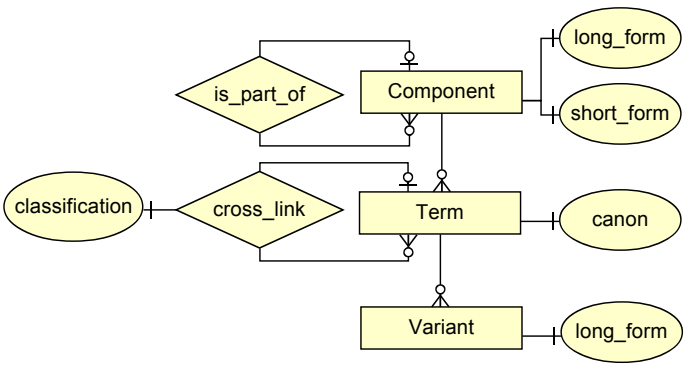


Abb. 1.5: Schematischer Überblick über die SAP Terminologie

Zusätzlich werden technisches Vokabular („Term“ in Abb. 1.5) und Akronyme („Variant“ in Abb. 1.5), die im Kontext von Softwaremodulen eine Rolle spielen, erfasst und Modulen zugeordnet. Im Februar 2008 betrug die Anzahl von technischen Konzepten und Akronymen ca. 150.000.

In *RankIE* wird SAPTerm genutzt, um technische Begriffe und Softwaremodule zu identifizieren und zu disambiguieren. Die Mustererkennung zur Extraktion von Entitäten wird zum Erlernen von Regeln für Fehlermeldungen verwendet, die zumeist einer spezifischen Syntax folgen, wie z. B. „CALL_FUNCTION_NOT_FOUND“ oder „java.lang.NullPointerException“. Die manuelle Erstellung von Regeln für Fehlermeldungen wäre extrem aufwändig, da sich deren Syntax stark mit anderen Sprachkonstrukten der Programmiersprachen *Java* und *ABAP* überschneiden. Beispiellisten für Fehlermeldungen können jedoch einfach beschafft und zur Inferenz von Regulären Ausdrücken herangezogen werden.

Abbildung 1.6 zeigt die Nutzerschnittstelle von *RankIE* zur Illustration einer Suchmethode auf Basis semantischer Anreicherung. Die Nutzerinteraktion und Anfrageverarbeitung erfolgt folgendermaßen:

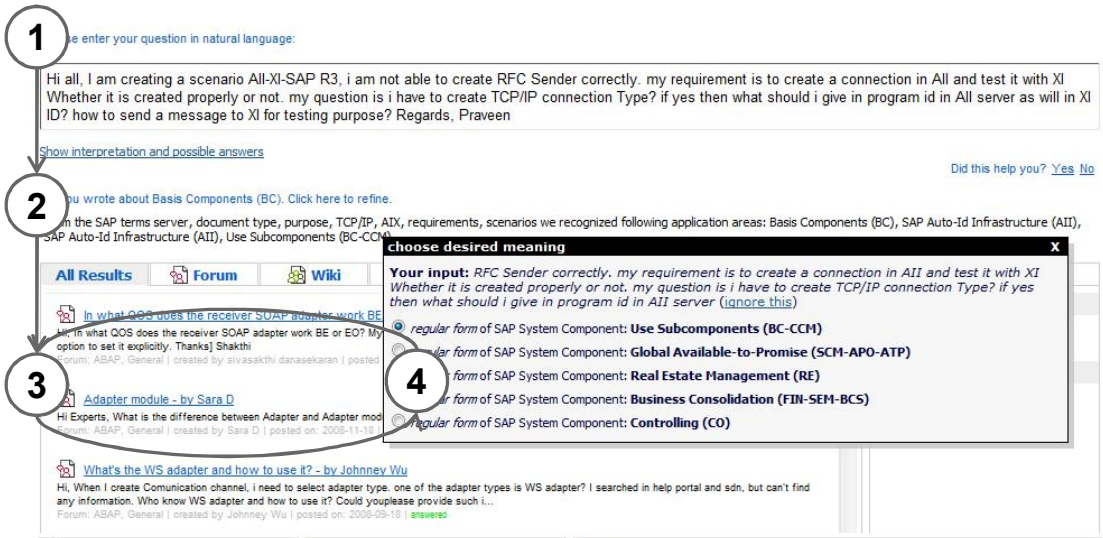


Abb. 1.6: Nutzerschnittstelle von RankIE zur Suche im SCN

RankIE erwartet eine ausführliche Beschreibung des Informationsbedarfes, wie in Abb. 1.6.(1) dargestellt, als Anfrage. Entitäten werden in der Anfrage extrahiert und identifiziert, so dass eine strukturierte Interpretation des Textes mit Hilfe des Domänenwissens erstellt werden kann (siehe Abb. 1.6.(2)). Eine strukturierte Interpretation der Anfrage ist ein Graph, der Entitäten und Beziehungen, die für den Text relevant sind, beschreibt. In der Nutzerschnittstelle werden jedoch nur die Entitäten, die die relevantesten Teilgraphen des Referenzdatensatzes subsumieren, angezeigt.

Die strukturierte Repräsentation der Anfrage wird mit Graphen von Dokumenten abgeglichen, die zuvor in gleicher Art und Weise interpretiert wurden. Die Ähnlichkeit von Anfrage- mit Dokumentgraphen bestimmt die Relevanz der Dokumente in der Resultatliste. Das System unterstützt eine iterative Veränderung der Interpretationen, so dass z. B. im Fall, dass Mehrdeutigkeiten nicht entsprechend dem Nutzerwunsch aufgelöst werden konnten, eine Korrektur möglich ist (siehe Abb. 1.6.(4)).

RankIE erlaubt weiterhin föderierte Anfragen an verschiedene unstrukturierte Datenquellen (z.B. Foren, Wikis oder Blogs), wie in Abb. 1.6.(3) zu sehen ist. Einzelne Inhalte der Resultatliste werden mit Hilfe eines Algorithmus zur Identifikation von Beinaheduplikaten auf Basis von erkannten und extrahierten Entitäten gruppiert (publiziert in [MBBL09]).

1.2.3 Szenario III: Syntaktische und semantische Anreicherung für die strukturierte Dokumentsuche und -analyse

Eine weitere Anwendungsklasse im Kontext der *Enterprise Search* ist die strukturierte Anfrage und Analyse von Textdaten durch komplexe ad-hoc Anfragen der Art „*Liste alle Absätze, welche die am häufigsten in Sätzen zusammen auftretenden Fehler und Softwaremodule enthalten*". Im Folgenden wird ein Anwendungsszenario in diesem Bereich vorgestellt und ein Überblick über ein Beispielsystem gegeben, das derartige Anfragen unterstützt (publiziert in [BBM+09b]).

Um verschiedenste ad-hoc Anfragen für verschiedenste Korrespondenzen von Entitäten, Merkmalen und Dokumentstrukturen zu unterstützen, können Dokumentstrukturen, aber auch Merkmale wie Verben oder Adjektive und Entitäten extrahiert und auf Basis von Nutzerhypothesen zur Laufzeit in Beziehung gesetzt werden. Hypothesengestütze Anfragen zur Analyse von strukturierten Daten werden im Bereich des *Online Analytical Processing (OLAP)* eingesetzt. Die im Kontext dieser Arbeit vorgeschlagene Extraktionsplan-basierte IE-Plattform ersetzt die normalerweise in OLAP-Systemen eingesetzten *Extract, Transform, Load*-Verfahren [VSS02].

Anwendungsfall: Analyse von Kundenanfragen

Der Mehrwert der strukturierten, analytischen Anfrage von extrahierten Daten wird im Folgenden am Beispiel der Analyse von Dokumenten des *SAP Customer Support Network (CSS)* motiviert. Das CSS ist eine Plattform, die es Kunden erlaubt, Unterstützung von der SAP-Kundenbetreuung anzufordern. Sobald der Kunde eine Nachricht abgeschickt hat, wird diese durch die Kundenbetreuung an einen Experten für das entsprechende Softwaremodul weitergeleitet. Die Auswertung der Kundennachrichten mit einer Anfrage wie „*Liste die am häufigsten vorkommenden Kombinationen von Softwaremodulen und Fehlernachrichten*" kann z. B. eine Entscheidungsunterstützung bei der Priorisierung von möglichen Produktentwicklungen bieten.

Ein Beispiel für eine Kundenanfrage ist in Bsp. 1.1 in Form von *XML* dargestellt. Eine Extraktionsplan-basierte Technologie zur Komposition von anwendungsspzifischen IE-Systemen kann eine effiziente Implementation dieses Anwendungsfalls wie folgt unterstützen:

Ein generisches IE-Modul zum Parsen von XML kann zur Extraktion von Entitäten, wie Produktname und -version sowie der Freitextinhalte konfiguriert werden. Die Ausgabe des IE-Moduls zur Extraktion von Freitextabschnitten kann mit Modulen zur Extraktion von Textstrukturen wie Absätzen und Sätzen verbunden werden. Zur Unterstützung von Anfragen hinsichtlich unbekannter Beziehungen sind weiterhin Verben oder zur Analyse von Kundenmeinungen Adjektive von Interesse, die aus Sätzen mit Hilfe eines Moduls zur Wortklassifizierung extrahiert werden. Zur Identifikation von nicht explizit in Freitextabschnitten genannten SAP-Softwaremodulen kann das in Kap. 1.2.2 diskutierte Verfahren mit dem Referenzdatensatz SAPTerm genutzt werden. Zur Extraktion von Fehlermeldungen (z. B. „RSR_LAUNCH_EXCEL047" in Bsp. 1.1.(13)) sind, wie oben diskutiert, generierte Regeln geeignet.

```
1 <Title><![CDATA[Fehler in Transaktion RRMX: RSR_LAUNCH_EXCEL047]]></Title>
2 <Product>SAP NETWEAVER</Product>
3 <ProductVersion>SAP NETWEAVER 7.0</ProductVersion>
4 <TextSection Type="99" Time="00000000000000" Title="Problem Description">
5 <![CDATA[Beim Starten des BEx Analyzer über die Transaktion RRMX wird zwar
6 Excel gestartet, es wird aber nicht automatisch eine Verbindung zum
7 BI–System hergestellt. Zusätzlich wird im GUI ein Fehlerprotokoll
8 ausgegeben, das mit der Fehlermeldung RSR_LAUNCH_EXCEL047 – Fehler
9 beim Start des RFC Servers: '3' beginnt (siehe Anlage 1)...
10 </TextSection>
```

Bsp. 1.1: Kundenanfrage in XML-Darstellung

Anfrage und Navigation auf extrahierten Daten

Im Folgenden erläutern wir ein Anfrageszenario, das den Mehrwert des Systems zur strukturierten Dokumentsuche und die Mächtigkeit von OLAP-Operationen wie *Slice*, *Dice*, *Drill Down/ Up*, *Roll Up* und *Pivot* zur Analyse von unstrukturierten Daten illustriert. Abbildung 1.7 visualisiert eine Sequenz von aufeinander aufbauenden Anfragen für das oben beschriebene Szenario zur Analyse von Fehlern und Softwaremodulen.

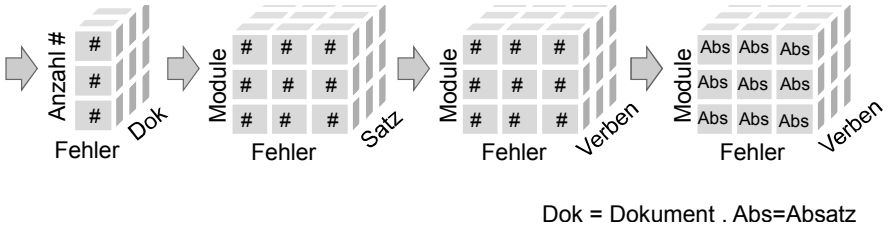


Abb. 1.7: Beispiel für die strukturierte Anfrage und Navigation zur Dokumentsuche

Ein Nutzer startet mit dem Auflisten der häufigsten Fehlermeldungen. Er wählt dazu die Dimension „Fehler" und „Dokumentstruktur", wobei für die Berechnung der „Anzahl" die Anzahl der Dokumente als Teil der „Dokumentstruktur" gewählt wird, da Mehrfachvorkommen eines Fehlers in einem Dokument nicht relevant sind. Er beschränkt die Analyse auf die häufigsten Fehler (*Slice*) und erweitert die Anfrage um Softwaremodule, die gemeinsam mit diesen Fehlermeldungen in Sätzen auftreten (*Pivot / Drill Down* in Dokumentstruktur). Wiederum kann die Suche auf die häufigsten gemeinsamen Vorkommen eingeschränkt werden (*Slice*).

Um genauere Resultate zu erhalten, wird die Anfrage um die Dimension „Verb" erweitert. Der Nutzer wählt (für Englische Kundenmeldungen) Verben wie „throws" und „get" in dem Ergebnis aus, die er im Kontext von Fehlermeldungen erwartet, um die Ergebnismenge einzuschränken. Um sich einen detaillierten Überblick über Nutzerprobleme zu verschaffen, verwendet er die *Roll Up*-Operation, um Textinhalte zu erhalten, und ein *Drill Up*, so dass er die Absätze, in denen die ausgewählten Fehler und Softwaremodule vorkommen, begutachten kann.

Als Ergebnis des Anfrageprozesses erhält der Nutzer eine Liste der häufigsten Fehlermeldungen, der betroffenen Softwaremodule und Abschnitte, in denen das Auftreten des Fehlers textuell beschrieben wird. Die Erstellung einer solchen Liste ohne die Unterstützung des Beispielsystems und einer Extraktionsplan-basierten Plattform ist enorm aufwändig.

1.2.4 Zusammenfassung

In Kap. 1.2 wurden Anwendunsszenarien im Bereich der *Enterprise Search* diskutiert, die durch die Informationsextraktion (IE) profitieren oder durch eine Informationsextraktion überhaupt erst ermöglicht werden. Insbesondere wurde die Notwendigkeit einer Extraktionsplan-basierten IE-Plattform zur flexiblen Komposition von IE-Modulen zu spezifischen IE-Systemen motiviert, so dass eine effiziente Entwicklung unterstützt wird.

Zur effizienten Entwicklung von IE-Systemen sind im Unternehmenskontext weiterhin generische IE-Module hilfreich, die vorhandene strukturierte Daten auf effektive Art und Weise ausnutzen. Es wurden Anwendungsfälle beschrieben, in denen eine automatische Generierung von Regeln auf Basis von gegebenen Beispielinstanzen eine deutliche Reduktion des manuellen Entwicklungsaufwands verspricht. Weiterhin diskutierten wir Anwendungsfälle, in denen eine Methode, welche Graph-strukturierte Referenzdaten zur Identifikation von Entitäten ausnutzt, einen echten Mehrwert bietet.

2

Stand der Technik und offene Forschungsfragen

Im Folgenden werden Problemstellungen, die sich aus den Anwendungsfällen in Kap. 1.2 ergeben, bekannten Ansätzen aus dem Bereich der Informationsextraktion gegenübergestellt und offene Forschungsfragen abgeleitet.

Das Ziel dieser Analyse ist nicht die detaillierte Abgrenzung von einzelnen Ansätzen zu den in dieser Arbeit vorgestellten Verfahren. In Kap. 3.4, Kap. 4.3 und Kap. 5.3 betrachten wir die Unterschiede und Gemeinsamkeiten verschiedener Verfahren zu den in dieser Arbeit entwickelten Ansätzen aus der Perspektive der Algorithmik im Detail. Vielmehr soll hier anhand einer Einordnung von Ansätzen und zusammenfassenden Beschreibungen die Relevanz aktueller Verfahren für die *Enterprise Search* beurteilt und des Weiteren auf offene Forschungsfragen eingegangen werden.

Bei der Informationsextraktion steht zumeist die Effektivität in der Extraktion im Vordergrund. Es gilt im Allgemeinen, dass je genauer ein anwendungsspezifisches IE-System die Charakteristiken der Domäne erfasst, desto höher ist die Qualität der Extraktion. Aus dieser Tatsache lässt sich schließen, dass je höher die Ausdrucksmächtigkeit der verwendeten Plattform zur Beschreibung anwendungsspezifischer Gegebenheiten und die Möglichkeiten zur Integration von generischen, erprobten Methoden ist, desto einfacher ist eine hohe Effektivität zu erreichen.

Aus diesem Zusammenhang kann wiederum geschlossen werden, dass je effizienter die Implementation eines IE-Systems gestaltet werden kann, desto höher ist in der Regel die zu erwartende Effektivität eines Systems bei gleichem Entwicklungsaufwand. Aus diesem Grund betrachten wir als Ausgangspunkt in diesem Kapitel Methoden (und Plattformen) zur Entwicklung von anwendungsspezifischen IE-Systemen (siehe Kap. 2.1).

Weiterhin ist die Effizienz hinsichtlich der Laufzeit eines IE-Systems in vielen Anwendungsfällen ein weiteres wichtiges Gütekriterium, die eng an die verwendete Entwicklungmethode (bzw. -plattform) gebunden ist. Aus diesem Grund analysieren wir im Anschluss an die Betrachtungen zu möglichen Enticklungsmethoden (und Plattformen) offene Forschungsfragen hinsichtlich der Laufzeitverbesserung von IE-Systemen in Kap. 2.2.

Die Ausnutzung von vorhandenem Domänenwissen (durch generische Methoden) ist ein wichtiger Aspekts hinsichtlich der Anpassung von IE-Systemen für eine bestimmte Domäne. Im Kontext der *Enterprise Search* sind vorhandene strukturierte Daten als Domänenwissen zur

effektiven Extraktion und Identifikation von geschäftsrelevanten Entitäten von besonderer Bedeutung. Ein Überblick zu Ansätzen, welche eine Identifikation von bekannten bzw. eine Extraktion von unbekannten Entitäten unter Ausnutzung strukturierter Daten unterstützen, wird in Kap. 2.3 gegeben.

Zur Einordnung und Bewertung von Ansätzen hinsichtlich ihrer Eignung im Kontext der *Enterprise Search* ziehen wir folgende Anwendungscharakteristiken nach [Sar08] heran:

Komplexität der zu extrahierenden Daten: Das Ziel eines IE-Systems kann die Extraktion oder Identifikation von einfachen Entitäten, binären Beziehungen oder komplexeren Datensätzen wie Tabelleneinträgen bzw. komplexen Graph-strukturierten Daten sein.

Verfügbares Domänenwissen: Anwendungsfälle unterscheiden sich in den Daten, die zum Training oder Testen der Extraktion/ Identifikation zur Verfügung stehen. Zum Beispiel kann zum Entwicklungszeitpunkt eines IE-Systems ein annotierter Trainingskorpus, ein nicht annotierter Dokumentenkorpus inklusive Beispielentitäten oder lediglich eine Menge von Beispielentitäten vorliegen. Stehen wenige oder keine Trainingsdaten zur Verfügung, können andere Arten von Domänenwissen, z. B. in Form von manuell durch Domänenexperten erstellten Regeln herangezogen werden.

Anwendbarkeit und Ausdrucksmächtigkeit: Ansätze zur Entwicklung von anwendungsspezifischen IE-Systemen unterscheiden sich hinsichtlich ihrer Eignung für verschiedene Texttypen. Beispiele für verschiedene Texttypen sind: (semi-)strukturierter Text (z. B. Webseiten mit mehr oder weniger festem Schema), unstrukturierte Datensätze (z. B. Adressdatensätze mit unterschiedlicher Formatierung), Fließtexte und irregulär strukturierte Texte, die eine Kombination von Fließtext, Stichpunkten und Tabellen enthalten (z. B. Rechnungen, Foreneinträge oder elektronische Präsentationen). Die Eignung für verschiedene Texttypen ist von der Ausdrucksmächtigkeit der verwendeten Methoden abhängig. Einige Methoden unterstützen z. B. nur (semi-)strukturierten Text, wogegen die Ausdrucksmächtigkeit von proprietären, monolithischen Systemen nur von der verwendeten Programmiersprache begrenzt wird.

Grad der Wiederverwendbarkeit und Automatisierung: Verschiedene Verfahren zur Entwicklung von anwendungsspezifischen IE-Systemen bieten einen unterschiedlichen Grad an Wiederverwendbarkeit von Extraktionsmethoden bzw. Automatisierung in der Entwicklung von Extraktionsmethoden. Zum Beispiel können Extraktionsregeln manuell erstellt oder automatisch (z. B. anhand eines Trainingskorpus) erlernt werden. Im Allgemeinen gilt, je höher die Ausdrucksmächtigkeit einer Methode (siehe oben), desto geringer der Grad der Wiederverwendbarkeit oder Automatisierung.

Unterstütze Merkmalstypen: Je nach Anwendungsfall können verschiedenste Merkmale zur Extraktion oder Identifikation von Daten verwendet werden. Beispiele für Merkmale sind: die Dokumentstruktur, der grammatikalische Aufbau von Fließtext (z. B. Subjekt, Prädikat, Objekt), morphologische Merkmale (z. B. Wort ist ein Substantiv), einfachere Kontextmerkmale (z. B. Wörter links und rechts einer Entität) oder die Syntax der Entität bzw. das Vorkommen einer Entität in einem Wörterbuch.

Charakteristiken des Extraktionskorpus: Neben der Textstruktur können Charakteristiken des Extraktionskorpus, der zum Extraktionszeitpunkt betrachtet wird, unterschieden werden. Beispiele für die Klassifikation eines Extraktionskorpus sind: Einzeldokument (z. B. im Falle von Extraktionsdiensten), statischer Dokumentenkorpus (feste Anzahl von unveränderlichen Dokumenten), veränderlicher Korpus (Dokumente werden editiert) und nicht abgeschlossener Korpus (es können in der Praxis nie alle Dokumente analysiert werden wie z. B. im WWW).

2.1 Effiziente Entwicklung von Systemen zur Informationsextraktion

Wir unterscheiden im Folgenden sechs verschiedene Systemklassen, wobei wir uns an [SDNR07, Sar08, KLR+08] orientieren. Sie unterscheiden sich hinsichtlich ihrer Annahmen

bezüglich der Extraktionslogik und damit der Anwendbarkeit und Ausdrucksmächtigkeit. Wir unterscheiden:

- Formalisierende Ansätze
  - Web-Wrapper
  - Statistische Maschinelle Lernverfahren
  - Regel-basierte Methoden
- Integrierende Ansätze
  - Extraktionsplan-basierte Methoden
  - Workflow-basierte Ansätze
  - Monolithische Systeme

Weiter oben gelistete Methoden unterstützen einen höheren Grad an Wiederverwendung von Extraktionslogik bzw. gar eine automatische Erzeugung der anwendungsspezifischen Extraktionslogik auf Basis von annotierten Trainingskorpora. Weiter unten gelistete Methoden sind durch geringere Annahmen bezüglich der Extraktionslogik gekennzeichnet und sind demnach für eine breitere Klasse von Anwendungsfällen einsetzbar. Je weniger Annahmen bezüglich der Extraktionslogik gemacht werden, desto geringer ist in der Regel die *Wiederverwendbarkeit/ Automatisierung*.

Web-Wrapper sind auf die Extraktion aus (semi-)strukturierten Dokumenten beschränkt. Je nach ausgewählten *Statistischen Maschinellen Lernverfahren* wird eine sequentielle und/ oder Vektorartige Merkmalskombination angenommen. *Regel-basierte Methoden* sind bezüglich der Ausdrucksmächtigkeit durch die Regelgrammatik in der Merkmalsbeschreibung eingeschränkt. *Extraktionsplan-basierte Methoden* folgen gewissen Annahmen hinsichtlich der Schnittstellen zwischen Komponenten im System. *Workflow-basierte Ansätze* gehen davon aus, dass ein Extraktionsprogramm in verschiedene Aufgaben zerlegt werden kann. Ein monolithisches System unterliegt zum Beispiel keinerlei Annahmen. Die Extraktionslogik muss jedoch im vollem Umfang programmatisch beschrieben werden und ist häufig schlecht wiederverwendbar.

Web-Wrapper, Statistische Maschinelle Lernverfahren und Regel-basierte Methoden formalisieren Domänenwissen auf bestimmte Art und Weise. Es wird durch eine generische Extraktionslogik interpretiert und auf einen Text angewendet. Extraktionsplan-basierte Methoden, Workflow-basierte Ansätze und monolithische Systeme sind zur Integration von verschiedenen Komponenten, die Teile der Extraktionslogik kapseln, geeignet. Dementsprechend können sie Methoden der vorherig beschriebenen Verfahren integrieren und mit anwendungsspezifischer Logik kombinieren.

Im Folgenden werden Vor- und Nachteile der hier eingeführten Ansätze im Kontext der *Enterprise Search* bewertet. In Kap. 2.1.1 führen wir die Methoden ein, die Domänenwissen formalisieren und diskutieren, warum sie jeweils nicht als alleinige Grundlage zur Informationsextraktion für reale Anwendungsfälle z. B. im Bereich der *Enterprise Search* dienen können (siehe auch [SDNR07]). In Kap. 2.1.2 diskutieren wir Vor- und Nachteile von integrativen Ansätzen und betrachten im Detail Extraktionsplan-basierte Ansätze, zu dem die hier vorgestellte *RapidUIM*-Plattform gehört.

2.1.1 Formalisierende Ansätze

Formalisierende Ansätze zur Entwicklung von IE-Systemen sind die manuelle oder automatische Erzeugung von Grammatiken für *Web-Wrapper*, Modellbildung auf Basis *Statistischer Maschineller Lernverfahren* und die manuelle oder automatische Erstellung von *Extraktionsregeln*. Jede Methode hat bestimmte Vorzüge hinsichtlich gewisser Teilprobleme bei der Informationsextraktion im Unternehmenskontext. Sie sind daher eher als Grundlage zur Entwicklung von IE-Systemen geeignet. Im Folgenden werden die einzelnen Ansätze eingeführt und Einschränkungen der verschiedenen Methoden als alleinige Technologie zur Informationsextraktion im Unternehmenskontext aufgezeigt.

Web-Wrapper

Für (semi-)strukturierte Textdaten wie z. B. HTML-Dokumente, die aus Datenbanken generiert werden, sind *Web-Wrapper* [BGG09] eine geeignete Extraktionstechnologie. Sie bilden im Allgemeinen die Dokumentstrukturen ab, die relevante Entitäten umschließen. Zur Entwicklung von derartigen *Wrappern* stehen Werkzeuge zur visuellen oder deklarativen Umschreibung von Dokumentstrukturen zur Verfügung (z. B. [BFG01, PRÁ+02]).

Wrapper können auch automatisch generiert werden, sofern ein Korpus von (annotierten) Trainingsdokumenten zur Verfügung steht. Ein Überblick zu Ansätzen im Bereich der (semi-)automatischen *Wrapper-Induktion* ist in [TAC06, CKGS06, BGG09] zu finden. Deren Ziel ist es, Muster in der Abfolge von statischen und variablen Inhalten der Dokumente zu erkennen und die Grammatik der verwendeten Dokumentenvorlagen abzuleiten.

Die verschiedenen Ansätze in der Wrapper-Induktion unterscheiden sich in der Ausdrucksstärke, also in den Strukturen, die erlernt/ beschrieben werden können, wie z. B. einfache Tabellen, Tabellenstrukturen, die mehrere Seiten umfassen, ineinander verschachtelte Dokumentstrukturen etc. *Wrapper* sind grundsätzlich nicht auf HTML-Dokumente beschränkt. Anwendungsfälle, in denen *Wrapper* als IE-Methoden im *Enterprise Search*-Kontext geeignet sind, treten jedoch äußerst selten auf, da diese auf die Extraktion aus Dokumenten spezialisiert sind, die als Träger für Inhalte einer strukturierten Datenquelle dienen. Auf strukturierte Datenquellen kann in einem Unternehmenskontext zumeist direkt zugegriffen werden.

Als Basis zur Entwicklung von IE-Systemen für die hier betrachteten Anwendungsfälle sind *Wrapper* ungeeignet, da Strukturen in Dokumenten wie Rechnungen, Kundenanfragen, Präsentation oder Forendaten häufig nicht durch die unterstützten Typen von Grammatiken in geeigneter Art und Weise beschrieben werden können.

Statistische Maschinelle Lernverfahren

In der Literatur werden verschiedene Modelle für das statistische maschinelle Lernen zur Informationsextraktion auf Basis von Trainingskorpora beschrieben. Viele Ansätze verwenden *Hidden Markov Models (HMM)*, *Maximum Entropy Markov Models (MEMM)* oder *Conditional Random Fields (CRF)*, wobei CRF heutzutage das am häufigsten verwendete Modell ist (siehe z. B. [TAC06, NS07, Sar08] für einen Überblick).

Populäre Plattformen zur Entwicklung von anwendungsspezifischen IE-Systemen sind in diesem Bereich z. B. *LingPipe* [Car04], *Stanford NLP Toolkit*[1] oder *Balie* [Nad05]. Die Implementation von IE-Systemen auf Basis von Statistischen Maschinellen Lernverfahren ist für Entwickler insofern transparent, als dass sich die Problemstellung meist zur Definition von (geeigneten) Merkmalen und der Bereitstellung eines annotierten Trainingskorpus reduziert.

Basis von Statistischen Maschinellen Lernverfahren ist die statistische Verteilung von nutzerdefinierten Merkmalen in einen Trainingskorpus. Aus positiven und negativen Beispielen für Merkmale im Kontext einer Entität wird ein stochastisches Modell abgeleitet, dass die Kombination und/ oder Sequenz von Merkmalen für das Auftreten eines Entitätstyps (z. B. signifikante Wörter im Kontext) in einem Textsegment (normalerweise Sätze) beschreibt.

Wenn ein Wörterbuch mit Beispielentitäten und ein nicht annotierter Dokumentenkorpus verfügbar ist, kann der Dokumentenkorpus mit dem Wörterbuch annotiert und das Modell initial trainiert werden. Mit Hilfe der (initial) extrahierten Daten kann das Wörterbuch erweitert und das statistische Modell iterativ angepasst werden (ein so genanntes *Bootstrapping* [NS07, Sar08]).

Statistische Maschinelle Lernverfahren sind der de facto Standard zur Extraktion von Entitäten in Fließtext in Anwendungsfällen, in denen ein großer (annotierter) Trainingskorpus verfügbar ist. Häufig sind aber zum Entwicklungszeitpunkt von IE-Systemen für Unternehmensanwen-

[1] `http://nlp.stanford.edu/software/index.shtml`, Stand: 01.05.2010

dungen lediglich wenige Besipieldokumente vorhanden. Zum Teil können erlernte Modelle für Entitätstypen von allgemeinem Interesse, wie z. B. Personen, Organisationen und Ortsangaben mit akzeptablen Qualitätsverlusten in anderen Anwendungsfällen wiederverwendet werden [NS07, Sar08].

Statistische Maschinelle Lernverfahren sind daher ein wichtiger Baustein von IE-Systemen für die *Enterprise Search*, können aber nicht als alleinige Basis dienen, da reine Fließtexte im *Enterprise Search*-Bereich nur in gewissem Umfang auftreten und Trainingsdaten für anwendungsspezifische Entitätstypen in der Regel nicht im ausreichenden Umfang vorhanden sind.

Weiterhin ist die Fehlersuche in und die manuelle Feinabstimmung von Extraktionsmethoden, die auf statistischen Modellen basieren, schwierig [NS07, Sar08]. Insbesondere in geschäftskritischen Anwendungen ist die Möglichkeit zur wenig aufwändigen Fehlersuche und Modifikation der Extraktionsmechanismen notwendig oder zumindest wünschenswert.

Regel-basierte Methoden

Die Fehlersuche und Feinabstimmung ist bei der Verwendung von Regel-basierten Methoden, die entsprechende Expertise vorausgesetzt, komfortabel, da sie zur manuellen Beschreibung von Extraktionsmechanismen entwickelt wurden. Einige Systeme sind auf Fließtext spezialisiert (z. B. [Ril93, KM95, Huf95, CBG99, CM99, Yan03]). Regeln werden zum Beispiel basierend auf logischen Ausdrücken erstellt, die die grammatikalische Struktur von Sätzen oder andere Merkmale im Kontext der Entitäten berücksichtigen.

Viele aktuelle Regel-basierte Systeme verwenden kaskadierende Grammatiken (z. B. *CSPL* [AHB+93] oder *JAPE* [CMBT02]). Basierend auf Regulären Ausdrücken zur Beschreibung von Entitäten (z. B. „⟨Person⟩:[A-Z]\. [A-Z][a-z]$^+$“) und/ oder deren Kontext, sowie darauf aufbauenden komplexeren Regeln (z. B. „⟨Anstellung⟩:⟨Person⟩ ‘ist angestellt bei’ ⟨Organisation⟩“) können anwendungsspezifische IE-Systeme entwickelt werden. Kaskadierende Grammatiken sind grundsätzlich auch zur Extraktion aus (semi-)strukturierten Dokumenten wie *HTML* oder *XML* geeignet. Aktuell häufig eingesetzte Regel-basierte Systeme sind z. B. *TextMarker* [KAP09], *ANNIE* [MC03], *IBM Language Ware* [2] oder *BusinessObjects Text Analysis*[3].

Regelsprachen abstrahieren von der konkreten Implementation des Extraktionssystems und erlauben eine effiziente Entwicklung von IE-Systemen für Nutzer mit entsprechender Expertise. In der Literatur werden weiterhin Verfahren diskutiert, die Regeln auf Basis von annotierten Trainingsdokumenten erlernen (siehe z. B. [Mus99, TAC06, Sar08] für einen Überblick). Ansätze zur Regelinferenz unterstützen die Extraktion aus (semi-)strukturiertem Text und Fließtext. Im Allgemeinen haben sich jedoch Statistische Maschinelle Lernverfahren im Bereich des maschinellen Lernens von annotierten Dokumentenkorpora durchgesetzt.

Die Ausdrucksmächtigkeit von Regeln hinsichtlich irregulär strukturierter Dokumente wie z. B. Rechnungen ist jedoch begrenzt. In der imperativen Programmierung relativ einfach zu beschreibende Aufgaben, wie eine anwendungsspezifische Vorsegmentation des Dokuments (z. B. für Rechnungen), die Rekursion in verschachtelten Dokumentstrukturen oder Traversierung von Graph-strukturierten Referenzdatensätzen zur Identifikation von Entitäten können nicht beschrieben werden. Daher können Regel-basierte Systeme nur bestimmte Teilprobleme von Anwendungsfällen im *Enterprise Search*-Bereich abbilden und sind wie Statistische Maschinelle Lernverfahren lediglich als ein Baustein für einen allgemeineren Ansatz zur Entwicklung von anwendungsspezifischen IE-Systemen im Unternehmenskontext zu sehen.

[2] `http://www.alphaworks.ibm.com/tech/lrw`, Stand: 01.08.2010

[3] `http://www.sap.com/solutions/sapbusinessobjects/large/eim/textanalysis/index.epx`, Stand: 01.08.2010

2.1.2 Integrative Ansätze

Um formalisierende Ansätze und anwendungsspezifischen Programmcode zu integrieren, wurden verschiedene Ansätze entwickelt, die wir im Folgenden diskutieren. Die Grundidee ist, dass oben erläuterte Ansätze in den meisten Anwendungsfällen Teilprobleme lösen, aber oft mit weiteren anwendungsspezifischen Komponenten kombiniert werden müssen, um IE-Systeme entwickeln zu können, die den gegebenen Anforderungen in vollem Umfang gerecht werden [SDNR07].

Monolithische Systeme

Eine naheliegende Lösung, um verschiedene generische Extraktionsmethoden zu kombinieren, ist deren Komposition mit anwendungsspezifischen Programmcode zu einem monolithischen IE-System. Der Vorteil einer derartigen Komposition ist, dass weder der Datenaustausch zwischen den einzelnen Komponenten, noch deren Interaktionsmuster bestimmten Annahmen unterliegt und demnach die mögliche Extraktionslogik lediglich durch die verwendete Programmiersprache selbst eingeschränkt wird. Die Wartbarkeit und Wiederverwendbarkeit von Komponenten monolithischer IE-Systeme ist jedoch schwierig, so dass dieser Entwicklungsansatz in der Praxis nur noch selten eingesetzt wird [FL04].

Workflow-basierte Ansätze

Aktuell sind Workflow-basierte Ansätze für die Entwicklung von anwendungsspezifischen IE-Systemen im Kontext der *Enterprise Search* der de facto Standard. Beispiele für Workflow-basierte IE-Plattformen sind *ATLAS* [LFGP02], *GATE* [CMBT02] oder *UIMA* [FL04]. Workflow-basierte Ansätze setzen voraus, dass die Extraktionslogik in aufeinander folgende, unabhängige Teilaufgaben zerlegt werden kann.

Weiterhin müssen notwendige Datenformate und der Datenaustausch zwischen einzelnen Komponenten entsprechend den Anforderungen in dem *Application Programming Interface (API)* abgebildet werden können. In den meisten Anwendungsfällen in der *Enterprise Search* ist dies ohne weiteres möglich. Es gibt jedoch Einschränkungen. Zum Beispiel ist die Abbildung von Dokument-übergreifenden Operationen in UIMA per se nicht vorgesehen.

Workflow-basierte Ansätze haben den Vorteil, dass einzelne IE-Module für generische oder anwendungsspezifische Zwecke relativ unabhängig voneinander entwickelt und getestet werden können. Verschiedene IE-Module werden zu einem Workflow, der das IE-System repräsentiert, durch eine entsprechende Konfiguration (z. B. in XML-Form in UIMA) verbunden. Die Wiederverwendbarkeit von Komponenten und Effizienz in der Entwicklung von IE-Systemen ist jedoch aufgrund einiger Nachteile, die im Folgenden erörtert werden, nicht befriedigend wie aktuelle Erfahrungsberichte im Umfeld von UIMA, wie z. B. [KNS+08, HBL+08] zeigen.

Ein Nachteil von Workflow-basierten Systemen ist, dass die Komposition von IE-Modulen im Allgemeinen lediglich einem sequentiellen Kontrollfluss von Modulaufrufen entspricht, deren Datenaustausch nicht explizit modelliert wird. Der Datenfluss zwischen einzelnen IE-Modulen basiert auf globalen Datencontainern [SDNR07], so genannten *„Blackboards“*, in denen IE-Module Datenobjekte, die für ein Dokument relevant sind, zwischenspeichern. Datencontainer für die verschiedenen Objekttypen müssen durch nachgelagerte Module adressiert und entsprechend der Objektstruktur und -semantik verarbeitet werden. Eine ad-hoc Komposition verschiedener nicht aufeinander abgestimmter IE-Module ist daher nicht möglich, wodurch die Effizienz in der Entwicklung von IE-Systemen stark eingeschränkt wird.

Extraktionsplan-basierte Methoden

Aktuelle Ansätze in der Informationsextraktion [KLR+08,SDNR07,BFBS10] versuchen die Freiheitsgrade von Workflow-basierten Systemen mit der effizienten Entwicklung, wie sie etwa in Regel-basierten Methoden anzutreffen ist, zu kombinieren. Diesen Ansätzen ist gemein, dass IE-Programme, die in verschiedenen Sprachen beschrieben werden, in Ausführungspläne übersetzt werden, die die Abhängigkeiten zwischen Daten und nicht die (sequentielle) Abarbeitung von Aufgaben, spezifizieren. Die Modellierung von Datenabhängigkeiten und/oder Nebenbedingungen unterstützt weiterhin neuartige Verfahren zur Optimierung von IE-Programmen. Im Folgenden werden die Ansätze von [KLR+08, SDNR07, BFBS10] diskutiert und bewertet.

SystemT. Im Kontext des SystemT-Projektes [KLR+08,RRK+08] werden Extraktionspläne mit AQL, einer an SQL angelehnten Sprache, beschrieben. In Bsp. 2.1 ist ein Beispielprogramm zur Extraktion von Musikern und zugehörigen Instrumenten in AQL-Syntax dargestellt. Primitive Operatoren (Reguläre Ausdrücke, Wörterbücher) und ein Operator zum Verbund von Zwischenergebnissen zu Relationen können mit Standardoperatoren der Relationalen Algebra kombiniert werden, wodurch die Ausdrucksmächtigkeit zum Beispiel gegenüber Regelbasierten Systemen vergrößert wird. Auch die Einbettung von nutzerdefinierten Funktionen für anwendungsspezifische Operationen wird in [KLR+08] diskutiert (jedoch bisher nicht realisiert bzw. publiziert).

```
 1 SELECT  name.match as name,    //  Personenname
 2         instrument.match as instr, //Instrument
 3         // Beziehung von Personennamen und Instrumenten (''spielt'')
 4         CombineSpans(name.match, instrument.match) as instr_2_name
 5 FROM      // Personenname: Regulärer Ausdruck
 6         Regex(/[A-Z]\w+(\s[A-Z]\w+)?/, DocScan.text) name,
 7           // Instrument: Wörterbuch
 8         Dictionary("instr.dict", DocScan.text) instrument
 9 WHERE     // Person u. Instrument im Kontext von 30 Zeichen
10         Follows(0, 30, name.match, instrument.match);
```

Bsp. 2.1: Extraktionsplanbeschreibung mit AQL

CIMPLE. Im Kontext des CIMPLE-Projektes [SDNR07, SDM+08] wurde XLog, eine an Datalog angelehnte Sprache, zur Implementation von anwendungsspezifischen IE-Programmen entwickelt. Schnittstellen von beliebigen IE-Modulen werden als Prädikate in XLog eingebettet. Nebenbedingungen, wie z. B. der Abstand zwischen zwei Entitäten zur Extraktion von Beziehungen, können über prozedurale Funktionen deklariert werden.

Ein Beispiel für einen Extraktionsplan in Xlog-Syntax ist in Bsp. 2.2 abgebildet (vgl. [SDM+08]). Das IE-Programm („q(x,p,a,h)“) extrahiert Attribute von Häusern („extractHouses(x,p,a,h)“) aus einer Menge von Webseiten („housePages(x)“) mit einem Preis p kleiner 5.000.000$ und einer Grundstücksgröße a größer als 4500 m^2. Die aus den Webseiten extrahierten Schulen werden mit bevorzugten Schulen („extractSchools(y,s)“) aus anderen Webseiten („schools(s)“) mit Hilfe eines approximativen Zeichenkettenvergleichs („approxM(h,s)“) abgeglichen.

```
houses(x,p,a,h) :- housePages(x), extractHouses(x,p,a,h)
schools(s):- schoolPages(y), extractSchools(y,s)
q(x,p,a,h):- houses(x,p,a,h), schools(s), p<500000, a>4500, approxM(h,s)
```

Bsp. 2.2: Extraktionsplanbeschreibung mit Xlog

Ein mit Xlog beschriebener Extraktionsplan wird, vergleichbar einem AQL basierten IE-Programm, in einen kostenoptimierten Ausführungsplan übersetzt (siehe Kap. 3.4 für Details). Die Unterstützung der Komposition von generischen und anwendungsspezifischen IE-Modulen erfüllt die Anforderungen an einen Ansatz zur Entwicklung von IE-Systemen im Unternehmenskontext hinsichtlich der effizienten Entwicklung und Ausdrucksmächtigkeit in weitaus höherem Umfang als bisher diskutierte Ansätze.

RapidUIM. Eine vergleichbare IE-Plattform, welche im Kontext der vorliegenden Arbeit entwickelt wurde, wurde in [BBH+09, BFBS10] diskutiert. Die so genannte *RapidUIM*-Plattform unterstützt die visuelle Modellierung von Extraktionsplänen, welche den Datenfluss zwischen IE-Modulen beschreibt, und erlaubt die Spezifikation von Modulparametern.

Der Exraktionsplan entspricht in *RapidUIM* einem möglichen Ausführungsplan in den oben genannten Ansätzen. In Abb. 2.1 ist ein einfaches Beispiel für einen Extraktionsplan zur Dokumentverarbeitung für das Szenario II aus Kap. 1.2.2 dargestellt. Links in Abb. 2.1 sind Parameter eines Regel-basierten IE-Modules, mittig der Datenfluss zwischen IE-Modulen und rechts die Teilmodule eines komplexen, komponierten IE-Moduls dargestellt.

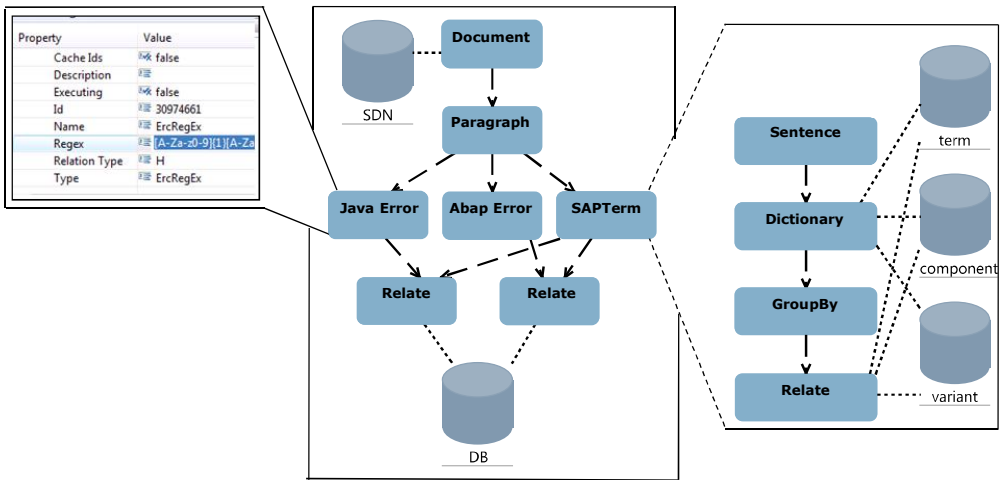


Abb. 2.1: Beispiel für einen Extraktionsplan in RapidUIM

RapidUIM ermöglicht eine effiziente Komposition von IE-Modulen zu anwendungsspezifischen IE-Systemen. Arbeiten im Bereich der *MapReduce*-orientierten Datenverarbeitung [ORS+08] argumentieren, dass eine Datenfluss-orientierte Programmsicht wie in *RapidUIM* für komplexe Datenverarbeitungsprobleme häufig intuitiver und leichter verständlich ist, als eine rein deklarative Beschreibung von Programmen wie z. B. in AQL oder XLog.

Weiterhin unterscheidet sich RapidUIM zu einem XLog-basierten System dahingehend, dass IE-Module einheitlichen Schnittstellen folgen und damit eine beliebige Kombination von IE-Modulen unterstützt wird. Die Wiederverwendbarkeit von IE-Modulen wird dadurch wesentlich verbessert. Weiterhin ermöglicht ein Graph-basiertes Datenmodell eine Implementation von generischen IE-Modulen auf Basis von Graphoperationen unabhängig von der Semantik der Daten (siehe Kap. 3.1). Es verbessert dadurch die Effizienz bei der Entwicklung von anwendungsspezifischen Systemen, z. B. im Vergleich zu [SDM+08].

2.1.3 Zusammenfassung

In Kap. 2.1 wurden Ansätze zur Entwicklung von anwendungsspezifischen IE-Systemen hinsichtlich Ihrer Eignung für die Informationsextraktion im Kontext der *Enterprise Search* bewer-

tet: Web-Wrapper adressieren eine Klasse von IE-Anwendungsfällen, die hauptsächlich bei der Extraktion im WWW auftreten. Statistische Maschinelle Lernverfahren sind auf die Extraktion aus Fließtexten spezialisiert. Weiterhin ist die Verfügbarkeit von einem großen Trainingskorpus zur Entwicklungszeit eines IE-Systems erforderlich.

Regel-basierte Methoden können zur Extraktion aus (semi-)strukturierten Text und Fließtext verwendet werden und erlauben eine effiziente Implementation von IE-Systemen, sind aber hinsichtlich ihrer Ausdrucksmächtigkeit durch die Regelgrammatik beschränkt. Workflow-basierte Systeme ermöglichen die „Aneinanderreihung" von spezifischen und generischen IE-Modulen. Die Kommunikation zwischen Modulen erfolgt über *Blackboards*, wodurch eine Wiederverwendung und flexible Komposition nur in gewissem Maße möglich ist. Die Effizienz in der Entwicklung ist geringer als z. B. bei *Regel-basierten Methoden*.

Extaktionsplan-basierte Verfahren modellieren die Abhängigkeiten zwischen Daten. Zwei Ansätze in diesem Bereich kombinieren die Flexibilität von Workflow-basierten Systemen mit einer effizienten Entwicklung, wie sie z. B. in Regel-basierten Ansätzen anzutreffen ist. Einzelne IE-Module können parametrisiert und miteinander kombiniert werden. Die im Kontext des *CIMPLE*-Projektes [SDNR07] entwickelte Methode schlägt eine deklarative, Datalog-basierte Syntax und eine Übersetzung in Ausführungspläne vor. *RapidUIM* [BBH+09, BFBS10] verfolgt den Ansatz der visuellen Komposition des Datenflusses zwischen IE-Modulen. Es unterstützt durch einheitliche Schnittstellen und ein Graph-basiertes Meta-Datenmodell eine Wiederverwendung von IE-Modulen in höherem Maße als [SDNR07] und ist daher besser als Plattform zur Entwicklung von IE-Systemen im Kontext der *Enterprise Search* geeignet. Details zur *RapidUIM*-Plattform werden in Kap. 3 diskutiert.

2.2 Effiziente Ausführung von Systemen zur Informationsextraktion

Die Steigerung der Effizienz von IE-Systemen, insbesondere hinsichtlich der Dokumentverarbeitungszeiten, ist eine bekannte, in der Literatur häufig beschriebene Problemstellung [Sar08]. Im Folgenden wird ein Überblick zu Methoden zur Laufzeitverbesserung gegeben, die im Bereich der Informationsextraktion für die *Enterprise Search* angewendet werden können. Weiterhin stellen wir offene Forschungsprobleme heraus.

Verfahren zur Verbesserung von Laufzeiten basieren auf verschiedenen Charakteristiken von Anwendungsklassen. Geeignete Dimensionen zur Klassifikation von Methoden zur Laufzeitverbesserung sind die Komplexität des Extraktionsprogramms und die Charackteristiken des zu analysierenden Korpus. In Abb. 2.2 sind die Dimensionen, die im Folgenden zur Klassifikation verwendet werden, mit einer Umschreibung des Verfahrens zur Laufzeitverbesserung dargestellt.

Die Komplexität der Extraktion reicht von der Extraktion von Entitäten oder Beziehungen eines Typs über die zeitgleiche Extraktion von mehreren Objekttypen (Entitäten oder Beziehungen) zur Extraktion von komplexen syntaktischen und semantischen Beziehungen, die insbesondere in der Informationsextraktion für die *Enterprise Search* häufig im Vordergrund stehen. Korpuscharakteristiken beschreiben den Fokus der Extraktion, also ob die Laufzeitverbesserung für die Extraktion aus einem Dokument, einem statischen Dokumentenkorpus, einem Korpus mit veränderlichen Dokumenten oder einem nicht abgeschlossenen Dokumentenkorpus wie dem WWW angestrebt wird.

Im Folgenden wird ein Überblick über Ansätze entsprechend der Klassifikationskriterien gegeben, deren Relevanz für die Informationsextraktion im Unternehmenskontext diskutiert und offene Forschungsfragen herausgestellt, die im Kontext der Laufzeitoptimierung zur Verarbeitung von Einzeldokumenten in komplexen Extraktionsprogrammen angesiedelt sind (farbig markierter Bereich in Abb. 2.2).

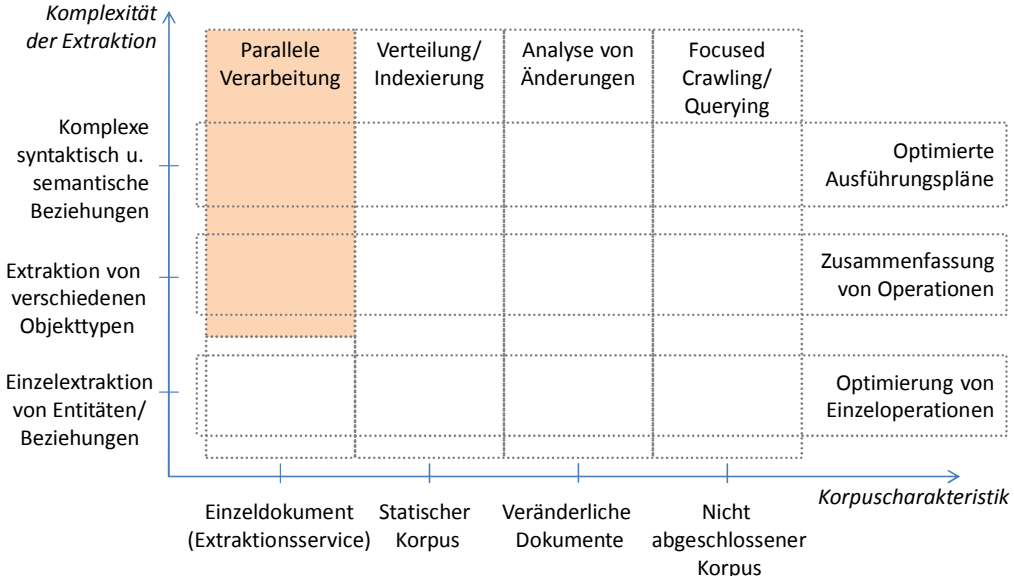


Abb. 2.2: Klassifikation von Ansätzen zur Laufzeitverbesserung von IE-Systemen

2.2.1 Laufzeitverbesserung hinsichtlich der Komplexität der Extraktion

In der Literatur werden verschiedene Verfahren zur Laufzeitverbesserung, die für verschieden komplexe Extraktionsaufgaben geeignet sind, beschrieben. Anzumerken ist, dass jeder, der im Folgenden vorgestellten Ansätze, im Kontext der Informationsextraktion im *Enterprise Search* relevant und anwendbar ist. Details zu ausgewählten Ansätzen werden in Kap. 3.4 vertieft.

Optimierung von Einzeloperationen. Die Optimierung von häufig in der Informationsextraktion auftretenden Einzeloperationen ist sehr gut erforscht. Eine umfangreich in der Literatur diskutierte Problemstellung ist z. B. die Laufzeitoptimierung des exakten oder approximativen Abgleichs von Dokumenttexten und Wörterbucheinträgen (siehe z. B. [MM02, EIV07, PWN06] für einen Überblick). Ansätze, die speziellere Problemstellungen, wie die Reduktion des approximativen Abgleichs zu einem exakten Abgleich oder die Begrenzung der Ergebnismenge beim Abgleich mit Hilfe von Relevanzmetriken untersuchen, sind [WXLZ09, CGX09, CGRM06].

Zur Optimierung der Regel-basierten Extraktion können mehrere Regeln zur Extraktion von einem Entitätstyp in einen gemeinsamen *Deterministischen Endlichen Automaten (DEA)* überführt werden [Sar08], wodurch die Laufzeit wesentlich verbessert wird, da ein Dokumenttext nur genau einmal, anstatt separat für jede Regel, gescannt werden muss. Eine kostenoptimale Auswertung von DEAs wird in [TOKG03] untersucht. [Dom99] und [ZE01] untersuchen eine Kosten-basierte Optimierung von Merkmalen in Statistischen Maschinellen Lernverfahren. Statistische Maschinelle Lernverfahren können zudem sehr gut parallelisiert werden [SCDZ06, RRP+07, HFL+08]).

Zusammenführung von Operationen. In Anwendungsfällen, in denen verschiedene Typen von Entitäten und Beziehungen auf Basis gleicher Operationen extrahiert werden, verspricht eine Zusammenführung der Operationen in vielen Fällen eine Laufzeitverbesserung. Die Grundlage einer derartigen Optimierung liegt in der Charakteristik von Extraktionsverfahren, dass zum einen Operationen auf einem Eingabetext und zum anderen Operationen auf dem Domänenwissen, das zur Extraktion verwendet wird, notwendig sind. Eine Zusam-

menführung des Domänenwissens z. B. von Wörterbüchern für die Extraktion von Personen und Organisationen ermöglicht, dass in einer Textoperation das gesamte Domänenwissen ausgewertet werden kann und nicht mehrere Textoperationen für die Bestandteile des Domänenwissens notwendig sind.

Durch die Vereinigung von Wörterbüchern [RRK+08] ist lediglich ein Scan des Textes notwendig. Treffer werden, entsprechend des Wörterbuchs, in denen sie aufgetreten sind, getypt, so dass eine konsistente Weitervearbeitung der extrahierten Daten durch andere Module möglich ist. Bei der Regel-basierten Extraktion können, anstatt den kompletten Text für jede Regel einzeln zu testen, Regeln zusammgefasst und in einen gemeinsamen *Deterministischen Endlichen Transduktor* überführt werden [Sar08]. Im Unterschied zu oben genannten DEAs geben *Deterministische Endliche Transduktoren* zusätzliche Typinformationen aus.

Optimierte Ausführungspläne. Ansätze im Bereich der Extraktionsplan-basierten Verfahren untersuchen die Evaluierung von verschiedenen Ausführungsplänen zur Kostenoptimierung der Ausführung. Techniken, die oft diskutiert werden, sind die frühzeitige Filterung von Zwischenergebnissen und die Kostenoptimierung der Ausführungsreihenfolge von IE-Modulen unter Ausnutzung von Datenabhängigkeiten [RRK+08, SDNR07].

Zum Beispiel kann bei der Extraktion der Beziehung zwischen Personennamen und Emailadressen in einem Kontext von 100 Buchstaben zuerst die Emailadresse mit einem Regulären Ausdruck extrahiert und lediglich der Kontext von 100 Buchstaben mit Hilfe von einem Namenswörterbuch analysiert werden. Für komplexere Beziehungen oder Blöcke, wie z. B. Adressfelder, gibt es ungleich mehr mögliche Ausführungsfolgen und ein wesentlich höheres Optimierungspotenzial. Eine derartige Vorgehensweise ist mit der Anfrageoptimierung in Datenbanken auf Basis von geschätzten Selektivitäten vergleichbar.

Weiterhin können Zwischenergebnisse durch das (automatische) Einfügen von zusätzlichen Nebenbedingungen im Ausführungsplan frühzeitig gefiltert werden [SDNR07], so dass zum Beispiel bei der Extraktion von Personennamen lediglich Textsegmente analysiert werden, die Wörter mit Großbuchstaben enthalten. Die Optimierung von Ausführungsplänen werden wir in Kap. 3.4 detaillierter untersuchen.

2.2.2 Laufzeitverbesserung hinsichtlich verschiedener Korpuscharakteristiken

Die oben diskutierten Verfahren sind orthogonal zu Verfahren anwendbar, die die Laufzeit von IE-Systemen unter Ausnutzung von Korpuscharakteristiken verbessern. Im Folgenden wird auf Charakteristiken von Extraktionskorpora eingegangen und entsprechende Verfahren zur Laufzeitverbesserung erläutert.

Nicht abgeschlossener Korpus. Wenn ein Dokumentenkorpus extrem groß ist (z. B. das WWW), ist die Ausführung eines IE-Programms für jedes einzelne Dokument oftmals nicht sinnvoll bzw. nicht möglich. Ansätze, die im Bereich des *Focused Crawling* entwickelt wurden [DSC+07, GHY02, CBD99], klassifizieren Dokumente hinsichtlich der Relevanz für die Extraktionsaufgabe und reduzieren damit die Laufzeit durch Einschränkung der zu analysierenden Dokumente erheblich.

Wenn ein Volltextindex für den Dokumentenkorpus existiert (z. B. eine Web-Suchmaschine), können Ansätze aus dem Bereich des *Focused Querying* [GHY02, CBD99] verwendet werden. Es wird eine Sequenz von Stichwortanfragen generiert, um für die Extraktion relevante Dokumente zu erhalten.

Beide oben genannte Verfahren führen zu einer geringeren Vollständigkeit bei der Extraktion/ Identifikation. In der *Enterprise Search* ist ein Korpus häufig abgeschlossen und eine geringe-

re Vollständigkeit bei der Extraktion nicht akzeptabel. Daher ist die Relevanz von derartigen Verfahren im Kontext dieser Arbeit eher gering.

Veränderlicher Dokumentenkorpus. Wenn Dokumente eines Korpus Veränderungen unterliegen, muss ein naiv arbeitendes IE-System periodisch den gesamten Korpus verarbeiten. Um die Aktualität der extrahierten Daten sicherzustellen, muss die Extraktion daher höchst performant geschehen, so dass die Anzahl der nicht aktuellen Resultate minimiert wird.

In der Literatur werden zwei Ansätze zur Steigerung der Effizienz in derartigen Anwendungsfällen diskutiert: Wenn (historische) Änderungsstatistiken für Dokumente vorhanden sind, kann eine optimale Reihenfolge für Dokumentzugriffe berechnet werden [DMSW09], so dass extrahierte Daten eine größere Aktualität bei gleichem Ressourceneinsatz aufweisen oder weniger Ressourcen bei gleicher Aktualität notwendig sind.

Eine weitere Möglichkeit zur Reduktion der Gesamtlaufzeit eines Systems ist die Analyse von Dokumenten auf Änderungen, wenn historische Versionen des Dokuments gespeichert werden, so dass ein Extraktionsprogramm nur auf veränderten Textsegmenten ausgeführt werden muss [CDYR08]. Veränderliche Dokumentinhalte treten in vielen *Enterprise Search*-Szenarien auf. Beide Verfahren sind daher für den Einsatz in der Informationsextraktion für die *Enterprise Search* relevant.

Statischer Dokumentenkorpus. Bei der Extraktion aus einem statischen Korpus werden in der Literatur zwei Klassen von Ansätzen zur Laufzeitverbesserung diskutiert. Wenn eine Vielzahl verschiedener Typen von Entitäten und Beziehungen extrahiert oder ad-hoc „Extraktionsanfragen“ unterstützt werden sollen, kann eine spezielle Indexierung von Texten sinnvoll sein. Spezielle Indexverfahren zur Informationsextraktion unterstützen morphologische Merkmale (siehe [CE05, Agi05]) oder erzeugen Indizes zur effizienten Evaluation von Regulären Ausdrücken auf einem kompletten Dokumentenkorpus [CR02, RBJ06].

Ein allgemeinerer Ansatz für die Laufzeitverbeserung zur Analyse statischer Dokumentenkorpora ist die Verteilung der Dokumentverarbeitung. Ansätze zur verteilten Stapelverarbeitung von Dokumenten sind [DG04, GCG+04, BTMC04, JPS05, ELB07, ORS+08]. Häufig werden mehrere Instanzen eines Extraktionsprogramm auf verschiedenen Systemen ausgeführt. Aktuell werden im Gebiet der Verteilung von IE-Programmen *MapReduce*-Ansätze [DG04, ORS+08] favorisiert, die insbesondere das Management der Infrastruktur vereinfachen.

Die verteilte Dokumentverarbeitung ist in Anwendungsfällen der Informationsextraktion im Kontext der *Enterprise Search* ein relevantes Problem und wird durch *MapReduce*-basierte Verfahren hinreichend gelöst. In Kap. 3.4 werden Verfahren zur parallelen, verteilten Verarbeitung detaillierter erläutert und insbesondere Gemeinsamkeiten und Unterschiede des hier entwickelten Ansatzes zur Parallelisierung mit *MapReduce*-basierten Verfahren erörtert.

Verarbeitung von Einzeldokumenten. Oft ist in der Informationsextraktion im Unternehmensumfeld die effiziente Verarbeitung eines einzelnen Dokuments, z. B. einer Rechnung oder einer Foren- bzw. Kundenanfrage, relevant. Die Verbesserung der Laufzeit für ein Einzeldokument ist jedoch auch außerhalb des Unternehmenskontext von Bedeutung, wie öffentlich zugängliche Extraktionsdienste, wie OpenCalais [DA09] zeigen. Die Beschreibung der Abhängigkeiten zwischen Daten in Extraktionsplänen [KLR+08, SDNR07, BFBS10] ermöglicht neben einer kostenoptimierten Ausführungsreihenfolge eine weitgehende Parallelisierung. Es können daher potenziell nicht nur voneiander unabhängige Teilaufgaben parallel, sondern auch voneinander unabhängige Daten nebenläufig verarbeitet werden.

Die Verarbeitung von Dokumenten durch Extraktionsdienste wie z. B. von Webseiten durch OpenCalais [DA09] nimmt in der Regel wenige Sekunden in Anspruch. Extraktionsdienste sind zumeist auf ressourcenreichen Systemen installiert, die für Lastspitzen optimiert sind. Wenn ungenutzte Ressourcen zur Verfügung stehen, ist es durchaus wünschenswert, die Verar-

beitungszeiten von wenigen Sekunden auf Sekundenbruchteile zu reduzieren. Weiterhin ist eine Intra-Dokument-Parallelisierung von Extraktionsprogrammen zur Verbesserung der Laufzeiten gerade für sehr große Dokumente (z. B. PDF-Dokumente), die häufig im Bereich der *Enterprise Search (ES)* anzutreffen sind, vielversprechend.

Ein solcher Ansatz wird in der Literatur jedoch bisher nicht beschrieben und erfordert eine neuartige Datenstrom-artige Dokumentverarbeitung, bei der nicht die Abarbeitung von Aufgaben pro Dokument, sondern eine Synchronisation der IE-Module auf Basis von Abhängigkeiten einzelner Zwischenergebnisse im Mittelpunkt steht.

2.2.3 Zusammenfassung

In Kap. 2.2 wurde ein Überblick zu Verfahren für die Steigerung der Effizienz von IE-Systemen gegeben. Die Mehrzahl der vorgestellten Verfahren sind im Bereich der Informationsextraktion für die *Enterprise Search* relevant. Als offenes Forschungsproblem wurde die Intra-Dokument-Parallelisierung von IE-Programmen, die durch komplexe Extraktionspläne beschrieben werden, identifiziert. Dabei steht nicht nur die Parallelisierung von unabhängigen Aufgaben, sondern die Parallelisierung auf Basis der Unabhängigkeit von Daten im Mittelpunkt. Die in dieser Arbeit entwickelten Konzepte zur Parallelisierung und Synchronisation werden in Kap. 3 vorgestellt.

2.3 Effektive Extraktion und Identifikation von Entitäten

Eine hohe Extraktionsqualität ist stark an die effektive Ausnutzung von vorhandenem Domänenwissen gebunden. In Abb. 2.3 ist eine Klassifikation von Domänenwissen nach Umgebungsmerkmalen und Entitätenmerkmalen mit Beispielen dargestellt.

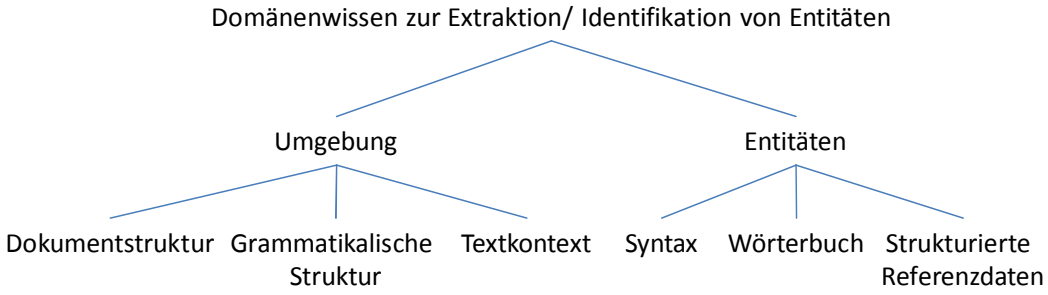


Abb. 2.3: Klassifikation von Domänenwissen zur Extraktion und Identifikation von Entitäten

Umgebungsmerkmale sind von den zu analysierenden Dokumententypen abhängig. Sie können z. B. in Form von Regeln manuell beschrieben oder durch ein maschinelles Lernverfahren als relevant bewertet werden. In Anwendungsfällen der Informationsextraktion im Unternehmensumfeld ist vor allem das Domänenwissen über Entitäten charakteristisch (siehe Kap. 1.1.2). Entitäten können markanten Bildungsvorschriften folgen, auf Basis eines Wörterbuchvergleichs lokalisiert oder mit Hilfe von komplex strukturierten Referenzdaten (mit Attributen, Entitäten und Beziehungen) in Texten identifiziert werden.

Bildungsvorschriften für Entitäten, die einer spezifischen Syntax folgen, können oft durch Reguläre Ausdrücke dargestellt werden. Die Entwicklung von geeigneten Regulären Ausdrücken ist jedoch nicht trivial, insbesondere wenn eine extrem große Menge von Beispielentitäten gegeben ist und ein Nutzer manuell Muster identifizieren muss, die zur Extraktion von unbekannten Entitäten der selben Klasse geeignet sind. In Kap. 2.3.1 werden Maschinelle Lernverfahren im Bereich der Informationsextraktion hinsichtlich ihrer Eignung für die Inferenz von

Bildungsvorschriften aus positiven Beispielinstanzen bewertet und offene Forschungsfragen abgeleitet.

Komplex strukturierte Referenzdaten stehen in vielen Anwendungsfällen im Kontext der *Enterprise Search* als Domänenwissen zur Verfügung. Der Aufbau von Referenzdaten kann dabei von Tabellen-artigen Strukturen bis hin zu komplexen Graphstrukturen variieren. Weiterhin enthalten Referenzdaten zumeist nicht nur einzelne Wörter als Bezeichner für Entitäten. Bezeichner oder andere Attribute von Entitäten werden häufig mit spezifischem Fachvokabular beschrieben.

Ein Überblick zu Ansätzen, die Strukturmerkmale von Referenzdatensätzen zur Identifikation von Entitäten in Texten ausnutzen, wird in Kap. 2.3.2 gegeben. Das Ziel ist es offene Forschungsfragen, die sich bei der effektiven Ausnutzung von Graph-strukturierten Referenzdaten ergeben, herauszustellen.

2.3.1 Mustererkennung zur Extraktion von Entitäten

In Abb. 2.4 ist eine Einordnung von Maschinellen Lernverfahren im Bereich der Informationsextraktion gemäß der Dimensionen: „vorhandenene Trainingsdaten“ und „erlernte Merkmale/ Merkmalskombinationen“ dargestellt. Anzumerken ist weiterhin, dass die Verfahren die Merkmale natürlich nicht automatisch erlernen, sondern vorkonfigurierte Merkmale oder Merkmalskombination bewerten und / oder selektieren. Die Verfahren unterscheiden sich hinsichtlich ihrer Unterstützung für verschiedene Merkmalstypen bzw. deren Kombination. Ansätze, die weiter unten in Abb. 2.4 angeordnet sind, sind auch für die darüber liegenden Arten von Trainingsdaten geeignet.

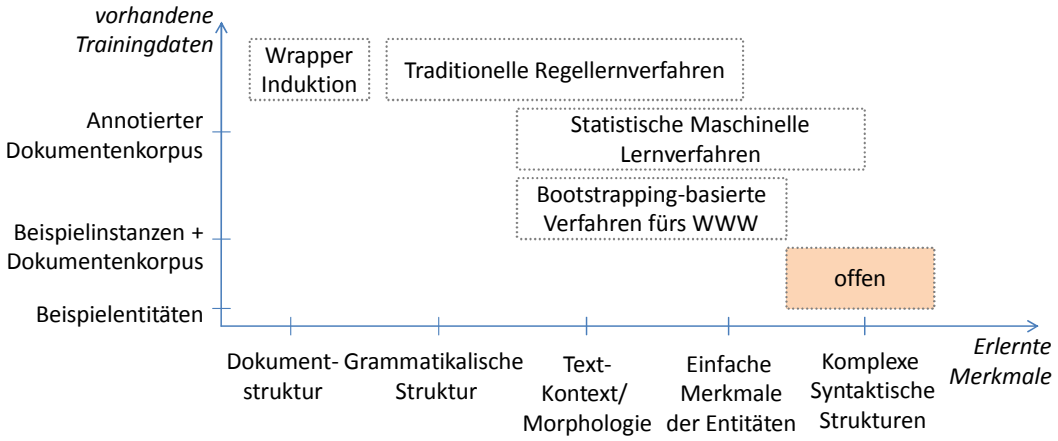


Abb. 2.4: Einordnung von Ansätzen zur Mustererkennung für IE-Systeme

Merkmalstypen im Kontext von Entitäten können in Dokumentstruktur, grammatikalische Struktur von Sätzen/ morphologische Merkmale und einfache Kontextmerkmale (z. B. häufige Wörter, Groß- oder Kleinschreibung) eingeteilt werden. Merkmalstypen für Entitäten können ebenfalls einfacher Natur sein, also häufige Wörter oder Wortklassen (z. B. Groß- oder Kleinschreibung) bzw. komplexe syntaktische Strukturen repräsentieren (z. B. die Kombination: Groß- oder Kleinschreibung, häufige Wörter und Zeichen-N-Gramme).

Verfügbare Trainingsdaten reichen von einem annotierten Korpus, über gegebene Beispielentitäten mit nicht annotierten Korpus, zu Anwendungsfällen, in denen lediglich Beipielentitäten und keine oder wenige Dokumente gegeben sind.

Im Folgenden wird ein Überblick über Verfahren zur Mustererkennung in der Informationsextraktion gegeben, Grundlagen zur Inferenz von Regulären Grammatiken aus positiven Bei-

spielinstanzen erläutert und ein zusammenfassender Überblick zu offenen Forschungsfragen gegeben (farbig markierter Bereich in Abb. 2.4).

2.3.1.1 Mustererkennung in der Informationsextraktion

Verfahren zur Wrapper-Induktion, Regellernverfahren und Statistische Maschinelle Lernverfahren wurden überblicksweise bereits in Kap. 2.1 eingeführt. Im Folgenden wird ein Überblick über Regellernverfahren und Statistische Maschinelle Lernverfahren und eine Bewertung hinsichtlich ihrer Eignung zur Inferenz von Bildungsvorschriften anhand von Beispielentitäten im Unternehmensumfeld gegeben. Auf die Wrapper-Induktion wird als Anwendungsfall der grammatikalischen Inferenz in dem darauf folgenden Unterkapitel eingegangen.

Regellernverfahren. Traditionelle Regellernsysteme wie z. B. AutoSlog [Ril93], PALKA [KM95], LIEP [Huf95], DIPRE [Bri98], TIMES [CBG99], RAPIER [CM99], ExDISCO [Yan03] oder SRV [Fre00] sind für das (semi-)automatische Training mit annotierten Dokumentenkorpora ausgelegt. Sie erlernen das Auftreten von Entitäten in der grammatikalischen Struktur von Sätzen bzw. einfachere Merkmale im Kontext von Entitäten, wie z. B. charakteristische Wörter, morphologische Charakteristiken oder andere Wortmerkmale. Ein Beispiel für eine durch SRV [Fre00] erlernte Regel, die Vortragende z. B. in Konferenzankündigungen auf Basis von Kontextmerkmalen extrahiert, ist in Bsp. 2.3 dargestellt.

```
1  speaker:-                        // Fragment F ist Entität "Vortragender", wenn:
2  length(=, 2))                    // F enthält 2 Wörter.
3  every(capitalizedp, true)        // Alle Wörter in F sind groß geschrieben.
4  every(quadruple_char_p,false)    // Alle Wörter in F haben nicht die Länge von 4.
5  some(?A, [], word, *unknown*)    // Es tritt ein Wort A und
6  some(?B, [], word, *unknown*)    // ein Wort B auf.
7  some(?A, [next_token], doubletonp, false) // Auf A folgt kein Wort mit 2 Zeichen.
8  some(?B, [prev_token], word, ":") // Vor B kann sich ein Doppelpunkt befinden.
9  some(?B, [prev_token prev_token], word, "who") // 2 Wörter vor B kann sich
10                                                // das Wort "who" befinden.
```

Bsp. 2.3: Mit SRV erlernte Regel

Regellernsysteme sind im Allgemeinen auf das Erlernen von Umgebungsmerkmalen auf Basis annotierter Dokumente spezialisiert (siehe [Ril93, KM95, Huf95, CBG99, Yan03] für einen Überblick), aber grundsätzlich nicht darauf beschränkt. Oft ist eine charakteristische Merkmalsmenge für die Position vor einem Entitätstyp (so genannte *Pre-Filler*), für einen Typ von Entität (so genannte *Filler*) und die Position nach einem Entitätstyp (so genannte *Post-Filler*) gesucht. Einige Systeme unterstützen mehrere Typen von Entitäten in einer Regel (so genannte *Multi-Slot-Systeme*). Wenn lediglich Beipielentitäten gegeben sind, könnten also potenziell einfach Merkmale für *Filler*-Positionen erlernt werden.

Bisherige Regellernsysteme unterstützen jedoch nur die Selektion von charakteristischen Merkmalsmengen auf Wortebene (häufige Wörter, Wortklassen, morphologische Merkmale etc.) und sind daher zur Beschreibung von Mustern in Entitäten der Art {d5100t, m8200n, m8000z} zu grobgranular. Zum Zweiten ist eine Mengendarstellung von Merkmalen zur Beschreibung der sequentiellen, syntaktischen Abfolge von Merkmalen in vielen Fällen ungeeignet. Zum Dritten basieren die Algorithmen zur Mustererkennung in Regellernsystemen häufig auf der Verteilung von positiven und negativen Beispielen für ein Merkmal im Korpus. Negativbeispiele sind im Falle einer Inferenz von positiven Beipielentitäten eben nicht gegeben.

Traditionelle Regellernsysteme sind aufgrund der drei genannten Einschränkungen für die Anwendung zur Rekonstruktion von Bildungsvorschriften ungeeignet. Eine detaillierte Abgrenzung zu Algorithmen zur Mustererkennung, die im Bereich der Regellernsysteme verwendet werden, wird in Kap. 4.3 vorgenommen.

Wie bereits in Kap. 2.1 erläutert, ermöglicht ein *Bootstrapping* die Verwendung von Maschinellen Lernansätzen, wenn lediglich Beispielentitäten und ein nicht annotierter Dokumentkor-

pus gegeben sind. *Bootstrapping*-basierte Regellernsysteme sind z. B. in DIPRE [Bri98], Snowball [AG00] oder AliBaba [PHL05, HSL07, NTL10] zu finden.

DIPRE [Bri98] und Snowball [AG00] unterscheiden sich von o. g. Regellernsystemen dahingehend, dass sie nicht auf die Extraktion aus Fließtexten, sondern insbesondere auf die Extraktion aus dem WWW spezialisiert sind. Es werden Regeln der Art „⟨Organisation⟩'s headquarters in ⟨Location⟩" erlernt. Morphologische Merkmale oder grammatikalische Satzstrukturen werden von Systemen zur Extraktion aus dem WWW meist nicht verwendet, da sie eher auf irregulär strukturierten Texten operieren. Die schiere Datenmenge im WWW erlaubt jedoch, dass auf eine Abstraktion z. B. von Wörtern zu Wortklassen verzichtet werden kann, da signifikante Wortphrasen häufig auftreten und zumeist nicht die Annotation von jedem einzelnen Dokument, sondern die Erstellung von „Extraktionsdatenbanken" im Vordergrund steht.

AliBaba [PHL05,HSL07,NTL10] erlernt Spezialisierungen von *Linguistischen Frames* – typischen grammatikalischen „Kernmustern" wie „⟨*Protein*⟩ Word* ⟨*Verb*⟩ Word* ⟨*Protein*⟩" –, um z. B. Protein-Protein-Interaktionen aus biomedizinischen Texten zu extrahieren. „⟨*Protein*⟩" und „⟨*Verb*⟩" sind aus Wörterbüchern bekannte Entitäten bzw. charakteristische Wörter. Anhand eines Dokumentenkorpus werden z. B. morphologische Merkmale, charakteristische Wörter oder die Wortanzahl für Platzhalter („Word*") erlernt.

In [LKR+08] wird ein Ansatz zur Optimierung von Regulären Ausdrücken hinsichtlich ihrer Genauigkeit und Vollständigkeit vorgestellt, der als Eingabe einen Regulären Ausdruck und einen annotierten Traininkorpus benötigt. Basierend auf den annotierten Trainindokumenten werden Reguläre Ausdrücke hinsichtlich ihrer Genauigkeit und Vollständigkeit in der Extraktion durch Umformung von Ausdrücken und Erzeugung von Negativlisten optimiert. Die Ergebnisse der experimentellen Evaluierung von [LKR+08] zeigen, dass optimierte Reguläre Ausdrücke dem Maschinellen Lernverfahren *Conditional Random Fields* [CMT05] für bestimmte Typen von Entitäten in Genauigkeit und Vollständigkeit bei Verwendung derselben Merkmalstypen überlegen sind.

Ein Ansatz, der Reguläre Ausdrücke auf Basis von Beispielentitäten erlernt, ist eine bisher in der Literatur nicht diskutierte Problemstellung, und ist angesichts der Ergebnisse von [LKR+08] erfolgversprechend.

Statistische Maschinelle Lernverfahren. Ansätze, die auf Statistischen Maschinellen Lernverfahren basieren, werden zumeist in Anwendungsfällen im Kontext annotierter Trainingskorpora angewendet, können aber auch ohne weiteres in *Bootstrapping*-Szenarien eingesetzt werden [NS07, Sar08]. Der große Vorteil gegenüber bisherigen Regellernverfahren liegt in der Verfügbarkeit von Ansätzen, die eine Kombination von abhängigen Merkmalen unterschiedlicher Granularität, wie z. B. von Wortklassen, Wörtern und Zeichen-N-Grammen unterstützen [NS07]. Statistische Maschinelle Lernverfahren sind daher, vom Standpunkt der unterstützten Merkmalstypen aus, zum Erlernen der Syntax von Entitäten besser geeignet, als bisher diskutierte Verfahren.

Der in der Vergangenheit populäre Ansatz auf Basis von *Hidden Markov Models* ist aufgrund seiner generativen Natur auf die Verwendung genau einer Merkmalsgranularität begrenzt (z. B. entweder Zeichen-N-Gramme oder Wortmerkmale) [LMP01]. Eine gemeinsame Verwendung von verschiedenen zum Teil abhängigen Merkmalstypen verschiedener Abstraktionsebenen, wie dies für die Inferenz von Bildungsvorschriften notwendig ist, wird nicht ohne weiteres unterstützt.

Conditional Random Fields und *Maximum Entropy Markov Models* sind diskriminative Modelle und gestatten die Kombination von untereinander abhängigen Merkmalstypen verschiedener Granularität in einem Modell. Ein Training von Statistischen Maschinellen Lernverfahren anhand von Beipielentitäten ohne Trainingskorpus ist aufgrund fehlender negativer Beispiele bei der Extraktion jedoch wenig effektiv und daher nicht sinnvoll. Wesentlicher ist jedoch das Problem, dass bei der Verwendung von statistischen Modellen die Fehlersuche bei einer automatischen Inferenz von Bildungsvorschriften und eine Feinjustierung im Falle einer semi-

automatischen Suche nach Bildungsvorschriften nicht trivial ist und einen enormen manuellen Aufwand verursacht.

2.3.1.2 Inferenz von Regulären Grammatiken

Das Erlernen von Regulären Ausdrücken aus einem gegebenen Satz von Beispielinstanzen hat als Ziel, die Bildungsvorschrift aus einem Satz von Beispielen zu rekonstruieren, die ursprünglich dazu genutzt wurde, um diese Instanzen zu generieren. Reguläre Ausdrücke beschreiben eine Reguläre Sprache und können alternativ durch Reguläre Grammatiken beschrieben werden. Die Problemstellung ist somit der Problemklasse der Inferenz von Regulären Grammatiken zuzurechnen.

Die Inferenz von Regulären Grammatiken wurde durch eine Formalisierung der menschlichen Sprache motiviert [FB86]. Für die Informationsextraktion aus Fließtexten haben sich jedoch Statistische Maschinelle Lernverfahren, die zumeist auf stochastischen Grammatiken, wie *Hidden Markov Models* basieren, durchgesetzt. Ein Reihe von weiteren Anwendungsmöglichkeiten für die Inferenz von Regulären Grammatiken wurde erforscht. Hierzu zählen z. B. das Erlernen von XML-Schema von gegebenen XML-Dateien (z. B. [HNW06, Fer09, GGR+00, BNSV10, BNST06]) oder das Erlernen von domänenspezifischen Scriptsprachen [Jav05]. Auch die Induktion von Web-Wrappern kann dieser Problemklasse zugeordnet werden. Das Ziel einer grammatikalischen Inferenz, die Basis der in dieser Arbeit untersuchten Lösung ist, wird nach [PH00] wie folgt definiert:

Definition 2.3.1 (*Grammatikalische Inferenz*)

Gegeben sei eine Menge von positiven Beispielinstanzen $\mathcal{I}^+$ und negative Beispielinstanzen $\mathcal{I}^-$ einer Sprache $\mathcal{L}(\mathcal{G})$. Das Ziel der grammatikalischen Inferenz ist es, eine Grammatik $\mathcal{G}^{\equiv}$ zu erlernen, die äquivalent zu $\mathcal{G}$ ist.

Bei den in dieser Arbeit diskutierten Problemstellungen sind in der Regel keine aussagekräftigen Negativbeispiele $\mathcal{I}^-$ gegeben. Die weiteren Ausführungen beschränken sich daher auf die grammatikalische Inferenz von positiven Beispielinstanzen.

Es ist anzumerken, dass das Erlernen einer Sprache mittels positiven Beispielinstanzen gemäß [Gol67] von einem theoretischen Standpunkt aus nicht möglich ist. Im Gebiet der *Theoretischen Informatik* werden fundamentale Problemstellungen wie Sprachklassen, für die ein exaktes Erlernen einer Grammatik von positiven Beispielen möglich ist, und entsprechende Annahmen untersucht (siehe z. B. [Ang82]). Weiterhin werden Algorithmen und Optimierungen zu derartigen Problemstellungen diskutiert (siehe z. B. [Pit89, Bra94, KMU95, PH00, FB86, NR04]). Dabei spielt jedoch eine Abstraktion von den Beispielinstanzen, wie sie hier notwendig ist, keine bzw. eine untergeordnete Rolle.

In verschiedenen anwendungsorientierten Forschungsbereichen wurden Heuristiken entwickelt, um geeignete Näherungsgrammatiken $G^{\approx}$ zu erlernen, die von den konkreten Beispielinstanzen abstrahieren und dennoch präzise Resultate erzeugen. Die Algorithmen, die verwendet werden, um Näherungsgrammatiken zu bestimmen, unterscheiden sich für verschiedene Anwendungsgebiete maßgeblich.

Bei der Inferenz von XML-Schema sind insbesondere Kardinalitäten von XML-Elementen und deren Wiederholungen im XML-Baum relevant [GGR+00, HNW06, BGNV08, Fer09]. Diese Problemstellung kann zur Inferenz von Regulären Ausdrücken abstrahiert werden, unterscheidet sich jedoch stark von der Inferenz von Bildungsvorschriften für Entitäten im Sinne dieser Arbeit. Wir analysieren die algorithmische Vorgehensweise in Kap. 4.3 im Detail.

Formal kann die Wrapper-Induktion [TAC06, CKGS06, BGG09] als Problemstellung der grammatikalischen Inferenz von positiven Beispieldaten klassifiziert werden [Sak97]. Das Ziel ist es, die Muster in den Dokumentstrukturen als Grammatik abzubilden, wobei fixe und variable Bestandteile, anders als bei der in dieser Arbeit verfolgten Problemstellung, anhand von anno-

tierten Beispieldokumenten gegeben sind. Es gibt jedoch Parallelen, die ebenfalls in Kap. 4.3 erläutert werden.

Aus Anwendungsperspektive gesehen wurden die in dieser Arbeit verfolgten Problemstellungen im Bereich der grammatikalischen Inferenz bisher nicht untersucht. In Kap. 4.3 werden Heuristiken aus dem Forschungsgebiet der grammatikalischen Inferenz detaillierter analysiert und abgegrenzt.

2.3.1.3 Zusammenfassung

In Kap. 2.3.1 wurde ein Überblick der Ansätze zur Mustererkennung im Bereich der Informationsextraktion gegeben. Bisherige Ansätze zur Inferenz von Extraktionsregeln unterstützen lediglich die Identifikation von Mustern in Dokumenten oder Fließtexten auf Basis von (annotierten) Trainingsdokumenten und nicht die Identifikation von Bildungsvorschriften aus einem gegebenen Satz von Beispielentitäten. Insbesondere wurde die Problemstellung der Bewertung und Selektion von abhängigen Merkmalstypen zur Ableitung von Bildungsvorschriften in der Literatur bisher nicht untersucht. Statistische Maschinelle Lernverfahren unterstützen derartige Merkmalskombinationen, erlernen jedoch ebenfalls Muster anhand von Trainingsdokumenten und erschweren eine manuelle Fehlersuche und Feinjustierung der Extraktionsmechanismen.

Weiterhin wurde die Inferenz von Regulären Grammatiken eingeführt, die als Zugang für die Ableitung von Bildungsvorschriften in dieser Arbeit herangezogen wird. Heuristiken, die bisher in anwendungsorientierten Forschungsgebieten im Kontext der grammatikalischen Inferenz vorgeschlagen wurden, sind für die in dieser Arbeit verfolgte Problemstellung nicht geeignet.

Ein Verfahren, welches die Inferenz von Regulären Ausdrücken auf Basis von positiven Beispielentitäten ermöglicht, wird in Kap. 4 vorgestellt. In Kap. 4.3 werden ausgewählte verwandte Ansätze, auch aus anderen Bereichen, wie z. B. der Datenbereinigung aufgegriffen und hinsichtlich der verwendeten Mustererkennungsverfahren detaillierter analysiert und abgegrenzt.

2.3.2 Identifikation und Disambiguierung von Entitäten mit Referenzdaten

Zur Entwicklung von IE-Systemen für die *Enterprise Search* ist ein IE-Modul wünschenswert, dass strukturierte Referenzdaten zur Identifikation von Entitäten auf effektive Art und Weise ausnutzt. Formal können die meisten Referenzdatensätze, unabhängig vom konkreten Datenformat, wie folgt als Graph abstrahiert werden:

Definition 2.3.2 (*Graph-strukturierter Referenzdatensatz*)

Ein Graph-strukturierter Referenzdatensatz sei ein gerichteter azyklischer Graph $\mathcal{G}_{\mathrm{R}} = (N, E, L_N, L_E)$. *$N$ sei eine Menge von Knoten, $E \subseteq N \times N$ eine Menge von gerichteten Kanten, L_N sowie L_E eine Menge von Knoten- und Kantenbezeichnungen. Quellknoten $N_a \subset N$ sind eine Menge von Attributtyp-Attributwert-Paaren eines Referenzdatensatzes. $N_e \subset N$, $N_e \cap N_a = \emptyset$ sei eine Menge von Entitäten eines Referenzdatensatzes, die die Semantik der zugeordneten Attributwerte repräsentiert.*

Anzumerken ist, dass die Annahme von gerichteten Kanten nicht für alle Referenzdatensätze zutreffend erscheint. Wir nehmen jedoch an, dass je nach Anwendungsfall eine Richtung von Kanten konfiguriert bzw. eine entsprechende Sicht auf die Daten erzeugt werden kann (z. B. „Postadresse“ $\mapsto$ „Geschäftspartner“ $\mapsto$ „Bestellung“ für das Beispiel der Rechnungsverifikation).

Ein einfacher Wörterbuch-basierter Ansatz auf Basis eines exakten Vergleichs von Text und Attributwerten ist in vielen Anwendungsfällen wenig effektiv, insbesondere wenn hohe Anfor-

derungen an die Genauigkeit und Vollständigkeit in der Identifikation von Entitäten bestehen. Zum Beispiel können Entitäten durch einen exakten Abgleich mit Attributwerten nicht identifiziert werden, wenn Attributwerte in Dokumenten in abweichender Form vorkommen, ein Attributwerttreffer die Entität nicht ausreichend beschreibt (z. B. ein Attributwert für die Farbe eines Produkts) oder Entitäten nur implizit durch in Beziehung stehende Entitäten referenziert werden (z. B. eine Bestellung an Hand einer Menge von Produkten). Zum anderen kann die Genauigkeit der Identifikationsergebnisse durch Mehrdeutigkeiten in Mitleidenschaft gezogen werden.

Im Folgenden wird ein Überblick zum Stand der Technik zur Identifikation und Disambiguierung von Entitäten gegeben und ungelöste Forschungsprobleme anhand des in Kap. 1.2.2 betrachteten Anwendungsfalls detailliert herausgestellt.

2.3.2.1 Ansätze zur Identifikation von Entitäten in Text

Ansätze in dem Bereich der Identifikation von Entitäten lassen nach der Struktur der Referenzdaten (Wörterbuch, tabellenartige oder Graph-strukturierte Referenzdaten) und der Art von Eingabe („Anfrage"), wie in Abb. 2.5 dargestellt, klassifizieren.

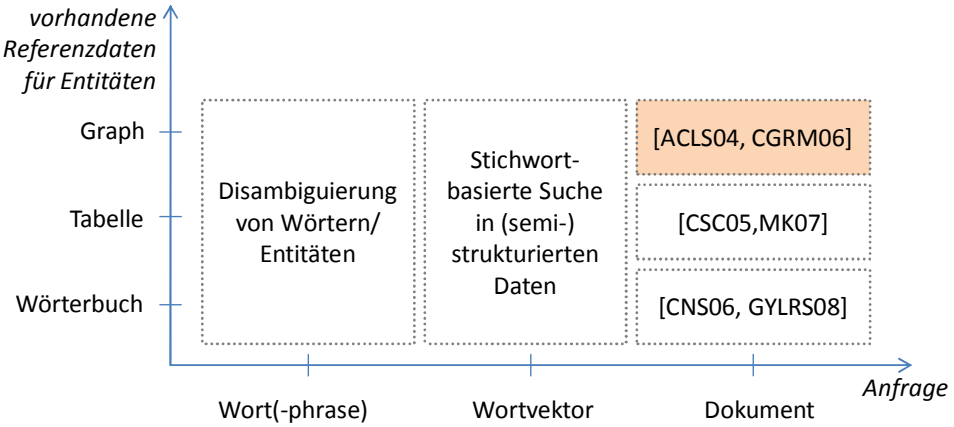


Abb. 2.5: Einordnung von Ansätzen zur Identifikation von Entitäten mit Referenzdaten

Es können die folgenden Anfragetypen unterschieden werden: 1.) Wort(-phrase) mit Kontext, wobei die Identifikation der Bedeutung eines mehrdeutigen Wortes (bzw. Wortphrase) im Mittelpunkt steht; 2.) Stichwortanfrage, die aus wenigen besonders relevanten Wörtern besteht und 3.) Anfragedokumente. Ziel ist es in jedem Falle eine nach Relevanz geordnete Resultatliste von Entitäten (mit eindeutiger Semantik) zu erhalten:

Anfragetyp: Wort(-phrase) mit Kontext. Die Eingabe von Disambiguierungsverfahren ist in der Regel ein Wort oder eine Wortphrase, die beim Abgleich mit einem Wörterbuch nicht eindeutig einer Bedeutung zugewiesen werden konnte. Beispiele für Dismabiguierungsprobleme sind: ob das Englische Wort „drug" im Sinne von „Betäubungsmittel" oder „Medikament" oder ob die Entität „Athens" in „Griechenland" oder „Georgia, USA" gemeint ist. Grundlage der Disambiguierung ist im Allgemeinen der Textkontext des Wortes/ der Wortphrase, also ein Fenster von Wörtern links und rechts der mehrdeutigen Entität.

Ist lediglich das Wörterbuch und ein annotierter Trainingskorpus verfügbar, können Lernverfahren, die *Entscheidungsbäume* [Ped01] ableiten, oder *Naive Bayes* [EMR00] eingesetzt werden, um charakteristische Wörter im Kontext zur Disambiguierung zu erlernen (siehe [Nav09] für einem Überblick). Sind Tabellen-strukturierte Referenzdaten gegeben, wie z. B. ein Glossar

mit Sinnumschreibungen, kann die Bedeutung mit der größten Wortschnittmenge von Kontext und Sinnumschreibung gewählt werden [Les86, IV98].

Sind Graph-strukturierte Referenzdaten gegeben, können verschiedene Abstandmaße zwischen Sinn und Kontextwörtern zur Disambiguierung herangezogen werden [Voo93, SP95, RS95, AR96, MTT+98, SOT03, NV03, HCK+04, NL07, ALS09, NL10]. Zum Beispiel kann der Wortsinn mit dem kürzesten Abstand zu den Kontextwörtern im Referenzgraphen als relevantester Sinn selektiert werden (genauer die Summe der Länge der kürzesten Pfade). In der Literatur werden verschiedene Abstands-/ Relevanzmaße zur Gewichtung von Kanten diskutiert (z. B. auch eine Adaption von *PageRank* [NL10]).

Es handelt sich jedoch in jedem Fall um ein Ranking von Wortsinnen für ein gegebenes Wort hinsichtlich des Kontexts, nicht um ein Ranking von Entitäten für einen Text. Syntaktische Heterogenitäten zwischen Text und Referenzdaten oder die implizite Identifikation von Entitäten spielen in dem Forschungsgebiet der Sinndisambiguierung keine Rolle. In Kap. 5.3 werden ausgewählte Verfahren aus dem Bereich der Sinndisambiguierung aufgegriffen und von einem algorithmischen Standpunkt zu dem in dieser Arbeit vorgestellten Ansatz abgegrenzt.

Anfragetyp: Wortvektor. Wenn Eingabetexte sehr kurz sind (z. B. kürzer als 8 Wörter), kann eine Wortvektor-basierte Darstellungsform eines Textes sinnvoll sein. Wenn es sich bei den Referenzdaten um ein einfaches Wörterbuch handelt, oder lediglich eine Tabelle durchsucht werden soll, kann eine Zuordnung häufig einfach mit Bordmitteln, die heutzutage in Datenbanken zur Verfügung stehen (wie z. B. einem Volltextindex für Tabellenspalten), hergestellt werden.

Das Forschungsgebiet der Stichwort-basierten Suche in (semi-)strukturierten Daten (relationale Daten, XML, Ontologien) hat zum Ziel Entitäten oder komplexere Objekte, die dynamisch aus in Beziehung stehenden Entitäten (über mehrere Tabellen hinweg) zusammengesetzt werden, hinsichtlich einer gegebenen Menge von Stichwörtern zu bewerten (siehe [YQC10] für einen Überblick).

Viele Verfahren im Bereich der Stichwort-basierten Suche in relationalen Daten präferieren kompakte Antwortgraphen [HP02, LYMC06, LLZ07, ZWX+07, LOF+08], da davon ausgegangen wird, dass kompaktere Informationen den Informationsbedarf eines Nutzers besser befriedigen. Derartige Bewertungsmetriken sind im Kontext der Schlüsselwort-basierten Suche durchaus sinnvoll, führen jedoch zu Qualitätsverlusten für Anfragen, die länger als acht Wörter sind, wie die Ergebnisse von [ZWX+07] nahe legen.

Die Nutzung von PageRank (bzw. dessen Übertragung auf Objektgraphen: „*ObjectRank*“ [BHP04]) wird unter anderem von *BANKS* [BHN+02] und *BANKS II* [KPC+05] vorgeschlagen. Wenn jedoch Beziehungen in den Referenzdaten nicht der Semantik von Popularität entsprechen, sind derartige Bewertungsmetriken ungeeignet. Aktuelle Ansätze orientieren sich an der Stichwort-basierten Suche von Dokumenten und verwenden zur Bewertung von komplexen Objekten häufig aggregierte *TF-IDF*-Wortgewichte der verschiedenen Attributübereinstimmungen [LYMC06, LLZ07, JWR00].

Ansätze wie z. B. [GSBS03, XP08, FHRW09], die XML-Teilbäume für eine gegebene Stichwortanfrage bewerten, adaptieren grundsätzlich die vorgestellten Verfahren zur Stichwort-basierten Suche in relationalen Daten und adressieren zusätzliche Spezifika von XML, wie z. B. *Hyperlinks* zwischen XML-Fragmenten. Suchmaschinen für das *Sematic Web* [RSA04] wie Swoogle [DFJ+04], [QHC06] oder Sig.Ma [TUD+09] adaptieren zumeist oben genannte Ansätze. Am populärsten sind an *PageRank* angelehnte Rankingverfahren [DFJ+04, RSA04]. Es werden jedoch auch *TF-IDF*-basierte Bewertungen für aggregierte Ontologiegraphen diskutiert [QHC06].

Der Ansatz von [ZWX+07] generiert und gewichtetet potenzielle *SPARQL*-Anfragen für eine gegebene Stichwortanfrage. Es werden auch approximative Vergleichsmethoden herangezogen (z. B. Levenshtein Distanz oder Synonymlisten). Ausgangspunkt der Bewertung sind einfache Konfidenzgewichte: je nach angewendeter Methode, die zu einer Übereinstimmung führte,

wird ein vordefinierter Konfidenzwert zugewiesen (z. B. 0,5 für Treffer mit Synonymlisten). Zur Bewertung von potenziellen *SPARQL*-Anfragen (entsprechen hier Teilgraphen der Ontologie) wird einfach der Durchschnitt der Konfidenzwerte und die Anzahl der Übereinstimmungen als Relevanzmaß herangezogen.

Extrem lange Anfragen (Dokumenttexte) und Referenzdaten, in denen Wörter einem extrem hohen Grad an Mehrdeutigkeit unterliegen, werden im Bereich der Stichwort-basierten Suche in (semi-)strukturierten Daten nicht diskutiert. Ebenso wird das Problem der Identifikation von Entitäten, deren Beschreibung aus mehreren Wörtern bestehen, die zusammen oder verteilt in einem Anfragetext auftreten können, nicht explizit untersucht. Die Identifikation von in der Anfrage implizit oder unscharf referenzierten Entitäten spielt im Gebiet der Stichwort-basierten Suche in strukturierten Daten eine untergeordnete Rolle. Weiterhin repräsentieren die meisten der oben genannten Ansätze Anfragen und Attribute von Entitäten als Wortvektor, vernachlässigen also die Reihenfolge von Wörtern in Anfragen und Attributen und sind daher zum *Ranken* von Entitäten hinsichtlich eines gegebenen Dokuments wenig effektiv.

Anfragetyp: Dokument. Der exakte und approximative Abgleich von Texten mit Wörterbüchern oder genereller der Vergleich von Zeichenketten ist sehr gut erforscht (siehe z. B. [MM02, EIV07] für einen Überblick). Im Kontext der Informationsextraktion stellt sich jedoch häufig das Problem, dass Wörter, die Bestandteile von Attributen einer Entität beschreiben, verteilt im Text auftreten (z. B. die Entität „*Regler für die Bremskraft von Anhängern*“ in einem Text der Art „*Können Sie mir einen Bremskraftregler für meinen Anhänger zusenden?*“).

Der Ansatz von [CNS06] schlägt für die Bewertung von Entitäten eines Wörterbuchs ein gleitendes Textfenster vor. Entitäten und Textfenster werden als Wortvektor interpretiert und mit *TF-IDF* bewertet. [GYLRS08] verwendet biologische Ontologien zur Identifikation von Entitäten, nutzt jedoch nicht die Graphstruktur. Textsegmente (z. B. Absätze) in Dokumenten werden als Wortvektor dargestellt und mit Entitäten, deren Bezeichner eher Umschreibungen ähneln (z. B. „activation of JNKK activity“), abgeglichen. [CSC05] und [MK07] aggregieren Übereinstimmungen von Text und verschiedenen Attributen einer Entität zur Bewertung, adressieren also insbesondere Daten, welche in Tabellenform vorliegen.

EROCS [CGRM06] *rankt* Entitäten, die Wurzelknoten eines Baum-strukturierten Referenzdatensatzes sind, hinsichtlich eines Textsegments (z. B. Sätze) und ermöglicht die Identifikation von implizit im Text referenzierten Entitäten (z. B. einer Geschäftstransaktion auf Basis von Produkten). Die *TF-IDF*-Wortgewichte für alle Attribute aller Entitäten, die zu einer „Wurzelentität“ in Beziehung stehen, werden aggregiert, ohne Charakteristiken der Baumstruktur zu berücksichtigen. Das Ziel ist nicht Bewertung der Relevanz einer Entität für den Text, sondern vielmehr eine „optimale“ Segmentierung des Dokumentes durch das Zusammenfassen feinergranularer Textsegmente zu höherwertigen Segmenten bezüglich der „Wurzelentitäten“.

In [AHSS04] wird ein Ansatz für das *Ranken* von dominanten Orten, welche in einer Taxonomie organisiert sind, für einen Text vorgestellt. Einfache Konfidenzkennzahlen (z. B. 0,5 für mehrdeutige Treffer) werden entlang des Taxonomiegraphs propagiert. Das Vorkommen von Wörtern verschiedener Attribute einer Entität im Text und die Bewertung von approximativen im Text verstreuten Übereinstimmungen werden nicht diskutiert und ist im Kontext der Identifikation von Orten auch nicht notwendig.

Weiterhin werden in der Literatur eine Reihe von anwendungsspezifischen Lösungen diskutiert, wie die Identifikation und Disambiguierung von Personen in Konferenzankündigungen [HAMA06] oder in Emails [MCN06]. Bisherige Ansätze adressieren lediglich Teilprobleme, die sich aus den in Kap. 1.2 beschriebenen Anwendungsfällen ergeben. Wir analysieren die oben vorgestellten und anwendungsspezifische Ansätze im Detail in Kap. 5.3.

2.3.2.2 Effektive Ausnutzung von Graph-strukturierten Referenzdaten

Zur Illustration der offenen Forschungsprobleme zur Identifikation und Disambiguierung von Entitäten wird im Folgenden das Szenario der Zuordnung von Softwaremodulen zu Foreneinträgen des *SAP Community Network (SCN)* mit Hilfe der *SAP Terminologie (SAPTerm)* diskutiert (siehe Kap. 1.2.2).

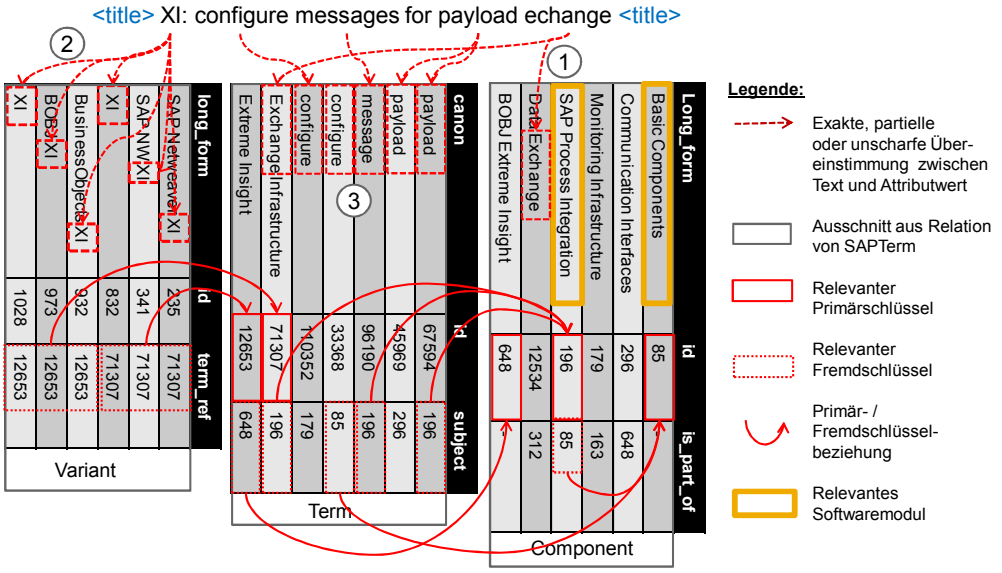


Abb. 2.6: Identifikation und Disambiguierung von Entitäten am Beispiel eines Titels eines SCN-Foreneintrags und einem Ausschnitt aus SAPTerm.

Im Beispiel in Abb. 1.5 ist ein Titel eines Foreneintrags dargestellt. Es wird ein Integrationsproblem zwischen dem Modul „*Process Integration*“ und einem nicht genannten Softwaremodul beschrieben, das durch das Modul „*Process Integration*“ bei Überlastsituationen benachrichtigt werden soll. Weiterhin ist in Abb. 1.5 ein Ausschnitt von SAPTerm dargestellt, mit dessen Hilfe relevante Softwaremodule identifiziert und Text-Referenzdaten-Übereinstimmungen disambiguiert werden sollen. Ein effektives Ranking von Softwaremodulen anhand des Referenzdatensatzes SAPTerm hinsichtlich des Titels sollte „*Process Integration*“ als relevantestes Softwaremodul und dessen übergeordnetes Modul „*Basic Components*“ in genau dieser Reihenfolge identifizieren.

Charakteristiken der hier betrachteten Anwendungsfälle, die durch einen Algorithmus für die Identifikation und Disambiguierung von Entitäten abgedeckt werden müssen, sind:

Fehlerhafte Textdaten. Textdaten enthalten häufig Fehler, z. B. Rechtschreibfehler (siehe Abb. 2.6(1): „*echange*“ statt “*Exchange*“) oder OCR-Fehler, die bei der Digitalisierung von Rechnungen auftreten können. Dieser Umstand muss beim Zuordnen von Texten zu Entitäten und bei der Aggregation von Treffergewichten innerhalb der Graphstruktur in Betracht gezogen werden. Erschwerend kommt hinzu, dass sich viele Attributwerte ähneln, so dass das Zulassen von Fehlern die Anzahl von möglichen Treffern enorm erhöht.

Identifikation anhand von Teilübereinstimmungen. Die Nutzung von Referenzdaten, die mehrdeutige Umschreibungen von Entitäten statt eindeutigen Bezeichnern enthalten, führt dazu, dass Entitäten häufig nicht durch einen Abgleich, der lediglich Abweichungen von einzelnen Zeichen erlaubt, identifiziert werden können. Als Konsequenz müssen auch Teilübereinstimmungen (Einzelwörter und Wortphrasen) zwischen Text und Referenzdaten zugelassen werden (siehe z. B. „*echange*“ und “*Exchange Infrastructure*“ in Abb. 2.6(1)), wodurch ein hoher Grad an Mehrdeutigkeit entsteht, da Umschreibungen häufig gleiche Wörter enthalten. Einzelne Übereinstimmungen müssen mit einem geeigneten Relevanzmaß bewertet werden.

Auch die Ordnung von Wörtern muss berücksichtigt werden, so dass korrekte Wortreihenfolgen bevorzugt, aber andere Reihenfolgen zugelassen werden (z. B. „*Process Integration*“ und „*Integration Process*“).

Indirekte Identifikation. Höherwertige Konzepte, wie z. B. das „*Basic Components*“-Softwaremodul in Abb. 2.6 oder Geschäftstransaktionen im Anwendungsfall aus Kap. 1.2.1, werden in Texten mitunter nicht direkt genannt. Sie werden vielmehr durch die Kombination mehrerer untergeordneter Entitäten in Texten referenziert, z. B. Softwaremodule durch technische Begriffe (siehe Abb. 2.6(3)) oder Abkürzungen (siehe Abb. 2.6(2)). Die Graphstruktur von Referenzdaten (Primär-/Fremdschlüsselbeziehungen in Abb. 2.6) kann zur Identifikation von „höherwertigen“ Entitäten verwendet werden. Häufig ist jedoch nicht die „Wurzelentität“ am relevantesten, da Treffer im Graphkontext einer „Wurzelentität“ z. B. auch durch Mehrdeutigkeiten entstehen.

Disambiguierung von direkten Übereinstimmungen. Das fehlertolerante Abbilden von Foreneinträgen auf SAPTerm über approximative und indirekte Zuordnungen erzeugt im hohen Maße mehrdeutige Treffer. Das Wort „*exchange*“ kommt z. B. in Attributen von mehr als 400 Entitäten vor. Weiterhin kann eine Abkürzung wie „*XI*“ im Beispiel häufig mehreren Bedeutungen wie „*Exchange Infrastructure*“ oder „*Extreme Insights*“ zugeordnet werden. Zur Verbesserung der Dokumentsuceh oder dem (semi-)automatischen Verarbeiten von Rechnungen ist eine Disambiguierung von solchen mehrdeutigen Einzeltreffern im Kontext des Gesamtdokuments notwendig.

2.3.2.3 Zusammenfassung

In Kap. 2.3.2 wurde ein Überblick zu Ansätzen im Bereich der Identifikation von Entitäten anhand von (Graph-)strukturierten Referenzdaten gegeben, die jedoch die Problemstellungen im Unternehmenskontext nur teilweise adressieren. In Kap. 5.3 ist eine detaillierte Analyse von verwandten Arbeiten zu finden, in der auch anwendungsspezifische Ansätze wie [BM05, HAMA06, Cuc07] aufgegriffen und abgegrenzt werden.

Weiterhin wurden in Kap. 2.3.2 detaillierte Problemstellungen beschrieben. Ein Ansatz, der alle Problemstellungen, der dieser Arbeit zugrunde liegenden Anwendungsfälle hinsichtlich der Identifikationseffektivität unterstützt, wurde in der Literatur bisher nicht beschrieben. Ein Verfahren, das diese Spezifika von Graph-strukturierten Referenzdatensätzen im Unternehmensumfeld adressiert, wird in Kap. 5.1 vorgestellt.

Bevor wir auf die in dieser Arbeit entwickelten Extraktions- und Identifikationsmethoden in Kap. 4 und Kap. 5 eingehen, beschreiben wir die Plattform, die Grundlage dieser Arbeit ist, in Kap. 3, und erläutern eine optimierte Datenverarbeitung in IE-Systemen, die aus anwendungsspezifischen und generischen IE-Modulen komponiert werden. Die hier entwickelten Extraktions- und Identifikationsmethoden werden als generische Module in der Plattform eingebettet.

3

Parallele Datenverarbeitung in der Informationsextraktion

Im Mittelpunkt dieses Kapitels steht ein Konzept zur Parallelisierung von Softwaremodulen, um eine effizienten Verarbeitung von Einzeldokumenten im Kontext von Extraktionsplan-basierten Systemen zur Informationsextraktion (IE) zu unterstützen. Die Alleinstellungsmerkmale des hier vorgestellten Ansatzes sind: 1.) eine Dokumentverarbeitung auf Basis von Datenströmen zwischen IE-Modulen, die die Idee des *Pipelining* [DG92], wie es aus herkömmlichen Datenbanksystemen bekannt ist, in den Bereich der Informationsextraktion adaptiert und 2.) effiziente Synchronisationsmechanismen, die Aufgaben- und Datenabhängigkeiten berücksichtigen, so dass Zwischenresultate so früh wie möglich nebenläufig verarbeitet werden können.

Wir diskutieren die hier entwickelten Konzepte zur Datenverarbeitung in Kap. 3.2. Die in der Praxis zu erzielenden Laufzeitverbesserungen im Vergleich zu einer herkömmlichen, *Workflow*-basierten Dokumentverarbeitung bei gleicher Extraktionslogik werden in Kap. 3.3 dargestellt. In Kap. 3.4 werden verwandte Ansätze zur Laufzeitoptimierung analysiert.

Bevor die Konzepte bezüglich der parallelen Datenverarbeitung vorgestellt werden, betrachten wir in Kap. 3.1 die zugrundeliegende Plattform zur effizienten Entwicklung von IE-Systemen. Sie ermöglicht eine graphische Komposition von generischen und anwendungsspezifischen IE-Modulen zu IE-Systemen. Einheitliche Schnittstellendefinition von IE-Modulen fördern eine flexible Integration und Wiederverwendung. Die Interoperabilität zwischen IE-Modulen wird durch ein generisches Graph-basiertes Meta-Datenmodell sicher gestellt.

3.1 Plattform zur Modellierung von Informationsextraktionssytemen

*RapidUIM („**Rapid** development of Systems for **U**nstructured **I**nformation **M**anagement“)* ist eine generische Plattform zur Entwicklung von anwendungsspezifischen IE-Systemen auf Basis von Extraktionsplänen. Der Aufbau dieses Kapitels, das die wichtigsten Konzepte von *RapidUIM* einführt, orientiert sich an den essentiellen Aspekten einer generischen, Extraktionsplan-basierten IE-Plattform nach [DNR+08]:

Nutzerschicht. Das Ziel eines IE-Systems ist es, strukturierte Informationen, die aus Dokumenten gewonnen wurden, für Nutzer zur Verfügung zu stellen. Geeignete Dienste, um auf extrahierte Daten zuzugreifen, sind: eine Stichwort-basierte Suche und strukturierte Anfragen, sowie die Möglichkeit zur Navigation in den Ergebnismengen. Derartige Dienste sind Anwendungen im Sinne dieser Arbeit und wurden in Kap. 1.2 beschrieben.

Verarbeitungsschicht. Der Kern eines Extraktionsplan-basierten IE-Systems sind generische und anwendungsspezifische IE-Module, die die eigentliche Logik für atomare Extraktionsoperationen kapseln. Sie können durch eine spezifische Sprache instantiiert, konfiguriert und zu einem IE-Programm komponiert werden. Ein Überblick zu IE-Modulen, Schnittstellen und der Sprache zur Komposition von IE-Programmen in *RapidUIM* wird in Kap. 3.1.2 gegeben.

Datenmodell. Aspekte, die das Datenmodell betreffen, sind z. B. die Verwaltung von Zwischenresultaten, die Speicherung oder Einbettung von Endresultaten etc. In Kap. 3.1.3 wird speziell auf das Laufzeit-Datenmodell eingegangen, welches die Entwicklung von generischen IE-Modulen ermöglicht und die Interoperabilität von Modulen sicherstellt. Metadaten beschreiben die Semantik von Textstrukturen und extrahierten Daten sowie Referenzen auf das verwendete Domänenwissen. Zusätzlich erhobene Verarbeitungsmetadaten, die in Kap. 3.2 eingeführt werden, ermöglichen die Synchronisation bei der parallelisierten Verarbeitung von Dokumenten.

Bevor Details zu den einzelnen Konzepten erläutert werden, wird in Kap. 3.1.1 ein Überblick zur Architektur von *RapidUIM* gegeben.

3.1.1 Architektur der RapidUIM-Plattform

In Abb. 3.1 ist ein schematischer Überblick über die Architektur von *RapidUIM* dargestellt. Die wesentlichen Bestandteile sind:

Schnittstellen zu Datenqellen. *RapidUIM* definiert zwei Typen von Datenquellen: Dokumente (siehe Abb. 3.1.(1)) und Domänenwissen (siehe Abb. 3.1.(2)). Domänenwissen kann z. B. in Form von Regulären Ausdrücken, Unternehmensdatenbanken oder anderen Arten von strukturierten Informationen zur Informationsextraktion genutzt werden. Für beide Typen von Datenquellen stehen Schnittstellen zur Erweiterung um spezifische Datenquellen zur Verfügung. Datenquellen werden unter anderem zur Konfiguration von IE-Modulen verwendet.

Modulbibliothek. Generische IE-Module werden in einer Modulbibliothek verwaltet, die die Wiederverwendung unterstützt. In Abb. 3.1.(3) sind exemplarisch Module, die an [KLR+08] zur Regel-basierten Extraktion angelehnt sind, abgebildet: ein Modul für Reguläre Ausdrücke (*REGEX*), für die Wörterbuch-basierte Extraktion (*DICT*), ein Modul zur Extraktion des linken oder rechten Kontext von bereits extrahierten Entitäten (*LC/RC*) und ein Modul zur Extraktion von Beziehungen auf Basis des Textes zwischen zwei Entitäten (*BETWEEN*). Zusätzlich ist ein Modul zum Markieren und/ oder Filtern verschiedener Wortarten (*POS*) dargestellt. Die Modulbibliothek kann beliebig erweitert werden.

Konfiguration und Komposition. Komponenten des Systems, wie z. B. IE-Module oder Komponenten zur Anbindung von Datenquellen haben spezifische Anforderungen bezüglich ihrer Konfiguration. Notwendige und optionale Parameter können deskriptiv spezifiziert und bei der Instantiierung einer Komponente entsprechend konfiguriert werden (siehe Abb. 3.1.(4)). Es steht ein Werkzeug zur Komposition von Modulinstanzen zu einem IE-System zur Verfügung, das die Modellierung von Extraktionsplänen anhand der Beschreibung des Datenflusses zwischen IE-Modulen ermöglicht (siehe Abb. 3.1.(5)).

Ausführung. Ein Extraktionsplan definiert die Abhängigkeiten zwischen IE-Modulen und Daten. Sie werden zur Laufzeit entsprechend dieser Abhängigkeiten ausgeführt (siehe Abb. 3.1.(6)). Der Datenaustausch zwischen IE-Modulen erfolgt durch Ein- und Ausgabe einer Teilmenge der Knoten des Graph-basierten Laufzeitdatenmodells. Die Extraktionslogik eines IE-Moduls wird auf die Menge der Eingabeknoten angewendet. Resultat der Operation ist eine Menge an Ausgabeknoten. Die Schnittstellen zwischen den Modulen sind dementsprechend definiert. Ein IE-Modul kann auf beliebige andere Knoten im Datenmodell, die durch Navigation im Graphen des Laufzeitdatenmodells von den Eingabeknoten aus erreichbar sind, zugreifen.

Persistenz. Zur Speicherung von IE-Ergebnissen wird ein RDBMS eingesetzt (siehe Abb. 3.1.(7)). Das Laufzeitdatenmodell wird in ein semantisch äquivalentes relationales Datenmodell überführt. Es unterstützt strukturierte Anfragen und eine Navigation auf extrahierten Daten wie in Kap. 1.2.3 beschrieben wurde. Zusätzliche erfasste Metadaten, z. B. zu den verwendeten Extraktionsmethoden und Domänenwissen, können beispielsweise zur Fehlersuche in Extraktionsprogrammen verwendet werden (siehe [BBM09a] für Details).

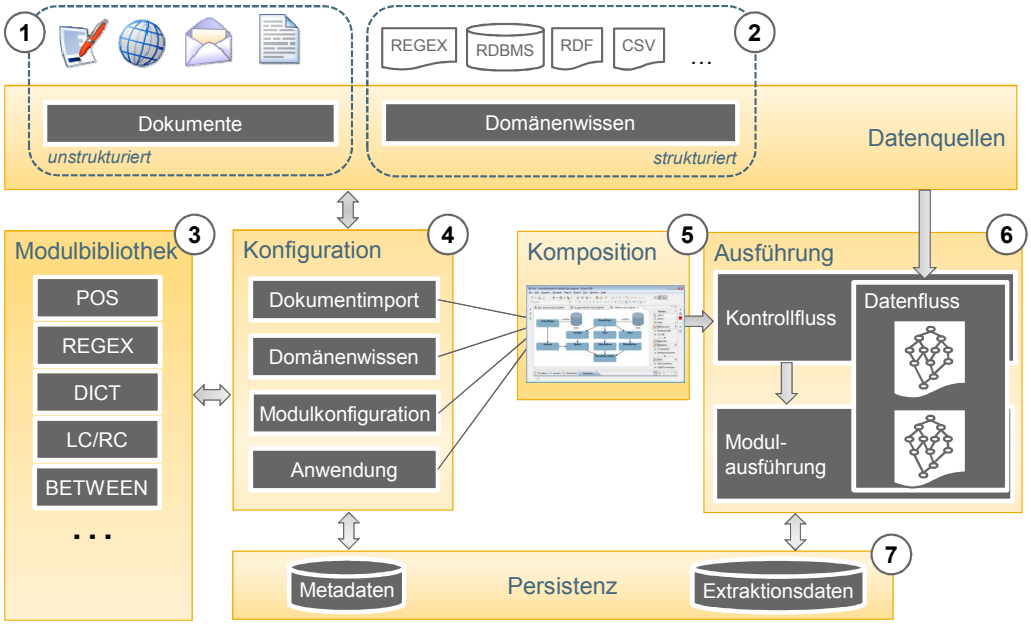


Abb. 3.1: Architektur von RapidUIM

RapidUIM unterstützt alle grundlegenden Mechanismen zur Entwicklung und Ausführung von anwendungsspezifischen IE-Systemen: die Entwicklung von IE-Modulen auf Basis eines gemeinsamen *Application Programming Interface (API)*, eine deskriptive Konfiguration und Komposition von IE-Modulen, und die effiziente Ausführung und Speicherung von Ergebnissen.

3.1.2 Komposition des Datenflusses

Grundlage des hier vorgestellten Ansatzes sind IE-Module mit einheitlichen Schnittstellen, die eine freie Komposition von IE-Modulen ermöglichen. IE-Module können in Kategorien unterteilt werden (siehe auch [BBLM09]), die ihre Schnittstellen und ihr allgemeines Verhalten festlegen. Wir definieren die drei folgenden Typen von Extraktionsmodulen:

Importmodule. Softwarekomponenten, die Dokumente ins System laden, sind die einfachste Art von Modulen in *RapidUIM*. Importmodule bilden Texte und Metadaten von Dokumenten in das Datenmodell von *RapidUIM* ab. Importmodule besitzen per Definition keine Vorgängermodule, sondern werden mit einer Datenquellenbeschreibung für den Dokumentimport konfiguriert.

Extraktionsmodule. Extraktionsmodule konsumieren zu einem Zeitpunkt genau einen Knoten des Graph-basierten Laufzeitdatenmodells (siehe Kap. 3.1.3), den es als Eingabedatum vom vorherigen IE-Modul erhält. Ein Eingabedatum kann der Dokumenttext oder ein anderes Zwischenergebnis vorheriger Operationen darstellen. Die Extraktionslogik wird dementsprechend für genau ein Datum spezifiziert. Die Operation, die auf den Knoten ausgeführt wird, ist unabhängig von vorherigen oder zukünftigen Eingabedaten. Ausgabe ist eine Menge von Daten. Extraktionsmodule dienen in der Regel dazu, einen Eingabetext in „kleinere" Bestandteile zu zerlegen. Typische Aufgaben von Extraktionsmodulen sind z. B. die morphologische Analyse, die anwendungsspezifische Segmentation eines Textes oder die Auswertung von Regulären Ausdrücken zur Extraktion von Entitäten. Extraktionsmodule können jedoch auch zum Filtern von Knoten, z. B. zur Auswertung von Nebenbedingungen verwendet werden. Das Verhalten von Extraktionsmodulen ähnelt dem einer *Map*-Funktion in *MapReduce*-basierten Ansätzen [RRP+07].

Aggregationsmodule. Aggregationsmodule unterscheiden sich von Extraktionsmodulen dadurch, dass sie eine Menge von Eingabedaten gemeinsam verarbeiten. Die Logik eines IE-Moduls definiert dementsprechend eine Operation auf einer Menge von Eingabeknoten des Laufzeitdatenmodells. Die Ausgabe ist wie bei Extraktionsmodulen eine Menge von Knoten. Aggregationsmodule dienen in der Regel dazu, Eingabedaten zu aggregieren, z. B. um Beziehungen zwischen Entitäten zu extrahieren. Auch Filter für die Überprüfung von komplexeren Nebenbedingungen können mit Hilfe von Aggregationsmodulen implementiert werden, wie z. B. die Selektion einer Teilmenge von „relevantesten" Entitäten für ein Textsegment. Aggregationsmodule verhalten sich ähnlich wie eine *Reduce*-Funktion in *MapReduce*-basierten Ansätzen.

RapidUIM adaptiert Verfahren aus dem Bereich der modellgetriebenen Softwareentwicklung [MCF03] zur Beschreibung von anwendungsspezifischen IE-Systemen. Die Konfigurationsmöglichkeiten von IE-Modulen, also notwendige und optionale Parameter, werden in einem abstrakteren, softwaretechnischen Meta-Modell beschrieben (dem *Eclipse Modeling Framework* – siehe z. B. [BBM03, Fav04]). Auch die notwendigen und möglichen Abhängigkeiten sind durch das Meta-Modell definiert, z. B. dass Importmodule keine Vorgängermodule erlauben, aber eine Datenquelle erfordern.

Die Verwendung eines softwaretechnischen Metamodells unterstützt zum einen eine deskriptive Beschreibung von notwendigen und optionalen Konfigurationsparametern und somit eine einfache Integration von IE-Modulen. Zum anderen können neben der hier diskutierten, weitere Beschreibungssprachen zur Komposition von IE-Systemen auf Basis eines stabilen Modells entwickelt werden. Alternative Beschreibungssprachen für Nutzer mit verschiedener Expertise diskutierten wir in [BBLM09]. Weiterhin wird die Komposition von IE-Modulen zu komplexeren IE-Modulen sowie deren Verwaltung in einer Modulbibliothek durch softwaretechnische Metamodelle sehr gut unterstützt.

Eine Datenfluss-orientierte Beschreibungssprache zur Komposition von Modulen zu einem IE-System ist in Abb. 3.2 illustriert. Das in Abb. 3.2.(2) dargestellte Beispiel zeigt einen zu dem in Kap. 2.1 durch Bsp. 2.2 dargestellten, in XLog beschriebenen IE-System äquivalentes System. Es extrahiert den Preis („Price") und die Grundstücksgröße („Area") von Häusern sowie Schulen in der Nähe („Schools") aus einer Menge von Webseiten („WebImport") mit einen bestimmten Preis und Grundstücksgröße („FilterByValue"). Die Resultate des Moduls „Schools" werden mit Eliteschulen aus anderen Webseite abgeglichen („ApproxMatch"). Ergebnisse, die alle Nebenbedingungen erfüllen, werden miteinander verbunden („Relate") und gemeinsam gespeichert.

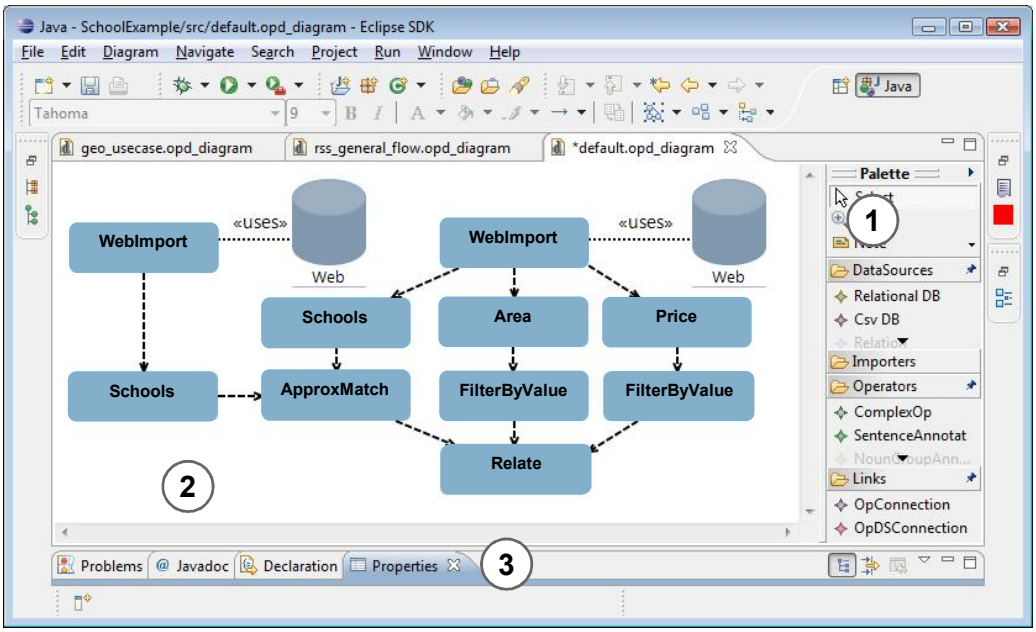


Abb. 3.2: Datenfluss-orientierte Beschreibungssprache für Informationsxtraktionssysteme

In Abb. 3.1.(1) sind Beispiele für generische IE-Module dargestellt, die durch Verschieben in den Bereich in Abb. 3.2.(2) instantiiert, entsprechend ihrer Aufgabe parametrisiert und optional nach ihrer Semantik benannt werden können. In dem Bereich, der in Abb. 3.2.(3) ausschnittsweise dargestellt ist, werden die Parameter eines IE-Moduls festgelegt.

3.1.3 Konzeptionelles Meta-Datenmodell

In bisherigen Extraktionsplan-basierten Ansätzen (z. B. [KLR+08,CBC+07]) werden zur Sicherstellung der Interoperabilität verschiedener IE-Module Dokumentstrukturen und Entitäten durch einen Typschlüssel und den Positionsindex im Dokument, dem so genannten „*Span*“ [FL04], repräsentiert (z. B. *Person* an Position $10 - 33$). Aggregierte Spans werden als Relationen dargestellt. Ein solches Datenmodell ist für viele Anwendungsfälle ausreichend, erschwert jedoch die Entwicklung von generischen IE-Modulen, die nicht nur auf einfachen *Spans*, sondern auf komplexer strukturierten Kontextdaten, wie z. B. einer hierarchischen Dokumentstruktur oder Graph-strukturierten Referenzdaten operieren.

Im Bereich des *Natural Language Processing* werden Baum-basierte Datenmodelle [MSM94, Pv08, LB10], z. B. zur Darstellung der Beziehung von morphologischen und/ oder grammatikalischen Einheiten von Sätzen, seit geraumer Zeit zur Darstellung syntaktischer Strukturen verwendet. Deren Basis ist ein einheitliches Verständnis von relevanten „Enitätstypen“ wie z. B. Substantiv, Substantivgruppe oder Subjekt.

Aktuelle Ansätze adaptieren die Idee eines Baum-basierten Datenmodells zur Beschreibung von nicht vordefinierten syntaktischen Merkmalen und Dokumentstrukturen in den IE-Bereich [Dec06, IR06, IS07]. Sie bilden im Vergleich zu *Span*-basierten Datenmodellen die Zusammenhänge zwischen verschiedenen syntaktischen Artefakten, die während der Verarbeitung von Textdaten eine Rolle spielen, explizit ab und erlauben damit einen unproblematischen

Zugriff auf Kontextdaten, wie z. B. der Dokumentstruktur, die durch vorherige Operationen abgebildet wurde.

Andere Ansätze im Bereich der Informationsextraktion, wie z. B. [MKS04, CSL+06, JKR+06, KSI+08, PNLH09] stellen semantische Abhängigkeiten, wie z. B. komplexere Beziehungen von Entitäten in einem Graph-basierten, Ontologie-artigen Datenmodell dar, insbesondere um strukturierte Anfragen an extrahierte Daten und die Navigation in Extraktionsergebnissen zu unterstützen. In einem Laufzeitdatenmodell verwendet, bietet dieser Ansatz einen wesentlichen Mehrwert hinsichtlich der Entwicklung von generischen IE-Modulen, da eine höhere Ausdrucksmächtigkeit im Vergleich zu einem relationalen Datenmodell erreicht und gleichzeitig von der konkreten Semantik der Daten abstrahiert wird.

Dokument-Konzept-Graphen. Das Meta-Datenmodell von *RapidUIM* (veröffentlicht in [BBM+09b]) vereinigt die verschiedenen Ideen der Graph-basierten Repräsentation von Dokumentstrukturen, Entitäten und komplexen Beziehungen. Ein Graph-basiertes Datenmodell ermöglicht nicht nur die Unterstützung von komplexen ad-hoc Anfragen an extrahierte Daten (siehe Kap. 1.2.3), sondern insbesondere auch die Entwicklung von generischen IE-Modulen, die zur Laufzeit durch Navigation im Graph auf Kontextinformationen zugreifen können. Das Meta-Datenmodell wird im Folgenden Dokument-Konzept-Graph (DKG) genannt, da es Dokumentstrukturen und semantisch bedeutsame Konzepte, wie Entitäten und deren Beziehungen aufeinander abbildet. Einen Dokument-Konzept-Graphen definieren wir wie folgt:

Definition 3.1.1 (*Dokument-Konzept-Graph*)

Ein Dokument-Konzept-Graph sei ein gerichteter Graph $DKG = (N, E)$, wobei N eine Menge von Knoten („Nodes") und E eine Menge von Kanten („Edges") definiert. Ein Knoten wird durch ein Quadrupel $N =< L, S, R, Q >$ repräsentiert. L („Label") stellt eine textuelle Repräsentation eines Merkmals oder Konzepts dar (z. B. Text eines Satzes oder die kanonische Form einer Entität). S („Span") sei der Positionsindex im Ursprungsdokument. $R =< T, I >$ („Reference") repräsentiere die Semantik eines Knotens und das Domänenwissen, auf Basis dessen der Knoten erzeugt wurde, wobei T dem Typ eines Knotens und I („Identity") eine Referenz auf das verwendete Domänenwissen entspreche (z. B. die verwendete Regel oder der Positionsindex in einem Wörterbuch). Q sei ein Vektor aus Metriken, die die Qualität eines Datums charakterisieren.

Beispiele für Metriken, die durch Q repräsentiert werden können, sind die Konfidenz oder die Relevanz eines Datums (formal in Kap. 5 definiert). Konfidenz beschreibt dabei die Vertauenswürdigkeit eines Datums und Relevanz die Wichtigkeit eines Datums, z. B. einer identifizierten Entität gegenüber einer anderen (potenziell) identifizierten Entität im selben Dokumentkontext.

Knoten eines DKG gehören zwei disjunkten Klassen an: $N = N_D \cup N_S$. Die Knotenmenge N_D repräsentiert Dokumente (Quellknoten des Datenmodells) und deren syntaktische Strukturen (z. B. Absätze oder Sätze), während die Knoten N_S semantisch bedeutsame Konzepte beschreiben, wie z. B. Entitäten oder Beziehungen.

Die Menge an Kanten $E \subseteq N \times N$ kann in drei Unterklassen unterschieden werden: $E_D \subseteq N_D \times N_D$ sei eine Menge an unidirektionalen Kanten zwischen Dokumentstruktur-Knoten, $E_{D\_S} \subseteq N_D \times N_S$ eine Menge an bidirektionalen Kanten zur Repräsentation von Beziehungen zwischen Dokumentstrukturen und semantischen Konzepten und $E_S \subseteq N_S \times N_S$ eine Menge an unidirektionalen Kanten zur Darstellung von semantischen Zusammengehörigkeiten innerhalb der Knotenmenge N_S.

Die Kanten E_D beschreiben eine partielle Ordnung von Dokumentstrukturen (notiert als $\leq_D$), die einander enthalten. Die Kanten E_{D_S} beschreiben den Übergang zwischen einer syntaktischen Einheit in der Dokumentstruktur und (möglichen) semantischen Bedeutungen wie z. B. eine Abbildung zwischen einer Wortgrupppe und einer Menge von potenziell durch sie referenzierte Entitäten.

IE-Module können entlang der Kanten im DKG navigieren. Das Meta-Datenmodell stellt damit ein mächtiges Konzept zur generischen Implementation von IE-Modulen zur Verfügung, das von der konkreten Semantik der extrahierten Daten abstrahiert und die Interoperabilität von IE-Modulen sicherstellt.

Beispiel für einen Dokument-Konzept-Graphen: Das Beispiel in Abb. 3.3 für den Anwendungsfall der Rechnungsverarbeitung (siehe Kap. 1.2.1) illustriert die Struktur eines DKG und die wichtigsten Daten, die während der Verarbeitung erfasst werden. Dokumentstrukturknoten N_D sind durch rechteckige Kästen und semantische Knoten N_S durch Ovale illustriert. Die partielle Ordnung von Dokumentknoten wird durch die Kantenmenge E_D repräsentiert (Zeile $\leq_D$ Segment $\leq_D$ Dokument).

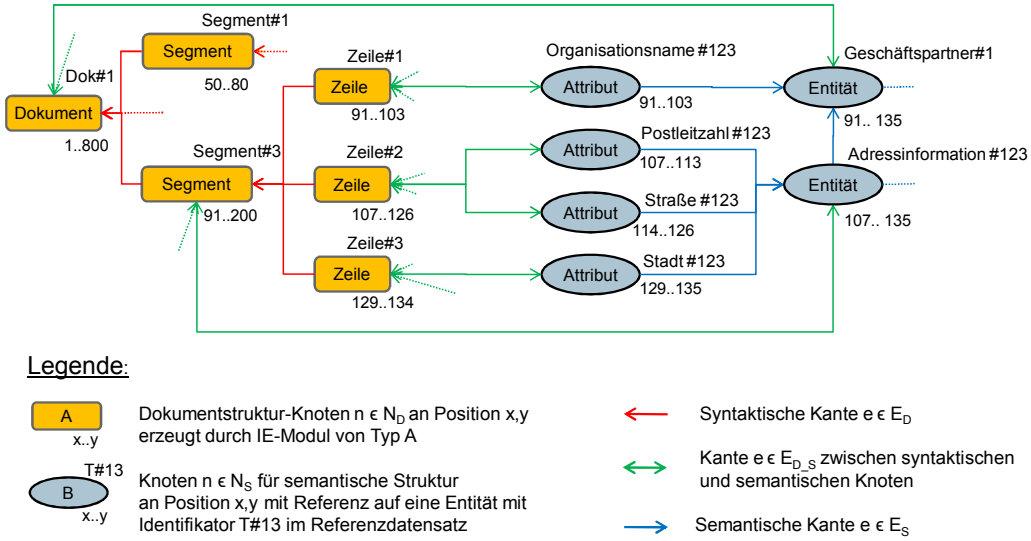


Abb. 3.3: Beispielstruktur für Dokument-Konzept-Graphen

Die Menge an Dokumentstruktur-Knoten N_D wird durch die Menge von Kanten E_{D_S} auf die Menge von semantischen Knoten N_S, in diesem Fall durch approximative Übereinstimmungen von Zeilen und Attributen, abgebildet. Die Ordnung $\leq_S$ bildet die Struktur der Aggregate von Attributen zu Entitäten im Referenzdatensatz ab. Entitäten werden weiterhin zu komplexeren Entitäten aggregiert (z. B. „Geschäftspartner#1“ besteht aus „Organisationsname#123“ und „Adressinformation#123“). Das Datenmodell bildet weiterhin höherwertige Entitäten zu Dokumentstrukturen ab (z. B. „Adressinformation #123“ zu „Segment#3“).

Das Datenmodell bietet Vorteile hinsichtlich der Teilbarkeit eines IE-Problems in feingranulare Operationen, der freien Kombinierbarkeit von IE-Modulen und der Entwicklung von generischen Modulen, die weit über die Möglichkeiten bisheriger Plattformen hinausgehen. Zum Beispiel wird für den in Abb. 3.3 dargestellten Anwendungsfall ein Standardmodul zum Import von Dokumenten verwendet. Zur Segmentation des Dokuments (Titel, Adressblöcke, Tabellenstrukturen etc.) sind ein oder mehrere anwendungsspezifische IE-Module nötig, die auf dem Eingabeknoten *Dokument* operieren. Ein Standard-IE-Modul bricht die anwendungsspezifisch extrahierten Segmente beispielsweise in *Zeilen*.

Der Abgleich zwischen *Zeilen* und *Attributwerten* kann durch ein generisches IE-Modul realisiert werden. Ein separates IE-Modul bildet auf Basis der Referenz-Metadaten ($R =< T, I >$) die zu *Attributknoten* gehörigen *Entitäten* des Referenzdatensatzes in das Datenmodell ab. Zusätzlich werden in diesem Beispiel zur indirekten Identifikation weitere zu *Entitäten* in Beziehung stehende *Entitäten* in das Dokumentmodell übertragen (z. B. „Geschäftspartner#1“).

Unabhängig von der Semantik der Daten und der Implementation vorheriger Module werden die vorher erzeugten Übereinstimmungen zwischen *Zeilen* und *Attributen* auf Basis der Kanten $E_{D\_S} \subseteq N_D \times N_S$ durch ein weiteres generisches Modul bewertet (Bewertung der Konfidenz). Auf Basis der Navigation entlang der Kanten $E_S \subseteq N_S \times N_S$ werden Relevanzwerte für *Entitäten*, z. B. einfach durch Abzählen von Kindknoten oder einer komplexeren Aggregation von Konfidenzwerten entlang der Graphstruktur (siehe Kap. 5), bestimmt. Verschiedene Algorithmen zur Relevanzbewertung können in *RapidUIM* als IE-Modul zur Verfügung gestellt, einfach ausgetauscht und getestet werden.

3.1.4 Zusammenfassung

In Kap. 3.1 stand die Entwicklung von anwendungsspezifischen IE-Systemen auf Basis von Extraktionsplänen im Mittelpunkt. Zusammenfassend kann festgestellt werden, das *RapidUIM* eine *API*, Schnittstellen und Datenmodell zur Entwicklung von anwendungsspezifischen und generischen IE-Modulen zur Verfügung stellt. Die einheitliche Schnittstellendefinition von IE-Modulen unterstützt eine flexible Integration und Wiederverwendung von Komponenten und fördert somit eine effiziente Entwicklung von IE-Systemen. Die Beschreibung von Modulen durch ein softwaretechnisches Metamodell erlaubt eine einfache Integration von neuen IE-Modulen und anderen Komponenten, wie z. B. zum Zugriff auf Datenquellen.

Insbesondere wird eine effiziente Entwicklung von IE-Systemen durch die Komposition von IE-Modulen auf Basis der hier vorgestellten Datenfluss-orientierten Beschreibungssprache und die Verwaltung von IE-Modulen in einer Modulbibliothek unterstützt. Das Graph-basierte Meta-Datenmodell unterstützt die Implementation von generischen IE-Modulen und stellt deren Interoperabilität sicher. Die Ausführung von komponierten IE-Systemen steht im folgenden Kapitel im Mittelpunkt.

3.2 Konzept zur parallelisierten Dokumentverarbeitung

Im Folgenden betrachten wir die Dokumenten-orientierte Verarbeitung in der Informationsextraktion. Extraktionspläne, die in *RapidUIM* modelliert wurden, beschreiben (unter anderem) die Abhängigkeiten zwischen verschiedenen Extraktionsaufgaben für ein Dokument. Sie können also grundsätzlich in herkömmliche, sequentielle Ausführungspläne übersetzt werden. Eine solche Aufgaben-orientierte Ausführung von Extraktionsplänen mit *Workflow*-basierten Plattformen wird in Kap. 3.2.1 betrachtet. Anhand des dort diskutierten Beispiels werden Einschränkungen bisheriger Dokumentverarbeitungsplattformen hinsichtlich der Effizienzsteigerung durch eine parallele Datenverarbeitung diskutiert.

In Kap. 3.2.2 beschreiben wir detailliert das Konzept zur Intra-Dokument-Parallelisierung von IE-Systemen. In der parallelen Datenverarbeitung wird im Allgemeinen in drei Klassen der Parallelität unterschieden (siehe z. B. [GTA06]): die parallele Ausführung von mehreren Programminstanzen, die Aufgabenparallelität und die Datenparallelität. Die hier vorgestellten Mechanismen ermöglichen die Kombination aller genannten Formen der Parallelität für IE-Systeme, wobei wir im Folgenden insbesondere auf die Aufgabenparallelität und die Datenparallelität eingehen. Eine parallele Ausführung von mehreren Programminstanzen zur parallelen Verarbeitung von verschiedenen Dokumenten ist ohne weiteres möglich.

Wir diskutieren in Kap. 3.2.2 drei Konzepte: 1.) Zur Realisierung einer Aufgabenparallelität wird die Idee des sequentiellen Ausführunsplans aufgegeben und eine nebenläufige Ausführung von untereinander unabhängigen IE-Modulen ermöglicht. 2.) Die Datenparallelität wird durch die Interpretation von Zwischenresultaten als Datenstrom erreicht. Ein effizienter Synchronisationsmechanismus ermöglicht die Parallelität auch in Situationen, in denen Operationen eines IE-Moduls von Zwischenresultaten mehrerer vorgelagerter Module abhängig sind. 3.) Die Datenstrom-basierte Verarbeitung und Synchronisation im Falle von Da-

tenabhängigkeiten unterstützt eine erweitere Datenparallelität, da mehrere Instanzen desselben IE-Moduls parrallel Zwischenresultate von vorherigen Operationen konsumieren können.

In Kap. 3.2.3 werden die theoretisch erreichbaren Laufzeitverbesserungen zur Illustration von Einflussfaktoren bezüglich der potentiellen Beschleunigung hergeleitet. Zum einen schätzen wir die Laufzeitverbesserungen unter Annahme unendlicher Ressourcen im Vergleich zu sequentiellen Ausführungsplänen ab. Zum anderen diskutieren wir das Laufzeitverhalten unter beschränkten Ressourcen und Einflussfaktoren für den *Overhead*, der durch eine Parallelisierung entsteht. Die Vorhersagen werden in Kap. 3.3 experimentell validiert.

3.2.1 Herkömmliche Dokumentverarbeitung

Bevor auf die parallelisierten Dokumentverarbeitung eingegangen wird, soll eine Diskussion zum einen zeigen, dass in *RapidUIM* entwickelte IE-Systeme auf herkömmlichen *Workflow*-basierten Dokumentverarbeitungsplattformen wie z. B. *UIMA* [FL04] oder GATE [CMBT02] ausgeführt werden können und zum anderen die Einschränkungen einer derartigen Lösung aufzeigen.

Als laufendes Beispiel zur Erläuterung von Problemstellungen und Lösungskonzepten wird im Folgenden der relativ einfache, in Abb. 3.4 dargestellte Extraktionsplan verwendet, dessen Ziel es ist, *Treffen* aus Texten zu extrahieren. Jedes *Dokument* wird in diesem Beispiel in *Absätze* zerlegt, die die Grenzen für das Auftreten von *Treffen* festlegen. Innerhalb der *Absätze* werden *Räume* für den Ort eines Treffens extrahiert. Der *Termin* eines *Treffens* besteht aus *Zeit* und *Datum*, die zusammen innerhalb eines *Satzes* auftreten müssen. *Termine* werden mit *Räumen* im selben *Absatz* zu *Treffen* aggregiert (ein *Absatz* entspricht quasi dem Verbundprädikat von *Terminen* und *Räumen*).

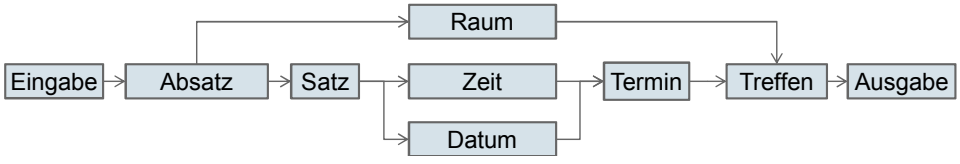


Abb. 3.4: Extraktionsplan zur Identifikation von Treffen

Für die sequentielle Ausführung dieses Extraktionsplans (der Überführung in einen Ausführungsplan) gibt es verschiedene Varianten, z. B. die Folge: „*Absatz*, *Raum*, *Satz*, *Zeit*, *Datum*, *Termin*, *Treffen*“ oder die Sequenz: „*Absatz*, *Satz*, *Zeit*, *Datum*, *Termin*, *Raum*, *Treffen*“. Beides sind valide sequentielle Ausführungspläne, da sie die definierten Aufgabenabhängigkeiten berücksichtigen. Die Unabhängigkeit von verschiedenen Aufgaben, z. B. zur Extraktion von *Zeit* und *Datum*, wird in herkömmlichen Dokumentverarbeitungsplattformen nicht abgebildet. Vielmehr wird der Ausführungsplan als sequentieller Kontrollfluss interpretiert.

Herkömmliche Plattformen nutzen *Blackboards* zum Datenaustausch zwischen Modulen. Für jedes Dokument wird eine Art Datencontainer angelegt, in denen Objektmengen anhand ihrer Typschlüssel zwischengespeichert werden. Jedes Modul verarbeitet in einer Operation immer alle Objekte eines Typs für das gesamte Dokument. Eine Unterscheidung zwischen Operationen, die auf einzelnen Zwischenergebnissen ausgeführt werden können (z. B. die Anwendung eines Regulären Ausdrucks), und Operationen, die eine Datenmenge verarbeiten (z. B. zur Extraktion von Beziehungen), ist von API-Seite nicht vorgesehen.

IE-Module adressieren in herkömmlichen Ansätzen die durch sie zu verarbeitende Objektmenge im Datencontainer anhand von Typschlüsseln. In einem Extraktionsplan, der mit *RapidUIM* erstellt wurde, werden die Beziehungen von Datenein- und -ausgaben explizit modelliert. Durch Definition von einem Objekttyp pro IE-Modul (mit zugehörigem Typschlüssel) und entsprechende Konfiguration von Nachfolgemodulen können derart Abhängigkeiten auch in herkömmliche Plattformen ohne manuellen Eingriff abgebildet werden.

Die Abhängigkeiten zwischen einzelnen Daten, z. B. dass zwei Entitäten, die zu einer Beziehung aggregiert werden sollen, in einem Satz vorkommen, werden in herkömmlichen Ansätzen nicht explizit dargestellt, sondern können lediglich auf Basis ihrer Positionsindizes im Dokument bestimmt werden. Die Bestimmung von Abhängigkeiten zwischen Daten, die zur Synchronisation von IE-Modulen bei einer Parallelisierung notwendig ist, gestaltet sich daher aufwändig.

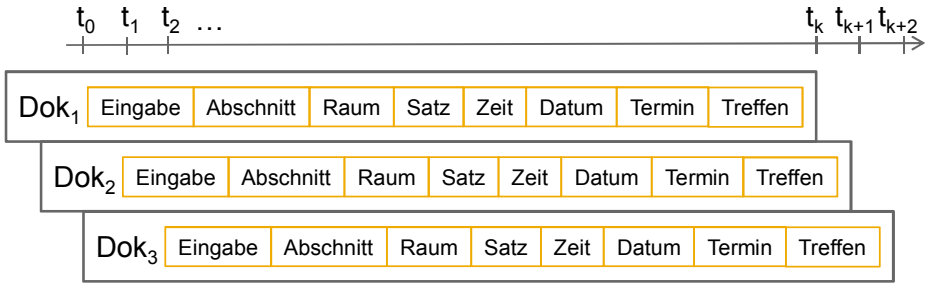


Abb. 3.5: Herkömmliche parallele Ausführung des Extraktionsplans aus Abb. 3.4

In Abb. 3.5 ist die Verarbeitung eines Dokuments (siehe z. B. Dok_1) für den Extraktionsplan aus Abb. 3.4 dargestellt. Die absolute Verarbeitungszeit für ein Dokument, welche in Abb. 3.5 durch t_k illustriert ist, entspricht der kumulierten Verarbeitungszeit für jedes einzelne Zwischenergebnis, das im *Workflow* erzeugt und durch ein nachgeordnetes Modul verarbeitet wird. Um einen hohen Dokumentendurchsatz für IE-Systeme zu realisieren, können im Allgemeinen einzelne Dokumente eines Dokumentenkorpus durch parallel, möglicherweise verteilt arbeitende Programminstanzen, wie in Abb. 3.5 dargestellt, analysiert werden.

Die durchschnittliche Dokumentenverarbeitungszeit kann auf diese Art und Weise im Allgemeinen proportional zur Anzahl der eingesetzten Prozessoren verbessert werden (siehe z. B. [LSH08]). Die Grenzen der Skalierbarkeit werden hauptsächlich durch Ein- und Ausgabeoperationen, z. B. für den Dokumentenzugriff und die Speicherung von Endresultaten, festgelegt.

Die absolute Verarbeitungszeit für ein einzelnes Dokument wird durch bisherige Ansätze zur parallelen Verarbeitung auf Basis mehrerer Programminstanzen jedoch nicht verbessert. Dies ist aber wünschenswert, wenn geringe Antwortzeiten wie z. B. bei Extraktionsdiensten relevant sind. Eine Intra-Dokument-Parallelisierung, die eine Aufgaben- und Datenparallelität unterstützt, verspricht eine solche Verbesserung.

Herkömmliche Dokumentverarbeitungsplattformen modellieren weder konkrete Abhängigkeiten von Aufgaben, so dass die Unabhängigkeit bestimmter Aufgaben nicht bestimmt werden kann, noch die Abhängigkeiten einzelner Zwischenergebnisse, welche z. B. eine nebenläufige Aggregation von *Datumsangaben* und *Zeiten* zu *Terminen* im Kontext eines *Satzes* für das Beispiel in Abb. 3.4 erlaubt. Weiterhin widerspricht das Prinzip in einer Operation, alle Objekte gleichen Typs für ein Dokument zu verarbeiten, einer Datenparallelität.

Eine Laufzeitverbesserung für Einzeldokumente erfordert daher eine neuartige Dokumentverarbeitung, bei der nicht die sequentielle Abarbeitung von Aufgaben pro Dokument, sondern die Abhängigkeiten von Aufgaben und Daten im Mittelpunkt steht. Die bisher beschriebenen Konzepte der *RapidUIM*-Plattform beschreiben die Aufgabenabhänigkeiten in Extraktionsplänen und Datenabhängigkeiten im Datenmodell und bilden daher eine notwendige Grundlage für nachfolgend beschriebene Konzepte zur Intra-Dokument-Parallelisierung.

3.2.2 Parallele Verarbeitung von Einzeldokumenten

Wie bereits beschrieben können Extraktionsoperationen in den meisten Anwendungsfällen auf verschiedenen Dokumenten parallel ausgeführt werden. Eine Intra-Dokument-Parallelisierung mit Aufgaben- und Datenparallelität erfordert die Sicherstellung der Unabhängigkeit von Einzeloperationen. Unabhängigkeiten ergeben sich indirekt aus der

Betrachtung von Abhängigkeiten, die definiert und geprüft werden können. Durch Betrachtung des Extraktionsplans in Abb. 3.6 lassen sich folgende drei Arten von Abhängigkeiten identifizieren:

Aufgabenabhängigkeiten: Zum Ersten bestehen Abhängigkeiten zwischen einzelnen Aufgaben, wie z. B. zwischen der Extraktion von *Sätzen* und *Zeiten* in Abb. 3.6. Voneinander unabhängige Aufgaben, wie z. B. zur Extraktion von *Räumen* und *Zeiten*, lassen sich ohne weiteres parallelisieren.

Direkte Datenabhängigkeiten: Zwischen einem IE-Modul und seinen unmittelbar vorgelagerten IE-Modulen besteht eine direkte Datenabhängigkeit. Die Extraktion einer *Zeit* setzt z. B. voraus, dass ein Zwischenergebnis von Typ *Satz* existiert. Unabhängig ist jedoch die Extraktion von *Zeiten* aus verschiedenen *Sätzen*. Während der Satzsegmentierung kann demnach das Modul zur Extraktion von *Zeiten* bereits zur Verfügung stehende Eingabedaten (*Sätze*) parallel weiterverarbeiten. Weiterhin können verschiedene Instanzen von IE-Modulen zur Extraktion von *Zeiten* nebenläufig verschiedene *Sätze* verarbeiten.

Indirekte Datenabhängigkeiten: Aggregationsmodule sind von der Verfügbarkeit einer bestimmten Menge von Zwischenergebnissen vorgelagerter Module abhängig. Im Extraktionsplan in Abb. 3.6 hängt zum Beispiel die Extraktion von *Treffen* von der Verfügbarkeit aller Zwischenergbnisse von Typ *Termin* und *Raum* im Kontext eines *Absatzes* ab. Im Umkehrschluss lassen sich also *Treffen* in verschiedenen *Absätzen* unabhängig voneinander extrahieren.

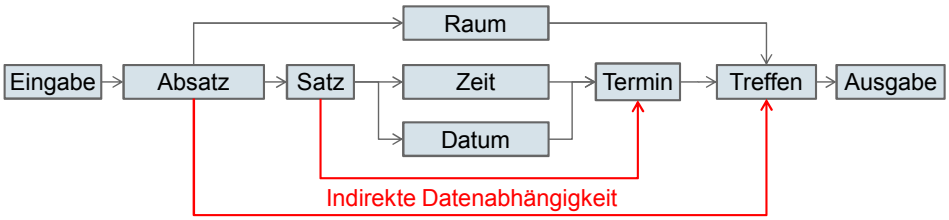


Abb. 3.6: Indirekte Datenabhängigkeiten bei der Extraktion von Treffen

Eine erweiterte Parallelisierung kann z. B. durch die Verteilung unabhängiger Operationen auf die Rechenresourcen eines Mehrprozessorsystems und einem effizienten Datenaustausch zwischen IE-Modulen auf Basis gemeinsam genutzter Hauptspeicherbereiche realisiert werden. Im folgenden Unterkapitel wird das Konzept zur nebenläufigen Dokumentverarbeitung für die Fälle von Aufgabenabhängigkeit und direkter Datenabhängigkeit diskutiert. Die nebenläufige Verarbeitung im Falle von indirekten Datenabhängigkeiten erfordert eine effiziente Synchronisation und wird im darauf folgenden Unterkapitel erläutert.

Nebenläufige Verarbeitung auf Basis von Datenströmen

Ein allgemeines Prinzip zur Parallelisierung von Datenverarbeitungskomponenten in Informationssystemen, das zum Beispiel in Datenstromsystemen eingesetzt wird (siehe z. B. [BBD+02]), aber auch Grundlage des *Pipelining* in Datenbanken [DG92] ist, basiert auf einer Entkopplung von einzelnen Datenverarbeitungskomponenten auf Basis von Warteschlangen. Ein Kontrollfluss im herkömmlichen Sinne existiert in solchen Systemen nicht. Die Aufgabe einer Kontrollinstanz, die die Komposition von Datenverarbeitungskomponenten interpretiert, beschränkt sich auf die Initialisierung von Komponenten und die Fehlerbehandlung (z. B. im Falle von Überlastsituationen). Im Folgenden wird die Adaption dieses Konzeptes für Extraktionsplan-basierte IE-Systeme erläutert.

Das Konzept der Datenstrom-basierten Dokumentverarbeitung für die Informationsextraktion kann am besten anhand eines Beispiels erläutert werden. In Abb. 3.7 ist die Verarbeitung für den in Abb. 3.7.(1) dargestellten Extraktionsplan illustriert. Der Ausführungsplan im Beispiel enthält zur Vereinfachung nur ein Aggregationsmodul (6 in Abb. 3.7.(1)).

Zur Initialisierung eines IE-Systems werden alle IE-Module eines Ausführungsplans durch einen *Controller* (siehe Abb. 3.7.(2)) instantiiert und über ihre direkten Abhängigkeiten zu anderen IE-Modulen informiert. Jeder Typ von IE-Modul besitzt genau eine Ausgabe- und eine Eingabewarteschlange (siehe Abb. 3.7.(3)). Sobald ein IE-Modul instantiiert wurde, verarbeitet es Daten unabhängig von anderen Komponenten des Systems. Zum Beispiel konsumieren 3 und 5 in Abb. 3.7.(3) die Ausgaben des Moduls 2.

Die Unterscheidung von Ein- und Ausgabewarteschlangen ist notwendig, da potenziell mehrere unabhängige IE-Module die Daten eines vorgelagerten IE-Moduls konsumieren. IE-Module generieren Daten, speichern sie in eigenen Hauptspeicherbereichen ab, auf die effizient von anderen Modulen zugegriffen werden kann, und reihen eine Referenz auf die erzeugten Daten in ihrer Ausgabewarteschlange ein. Ausgabewarteschlangen propagieren die Datenreferenzen zu den Eingabewarteschlangen unmittelbar nachgelagerter Module.

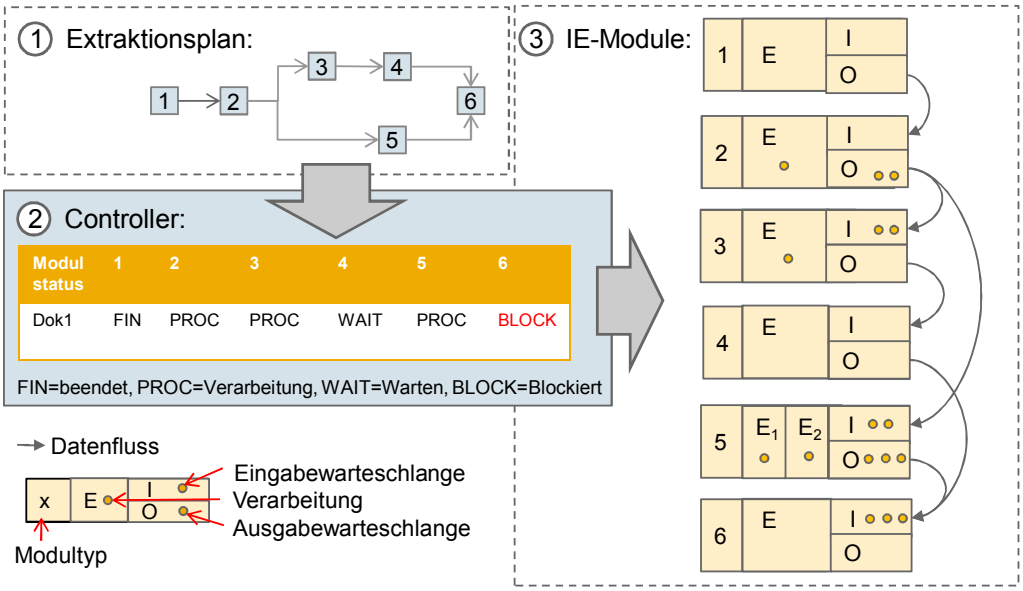


Abb. 3.7: Beispiel zur parallelisierten Ausführung von Extraktionsplänen

Es können dank der Entkopplung von IE-Modulen mehrere Instanzen eines Modultyps zur Datenverarbeitung, wie Abb. 3.7.(3) für Modul 5 dargestellt, erzeugt werden. Die Modulinstanzen operieren auf einer gemeinsamen Ein- und Ausgabewarteschlange, arbeiten aber davon abgesehen autonom. Die Anzahl von Modulinstanzen kann zum einen vorkonfiguriert werden. Zum anderen werden im Falle von Überlastsituationen, die sich durch „Staus“ in den Warteschlangen kennzeichnen, Modulinstanzen zur Laufzeit hinzugefügt, wenn ungenutzte Systemressourcen zur Verfügung stehen.

Der Status jedes IE-Moduls für das aktuelle Dokument wird durch einen *Controller* überwacht, so dass z. B. die Fertigstellung eines Dokumentes festgestellt werden kann oder eine Fehlerbehandlung möglich ist. Zu dem in Abb. 3.7.(3) dargestellten Zeitpunkt hat das IE-Modul 1 die Verarbeitung für das Dokument *Dok1* beendet, die IE-Module 2, 3 und 5 verarbeiten aktuell Daten und das Modul 4 erwartet Eingabedaten.

Das Aggregationsmodul 6 in Abb. 3.7.(3) ist im Status *Blockiert*, da zum dargestellten Zeitpunkt nicht alle notwendigen Eingabedaten auf Grund einer indirekten Datenabhängigkeit zur Verfügung stehen. Das Modul kann Eingabedaten erst verarbeiten, sobald die Module 3,

4 und 5 die Verarbeitung eines Ausgabedatums von 2 fertiggestellt haben. Für IE-Module zur Aggregation von Zwischenergebnissen sind also Mechanismen zur Synchronisation und effizienten Überprüfung von indirekten Datenabhängigkeiten notwendig.

Synchronisation bei indirekten Datenabhängigkeiten

In vielen Anwendungsfällen ergibt sich eine indirekte Datenabhängigkeit zwischen Modulen direkt aus dem Extraktionsplan. Für das Beispiel aus Abb. 3.7.(1) generiert das Modul 2 die Daten, die die „Grenzen" von unabhängigen Eingabedatenpartitionen für das Modul 6 definiert. Das Modul, das die für die Synchronisation relevanten Daten erzeugt, wird in derartigen Anwendungsfällen durch den ersten gemeinsamen Vorgänger der unmittelbar vorgelagerten IE-Module bestimmt.

In einigen Anwendungsfällen soll auf Basis eines weiter vorgelagerten Moduls synchronisiert werden. In Kap. 2.2 wurde ein derartiger Fall für die Disambiguierung von Entitäten im gesamten Dokumentkontext erläutert. Die Einführung von optionalen Synchronisationsabhängigkeiten im Extraktionsplan erlaubt es, indirekte Datenabhängigkeiten zu modellieren, wenn diese sich nicht direkt aus dem bestehenden Extraktionsplan ergeben.

Die Problemstellungen, die sich bei der Konzeption der Parallelisierung im Falle von indirekten Datenabhängigkeiten stellen, sind:

- Wie können zusätzliche Metadaten zur Synchronisation effizient erhoben werden, ohne dass die eigentliche Extraktionslogik hinsichtlich der Verarbeitungszeiten negativ beeinflusst wird?
- Wie kann eine effiziente Überprüfung von Abhängigkeiten zur Synchronisations realisiert werden?

Erfassung von Verarbeitungsinformationen. Die Synchronisation im Falle von indirekten Datenabhängigkeiten erfordert die Erhebung von zusätzlichen Verarbeitungsinformationen, die wenn möglich direkt im Datenstrom zwischen den Modulen zu kodieren sind, so dass IE-Module Daten autonom und unabhängig von anderen Komponenten des Systems verarbeiten können. Weiterhin muss sichergestellt werden, dass die Verarbeitungsinformationen effizient erhoben werden und diese Erhebung nur minimal die eigentliche Ausführung der Extraktionslogik verzögert.

Die Verfügbarkeit aller notwendigen Daten kann in einem hoch parallelen System nicht direkt überprüft werden. Vielmehr ist eine Prüfung hinsichtlich der Fertigstellung der Verarbeitung von Daten durch Module notwendig, die für eine Synchronisation relevant sind. Aufgrund des Datenaustauschs in gemeinsamen Hauptspeicherbereichen können IE-Module zur Laufzeit ohne Performanceeinbußen lesend auf Kontextinformation in einem Dokument-Konzept-Graph (DKG) zugreifen. Eine Erweiterung des konzeptionellen Datenmodells aus Kap. 3.1.3 um Metadaten zum aktuellen Verarbeitungszustand von Daten erscheint daher sinnvoll. Wir definieren Verarbeitungsinformationen, die in einem DKG kodiert werden, wie folgt:

Definition 3.2.1 (*Verarbeitungsinformationen im DKG*)

Ein Knoten eines DKG, vormals definiert als $N =< L, S, R, Q >$, mit Text L, Positionsindex S, Referenzdatum R und Qualitätsmetriken Q, wird erweitert zu $N =< L, S, R, Q, V >$, wobei V die Verarbeitungsinformationen repräsentieren. Verarbeitungsinformationen V werden als Menge von Paaren definiert, wobei ein Paar aus einer Referenz auf das IE-Modul, das die Verarbeitung eines Eingabedatums fertiggestellt hat, und einer Referenz auf das zuletzt durch das IE-Modul ausgegebene Zwischenergebnis besteht.

Auf die Verwendung des letzten Zwischenergebnisses wird im Folgenden noch eingegangen. Technisch ist die beschriebene Erhebung von Verarbeitungsinformationen effizient zu realisieren, da IE-Module während der Verarbeitung auf DKG-Knoten der Eingabewarteschlange

iterieren. Bevor der nächste Knoten der Eingabewarteschlange zur Verarbeitung freigegeben wird, können die Verarbeitungsinformationen ohne aufwändige Operationen an den vorherig verarbeiteten Knoten angehangen werden.

Überprüfung von indirekten Datenabhängigkeiten. Ein Extraktionsmodul erhält als Eingabe die DKG-Knoten direkt vorgelagerter Module. Eine Überprüfung von Verarbeitungsinformationen aller Knoten, die direkt oder indirekt mit einem Eingabeknoten verbunden sind, ist wenig effizient. Vielmehr muss der Test von Verarbeitungsinformationen von dem DKG-Knoten ausgehen, der die Grenzen der Datenunabhängigkeit für nachfolgende Module festlegt und rekursiv im DKG fortgeführt werden.

In Alg. 1 ist die Initialisierung und Ausführung von Aggregationsmodulen sowie eine vereinfachte Version der Synchronisationsmechanismen in Pseudocode dargestellt. Die Darstellung in Alg. 1 basiert zu Illustrationszwecken auf einem „*Aktiven Warten*". Mit entsprechenden Benachrichtigungsmechanismen zwischen verschiedenen autonom arbeitenden Modulen, die von der verwendeten Laufzeitumgebung und Programmiersprache abhängig sind, kann die Synchronisation effizienter gestaltet werden. Im Folgenden erläutern wir im Einzelnen die wichtigsten in Alg. 1 dargestellten Konzepte.

Zur gerichteten Überprüfung von Verarbeitungsinformationen werden zum Ersten Aggregationsmodule im Vergleich zu Extraktionsmodulen so modifiziert, dass sie neben den Daten unmittelbar vorgelagerter Module, auch Daten der Synchronisationsquellen konsumieren (siehe Alg. 1.(1)– (12)). Es ist zu bedenken, dass es dabei durchaus mehrere Synchronisationsquellen geben kann (siehe Eingabe von Alg. 1 sowie Zuordnung in Alg. 1.(18)–(20))

Zur Verdeutlichung weiterer Problemstellung ist in Abb. 3.8.(1) auf S. 61 nochmals ein Beispiel dargestellt. Modul [6] ist hier von der vollständigen Verarbeitung der Zwischenergebnisse von [2] durch [3], [4] und [5] abhängig. Zur gerichteten Überprüfung von Verarbeitungsinformationen konsumieren Aggregationsmodule die für die Synchronisation relevanten Daten direkt (gestrichelter Pfeil in Abb. 3.8.(1) und Alg. 1.(15)-Alg. 1.(16)).

In Abb. 3.8.(1) sind weiterhin zwei Gegebenheiten dargestellt, die bei der Synchronisation eine Rolle spielen: Zum einen bestehen zwei so genannte Abhängigkeitspfade und zwar [2] $\rightsquigarrow$ [5] $\rightsquigarrow$ [6] und [2] $\rightsquigarrow$ [3] $\rightsquigarrow$ [4] $\rightsquigarrow$ [6]. Ein Abhängigkeitspfad ist definiert als:

Definition 3.2.2 (*Abhängigkeitspfad*)

Ein Abhängigkeitspfad entspricht einer miteinander verbundenen Modulsequenz in einem Extraktionsplan, die mit dem Modul (der Synchronisationsquelle) beginnt, das die Quelldaten generiert, die für die Überprüfung von indirekten Datenabhängigkeiten relevant sind, und mit dem Modul endet (dem Synchronisationsziel), welches einer Synchronisation bedarf.

Wir merken an, dass bei der Überprüfung von Verarbeitungsinformationen das Synchronisationsziel nicht betrachtet wird. Für beide Abhängigkeitspfade in Abb. 3.8.(1) sind die Verarbeitungsinformationen einzeln zu prüfen. Zum anderen ist der Abhängigkeitspfad [2] $\rightsquigarrow$ [3] $\rightsquigarrow$ [4] $\rightsquigarrow$ [6] mehrstufig. Es muss demnach geprüft werden, ob [3] einen DKG-Knoten von [2] vollständig verarbeitet und zusätzlich, ob [4] die Bearbeitung aller Zwischenergebnisse, die daraufhin von [3] erzeugt wurden, fertiggestellt hat.

Die Idee ist nun, dass ausgehend von einem DKG-Knoten, der die „Grenzen" der Unabhängigkeit von nachfolgend erzeugten Daten definiert, für jeden Abhängigkeitspfad die Verarbeitungsinformationen überprüft werden (siehe Alg. 1.(21)–(25)). Im Falle eines positiven Tests, dass z. B. ein Datum von [2] durch [3] verarbeitet wurde, wird rekursiv mit den in den Verarbeitungsinformationen hinterlegten Referenz auf das letzte Ausgabedatum fortgefahren (siehe Alg. 2). Eine Überprüfung der Verarbeitungsinformationen von DKG-Knoten, die keine zusätzlichen Informationen hinsichtlich des Verarbeitungszustandes liefern, kann auf diese Weise ausgeschlossen werden.

Algorithmus 1 Aggregationsmodul(): Ausführung von Aggregationsmodulen

```
Input: Modul[] synchronisationsModule, Modul[] eingabeModule
 1:   // Initialisierung.
 2: Warteschlange ausgabeWS   // Ausgangswarteschlange.
 3: Warteschlange eingabeWS   // Eingabewarteschlange.
 4: Warteschlange syncWS   // Synchronisationswarteschlange.
 5:   // Vereinige Synchronisationswarteschlangen.
 6: for all (syncModul ∈ synchronisationsModule) do
 7:     syncWS.add(syncModul.warteSchlange())
 8: end for
 9:   // Vereinige Eingabewarteschlangen.
10: for all (eingabeModul ∈ eingabeModule) do
11:     eingabeWS.add(eingabeModul.warteSchlange())
12: end for
13:   // Verarbeitung.
14: loop
15:       // Überprüfe Status der Synchronisationswarteschlange
16:     if (syncWS.enthältEingaben()) then
17:         for all (syncKnoten ∈ syncWS) do
18:             // Zuordnung zu Synchronisationsquelle
19:             for all (syncMod ∈ synchronisationsModule) do
20:                 if (syncKnoten.wurdeVerarbeitet(syncMod)) then
21:                     // Prüfe Fertigstellung für alle Module im ...
22:                     // Abhängigkeitspfad, die dem Modul folgen.
23:                     for all (modul ∈ syncMod.abhängigeModule()) do
24:                         warteAufFertigstellung(syncKnoten,modul)   // siehe Alg. 2
25:                     end for
26:                 end if
27:             end for
28:             // Partition von Eingabeknoten für syncKnoten
29:             Knoten[] zuVerarbeitendeKnoten   // aktuell zu verarbeitende Knoten
30:             for all (eingabeKnoten ∈ eingabeWS) do
31:                 if (eingabeKnoten.istZugehörig(syncKnoten)) then
32:                     zuVerarbeitendeKnoten.fügeHinzu(eingabeKnoten)
33:                 end if
34:             end for
35:             // Verarbeite fertgigestellte Knoten und propagiere Ergebnisse –
36:             // Rückgabe ist das zuletzt erzeugte Datum
37:             Knoten letzterAusgabeKnoten = verarbeite(zugehörigeKnoten,ausgabeWS)
38:             // Schreibe Verarbeitungsinformationen.
39:             markiereAlsVerarbeitet(zuVerarbeitendeKnoten,letzterAusgabeKnoten)
40:         end for
41:     else
42:         warte()   // Warte auf Resultate
43:     end if
44: end loop
```

Daten aus den unmittelbar vorgelagerten IE-Modulen werden den für die Synchronisation relevanten Daten auf Basis der DKG-Struktur zugewiesen (Alg. 1.(30)–(34)). Im Beispiel in Abb. 3.8.(1) werden demnach Ergebnisse von [4] und [5] in der Warteschlange von [6] anhand der für die Synchronisation relevanten Daten von [2] gruppiert. In einer Eingabewarteschlange eines Aggregationsmoduls entstehen dadurch Datenblöcke, die unabhängig voneinander verarbeitet werden können (Alg. 1.(35)–(40)).

Das Beispiel in Abb. 3.8.(2) illustriert das Vorgehen nochmals anhand eines Beispielworkloads. Es ist ein DKG, der bei der Verarbeitung eines Dokuments durch den Extraktionsplan aus Abb. 3.8.(1) entstehen könnte, zu drei verschiedenen Zeitpunkten (t_3,t_5 und t_7) skizziert. Kreise repräsentieren DKG-Knoten, die entsprechend der Module in Abb. 3.8.(1) nummeriert sind.

Algorithmus 2 warteAufFertigstellung(): Rekursiver Test auf Fertigstellung

```
Input: Modul[] zuPrüfenderKnoten, Modul[] zuPrüfendesModul
 1: loop
 2:         // Prüfe ob Knoten durch Modul verarbeitet wurde.
 3:     if (zuPrüfenderKnoten.wurdeVerarbeitet(zuPrüfendesModul)) then
 4:             // Bestimme nächsten rekursiv zu überprüfender Knoten, ...
 5:             // der in den Verarbeitungsinformationen hinterlegt wurde.
 6:         nächsterZuPrüfenderKnoten = zuPrüfenderKnoten.letzteAusgabe(zuPrüfendesModul)
 7:             // Prüfe alle weiteren Module im gegebenen Abhängigkeitspfad, ...
 8:             // die dem zu prüfenden Modul folgen.
 9:         for all (modul ∈ zuPrüfendesModul.abhängigeModule()) do
10:             warteAufFertigstellung(nächsterZuPrüfenderKnoten,modul)
11:         end for
12:         return
13:     else
14:         warte()    // Warte auf Fertigstellung.
15:     end if
16: end loop
```

Zusätzlich sind Verarbeitungsinformationen dargestellt, die durch [V:z] notiert werden, wobei z ein Modul aus Abb. 3.8.(1) beschreibt, dass die Verarbeitung des Knotens beendet hat. Die Referenzen auf zuletzt durch ein Modul ausgegebene DKG-Knoten werden durch gestrichelte Pfeile visualisiert, die eine Farbe entsprechend der Abhängigkeitspfade in Abb. 3.8.(1) tragen, für die sie relevant sind. Für das Schaubild unwesentliche Daten und Metadaten sind ausgegegraut (bzw. gar nicht dargestellt).

Der zeitliche Verlauf in der Überprüfung der Datenverfügbarkeit für Abb. 3.8.(2) ist wie folgt:

$t_0 \leq t \leq t_3$: Zum Zeitpunkt t_0 wird durch Modul [1] ein Knoten für das Dokument erzeugt (① in Abb. 3.8.(2)). Das Modul [2] verarbeitet diesen und übergibt zum Zeitpunkt t_1 einen DKG-Knoten an nachfolgende Module. Zusätzlich wird dem Modul [6] eine Referenz des selben Knotens zur Überprüfung der indirekten Datenabhängigkeiten übergeben. Das IE-Modul observiert von diesem Zeitpunkt an Veränderungen in den Verarbeitungsinformation des DKG-Knotens (in Abb. 3.8.(2) rötlich hervorgehoben). [3] und [4] verarbeiten den DKG-Knoten parallel. Das Modul [4] erhält zum Zeitpunkt t_2 Eingabedaten und arbeitet fortan ebenfalls nebenläufig. Zum Zeitpunkt t_3 fordert [5] weitere Eingabedaten aus seiner Eingabewarteschlange an, schließt also die Verarbeitung des aktuell durch [6] beobachteten DKG-Knotens ab. Die Verarbeitungsinformationen werden entsprechend aktualisiert, das Modul [6] benachrichtigt und die Prüfung des Abhängikeitspfads [2] $\rightsquigarrow$ [5] $\rightsquigarrow$ [6] abgeschlossen. Es werden keine weiteren Eingabedaten von [5] zur einmaligen Ausführung von [6] benötigt. Die dargestellte Referenz auf den zuletzt durch [5] ausgegebenen Knoten wird daher nicht weiter betrachtet.

$t_3 < t \leq t_5$: Das Modul [6] überwacht nach wie vor den DKG-Knoten (② in Abb. 3.8.(2)) für den Abhängigkeitspfad [2] $\rightsquigarrow$ [3] $\rightsquigarrow$ [4] $\rightsquigarrow$ [6]. Zum Zeitpunkt t_5 beendet auch [3] die Verarbeitung dieses Knotens. Es werden keine weiteren Eingabedaten von [3] zur Ausführung von [6] benötigt. Es ist jedoch die Fertigstellung aller Operationen von [4] für die durch [3] in der Zwischenzeit erzeugten Daten sicherzustellen. Anstatt nun alle Daten zu testen, kann die Überwachung auf den zuletzt durch [3] ausgegebenen DKG-Knoten beschränkt werden (in Abb. 3.8.(2) blau hervorgehoben), da die Infrastruktur sicher stellt, dass genau dieser als letztes durch [4] angefordert wird.

$t_5 < t \leq t_7$: Ab dem Zeitpunkt t_5 wird durch [6] ausschließlich der letzte durch [3] ausgegebene Knoten überwacht (rot markiert). Das Modul [4] beendet die Operationen auf diesem Knoten zum Zeitpunkt t_7. Alle notwendigen Daten zur einmaligen Ausführung der Extraktionslogik von [6] sind damit vorhanden. Die Eingabedaten von [6], die sich in dem

Datenblock befinden, dessen Abhängigkeiten getestet wurden, werden verarbeitet. Die Verarbeitung der anderen Module wird nicht beeinflusst. So denn ein weiterer Datenblock zur Verarbeitung von [6] zur Verfügung steht, kann dieser auf Vollständigkeit geprüft und mit der Verarbeitung direkt fortgefahren werden.

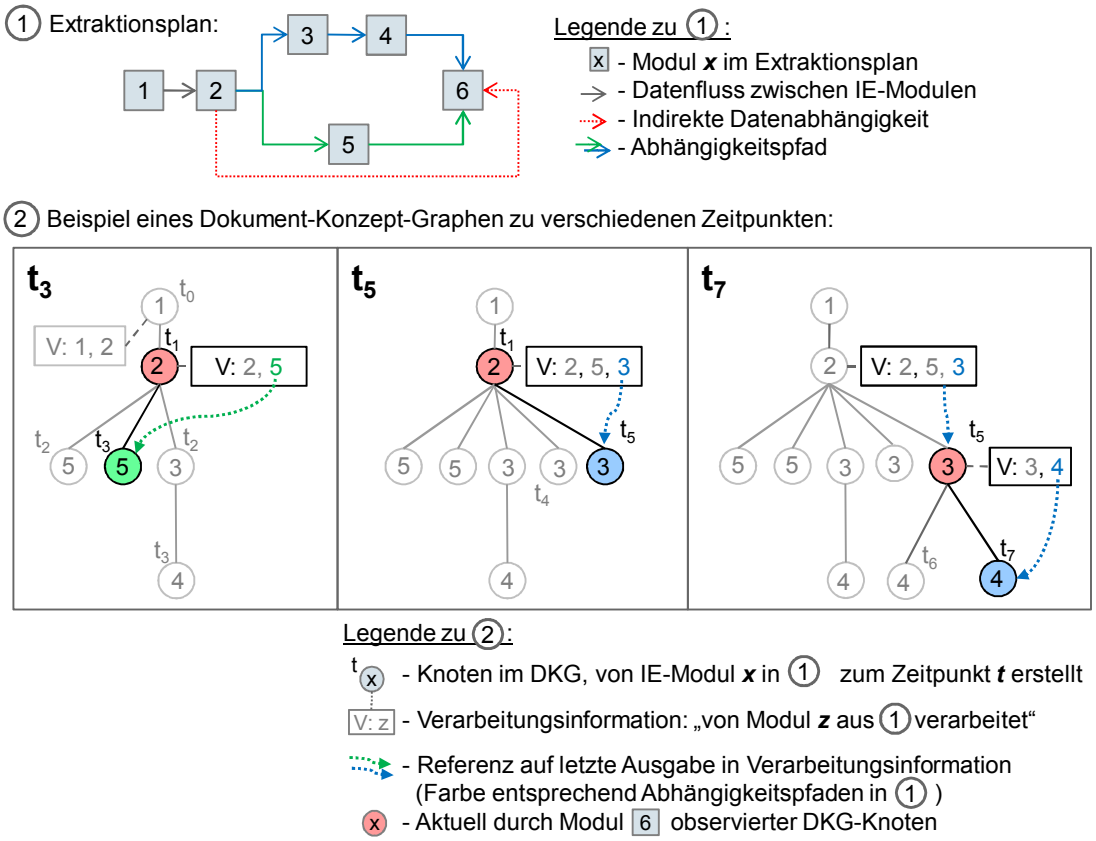


Abb. 3.8: Überprüfung der Datenverfügbarkeit bei indirekten Datenabängigkeiten

Die hier vorgestellten Konzepte unterstützten eine hoch parallele Verarbeitung von Dokumenten auf Basis von durch Warteschlangen entkoppelte IE-Modulen. Die Konsistenz der Ergebnisdaten wird durch den vorgestellten Synchronisationsmechanismus auf effiziente Art und Weise sichergestellt. Der *Overhead*, der durch die Synchronisation entsteht, kann durch die Erweiterung des Laufzeitdatenmodells um Metadaten für Verarbeitungsinformationen und eine effiziente Evaluation von Abhängigkeiten begrenzt werden.

3.2.3 Einflussfaktoren zur Abschätzung der Laufzeitverbesserung

Im Folgenden soll eine theoretische Abschätzung der Laufzeiten von sequentieller und paralleler Ausführung die Einflussfaktoren bezüglich der Beschleunigung durch eine Intra-Dokument-Parallelisierung darstellen. Anhand der Einflussfaktoren kann das Potenzial der Intra-Dokument-Parallelisierung für einen konkreten Anwendungsfall abgeschätzt werden. In der experimentellen Evaluation in Kap. 3.3 untersuchen wir, ob die hier dargestellten Einflussfaktoren so in der Praxis auftreten.

Abschätzung für unbeschränkte Ressourcen.

Die konkrete Konstellation von IE-Modulen in einem anwendungsspezifischen Extraktionsplan sowie die Spezifika der zu verarbeitenden Dokumente beeinflussen die Beschleunigung, die durch eine Intra-Dokument-Parallelisierung erreicht werden kann.

Sei $\mathcal{P} = (M, A)$ ein gerichteter azyklischer Graph, der einen Extraktionsplan gemäß Def. 1.1.2 beschreibt. M sei eine Menge von IE-Modulen (Knoten) und A eine Menge von Abhängigkeiten (Kanten). Das IE-Modul m_0 sei die Datenquelle (Importmodul) und m_k die Datensenke – ein virtuelles Modul am Ende eines Ausführungsplans, das mit allen IE-Modulen verbunden ist, deren Ausgaben nicht von anderen Modulen konsumiert werden. Sei weiterhin $p(m_0, m_k)$ ein zusammenhängender Pfad mit dem gemeinsamen Startknoten m_0 und Endknoten m_k und P die Menge aller solcher Pfade.

Einflussfaktoren der zu verarbeitenden Dokumente und auszuführenden Operationen: Charakteristiken von Dokumenten sowie die Eigenschaften des Extraktionsplans beeinflussen die mögliche Beschleunigung. Sei D eine charakteristische Menge von Dokumenten (oder Dienstaufrufen) für einen gegebenen Anwendungsfall. Definiere weiterhin $n(d, m), d \in D, m \in M$ die Anzahl der Eingabedaten, die bei der Analyse eines Dokuments durch ein Modul m verarbeitet werden müssen, so kann die durchschnittliche Anzahl von Eingabedaten eines Moduls pro Dokument durch $\bar{n}(m) = 1/|D| \sum_{d \in D} n(d, m)$ bestimmt werden. Anzumerken ist, dass im Falle von Aggregationsmodulen mit „ein Eingabedatum“ ein Datenblock von abhängigen Eingabedaten gemeint ist.

Sei $t(d, m)$ die Zeit, die ein Modul bei sequentieller Verarbeitung aller Eingabedaten im Kontext eines Dokuments benötigt, so kann zur Abschätzung der Laufzeiten mit einer Intra-Dokument-Parallelisierung die durchschnittliche Verarbeitungszeit für ein einzelnes Eingabedatum durch $\bar{t}(m) = 1/|D| \sum_{d \in D} t(d, m)/n(d, m)$ beschrieben werden. Für ein Importmodul m_0 sei $\bar{n}(m_0) = 1$ und $\bar{t}(m_0)$ die durchschnittliche Zeit, die für den Dokumentzugriff und das Abbilden des Dokuments in das Datenmodell notwendig ist.

Verarbeitungszeiten bei sequentieller Ausführung: Die Laufzeit $T_{\max}(\mathcal{P})$ eines IE-Systems für ein Dokument bei strikt sequentieller Ausführung eines Extraktionsplans $\mathcal{P}$, lässt sich durch Aufsummieren des Produkts aus $\bar{n}(m), m \in M$ und $\bar{t}(m), m \in M$ wie folgt approximieren:

$$T_{\max}(\mathcal{P}) = \sum_{m \in M} \bar{t}(m) \cdot \bar{n}(m). \tag{3.1}$$

Verarbeitungszeiten bei paralleler Ausführung: In der in Kap. 3.2.2 vorgestellten Art der Parallelisierung bestimmt vor allem ein Pfad $p_{\max}(m_0, m_k) \in P$, der so genannte *Kritische Pfad* [KA96], die Laufzeit. Der Kritische Pfad ist dadurch gekennzeichnet, dass die Summe der durchschnittlichen Zeiten für Einzeloperationen $\bar{t}(m)$ über die Module im Pfad $m \in p_{\max}(m_0, m_k)$ im Vergleich zu anderen Pfaden maximal ist. Da alle von $p_{\max}(m_0, m_k)$ unabhängigen Einzeloperationen durch die Aufgaben- und Datenparallelität nebenläufig ausgeführt werden, können die Pfade $P \backslash p_{\max}(m_0, m_k)$ und die Anzahl der Einzeloperationen eines Moduls $\bar{n}(m)$, unter der Annahme unendlicher Ressourcen (Prozessoren), vernachlässigt werden. Wir diskutieren diesen Zusammenhang noch anhand eines Beispiels.

Die Laufzeit $T_{\min}(\mathcal{P})$ eines parallelisierten IE-Systems für ein durchschnittliches Dokument berücksichtigt demnach nur die Module in $p_{\max}(m_0, m_k)$ und ist (theoretisch) unabhängig von der Anzahl der Einzeloperationen $\bar{n}(m)$:

$$T_{\min}(\mathcal{P}) = \sum_{m \in p_{\max}(m_0, m_k)} \bar{t}(m). \tag{3.2}$$

Das Potenzial zur Verbesserung der Laufzeiten ist demnach insbesondere in Anwendungsfällen mit komplexen Extraktionsplänen und Dokumenten, die viele Einzeloperationen zur Extraktion von Entitäten erfordern, enorm.

Beispiel zu den Einflussfaktoren. Das Beispiel in Abb. 3.9 soll die Abhängigkeit der Laufzeit vom *Krititschen Pfad* und die Unabhängigkeit von der Anzahl der Einzeloperationen unter der Annahme von unendlichen Ressourcen illustrieren. Gegeben sei der Extraktionsplan in Abb. 3.9.(1). In Abb. 3.9.(2) ist ein Beispiel-*Workload* dargestellt. Kreise in Abb. 3.9.(2) illustrieren die Ausgabe eines DKG-Knotens und Pfeile Einzeloperationen zur Verarbeitung eines Eingabedatums. Auch die parallele Verarbeitung durch mehrere Instanzen eines IE-Moduls ist beispielhaft für die Module [3] und [5] durch parallele Pfeile illustriert. Gestrichelte Linien repräsentieren eine Datenübernahme durch Nachfolgemodule. Anzumerken ist, dass die Ausgabezeitpunkte von Daten nicht von der Gesamtlaufzeit eines Moduls abhängig sind, wie z. B. für Modul [2] in Abb. 3.9.(2) zu sehen ist.

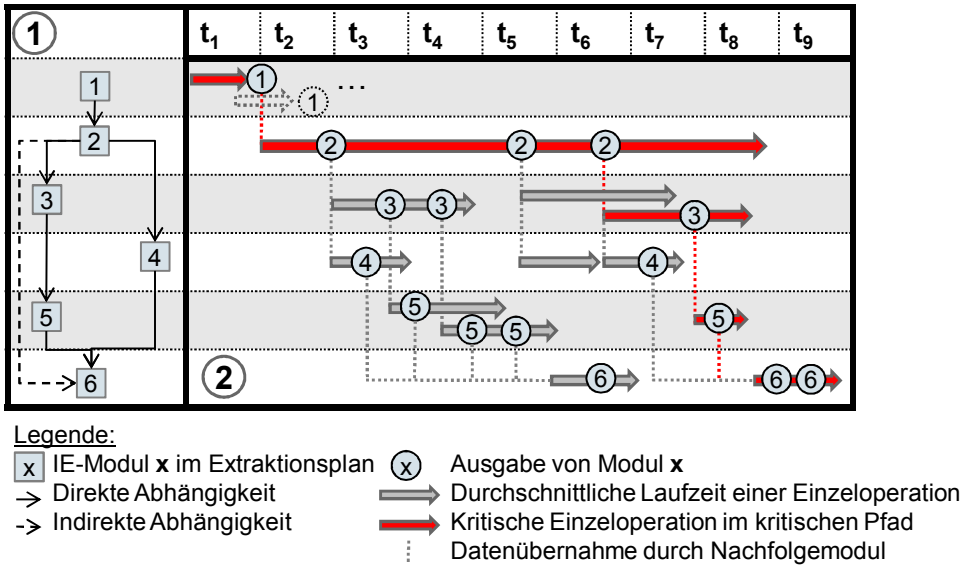


Abb. 3.9: Beispiel zum Laufzeitverhalten bei einer parallelen Ausführung

Das Beispiel in Abb. 3.9 illustriert das oben beschriebene Verhalten, dass es jeweils eine kritische Einzeloperation für Module des Kritischen Pfades gibt und andere Einzeloperationen parallel dazu ausgeführt werden. Ausgaben von IE-Modulen werden, wenn keine Synchronisation notwendig ist, durch nachfolgende Module unmittelbar, ohne zeitliche Verzögerung weiterverarbeitet. Das heißt, dass einzelne Arbeitszyklen im optimalen Fall beendet werden können, bevor das vorgelagerte Modul die zugehörige Operation abschließt. Im Beispiel-*Workload* würden z. B. die Laufzeiten der Operationen von [3] und [5] die Gesamtlaufzeit nicht beeinflussen, obwohl sie im *Krititschen Pfad* sind, da im Beispiel die letzte Ausgabe von Modul [2] zu einem günstigen Zeitpunkt erfolgt. Demnach definiert $T_{\min}(\mathcal{P})$ sogar die Obergrenze der Dokumentverarbeitungszeit, wenn jeglicher *Overhead* vernachlässigt und unendliche Ressourcen angenommen werden.

Abschätzung für beschränkte Ressourcen.

Im Allgemeinen lässt sich die Beschleunigung eines parallelisierten Programms unter Betrachtung der zur Verfügung stehenden Ressourcen nach dem Amdahlschen Gesetz [Amd67] mit Hilfe des Verhältnisses der parallelisierbaren und nicht parallelisierbaren Programmanteile abschätzen. Die Beschleunigung gibt das Verhältnis der sequentiellen Laufzeit $T_{\max}$ (siehe Gleichung 3.1) zur durch Parallelverarbeitung erreichten Laufzeit an. Die Laufzeit nicht parallelisierbarer Programmteile werden durch $T_{\min}$ (siehe Gleichung 3.2) und die für parallelisierbare Programmteile durch die Differenz $T_{\max} - T_{\min}$ approximiert.

Sei N die Anzahl der eingesetzten Prozessoren und $\text{ø}(N)$ der *Overhead* der durch eine Parallelisierung entsteht, so kann man den Faktor der Laufzeitbeschleunigung $B(\mathcal{P})$ für einen Extraktionsplan $\mathcal{P}$ wie folgt approximieren:

$$B(\mathcal{P}) = \frac{T_{\max}}{T_{\min}(\mathcal{P}) + \text{ø}(N) + \frac{(T_{\max} - T_{\min})}{N}}. \tag{3.3}$$

Die Abschätzung der Beschleunigung durch Gleichung 3.3 illustriert folgende Zusammenhänge: Zum einen wird dargestellt, dass die nie zu erreichende Obergrenze der Beschleunigung bei $B(\mathcal{P})_{max} = T_{\max}/T_{\min}$ liegt. Der parallelisierte Anteil $(T_{\max} - T_{\min})/N$ bestimmt mit steigendem N immer weniger den Beschleunigungsfaktor, wogegen der seqentielle Anteil $T_{\min}$ immer mehr an Gewicht gewinnt. Es ist anzumerken, dass auch eine Obergrenze für die Parallelisierung existiert, die durch die Einzeloperationen und deren Abhängigkeiten festgelegt ist.

Wenn ein Extraktionsplan aus einer Sequenz von Modulen, die jeweils eine Einzeloperation ausführen, besteht, ergibt sich unabhängig von N eine Beschleunigung von $B(\mathcal{P})_{max} = 1$. Wenn jedoch ein relativ „flacher“ und „breiter“ Extraktionsplan vorliegt, ist die Steigerung der Beschleunigung mit steigendem N hoch, da der parallelisierbare Anteil $(T_{\max} - T_{\min})$ hoch, aber der sequentielle Anteil $T_{\min}$ klein ist.

Der *Overhead* $\text{ø}(N)$ ist gemäß dem Amdahlsches Gesetz eine monoton steigende Funktion von der Anzahl der eingesetzten Ressourcen (genauer dem Grad der Parallelisierung). Während der Einfluss von $(T_{\max} - T_{\min})/N$ auf die Beschleunigung mit steigendem N abnimmt, nimmt der Einfluss des *Overheads* $\text{ø}(N)$ zu. Einflussfaktoren für den *Overhead* sind neben der Anzahl der Prozessoren: die Struktur des Extraktionsplans, also die Charakteristiken der IE-Module und deren Konstellation untereinander, und die konkreten Spezifika der verarbeiteten Dokumente, wie z. B. die Größe der Dokumente, die die Anzahl der Einzeloperationen bestimmt, die Zeitpunkte, zu denen Ausgabeknoten zur Verfügung stehen etc. Der *Overhead* kann daher nicht ohne weiteres a priori bestimmt werden.

3.2.4 Zusammenfassung

In Kap. 3.2 wurden Konzepte zur Intra-Dokument-Parallelisierung von Extraktionsmodulen vorgestellt, die die Ausführungszeiten von IE-Systemen verbessert, ohne dass die konkrete Implementierung und Charakteristiken der IE-Module für die Ausführungsplattform bekannt sein müssen. Basierend auf der Komposition von IE-Modulen in einem Extraktionsplan können verschiedenste Abhängigkeiten identifiziert und daraus folgend unabhängige Operationen parallel ausgeführt werden.

Die asynchrone, nebenläufige Ausführung von Modulen wird durch eine lose Kopplung von Modulen über Warteschlangen erreicht. Zusätzlich können mehrere Modulinstanzen zur parallelen Verarbeitung von Zwischenergebnissen derselben Warteschlange eingesetzt werden. Eine Erweiterung des Laufzeitdatenmodells ermöglicht die Überprüfung von indirekten Datenabhängigkeiten und damit eine Synchronisation und nebenläufige Ausführung von Aggregationsmodulen.

Eine Abschätzung der Laufzeitverbesserung ergab, dass unter der Annahme von unbeschränkten Ressourcen bei komplexen Ausführungsplänen eine enorme Beschleunigung zu erwarten ist. Lediglich eine Sequenz von kritischen Einzeloperationen bestimmt die Laufzeit. Die Anzahl der notwendigen Einzeloperationen zur Verarbeitung von Eingabedaten eines Moduls und die Ausführung von IE-Modulen außerhalb des *Kritischen Pfades* können theoretisch vernachlässigt werden.

Eine Abschätzung hinsichtlich der eingesetzten Ressourcen ergab, dass die Gesamtlaufzeiten jedoch nicht linear mit der Anzahl der eingesetzten Prozessoren skalieren können. Insbesondere erzeugt die Synchronisation von Modulen und konkurrierende Schreibzugriffe *Overhead*. Der *Overhead* hängt von einer Vielzahl von Faktoren hinsichtlich der verwendeten IE-Module, deren Konstellation und von Dokumentcharakteristiken ab und kann daher nicht a priori bestimmt werden. Eine experimentelle Evaluation von Laufzeitverbesserungen, die in der Praxis erreicht werden können, ist daher geboten.

3.3 Experimentelle Evaluation

Im Folgenden werden die vorgestellten Konzepte zur Laufzeitverbesserung anhand ausgewählter Experimente evaluiert. Wir vergleichen das Laufzeitverhalten einer sequentiellen Ausführung mit *UIMA* 2.2.2 [FL04] zur parallelisierten Verarbeitung auf Basis der vorgestellten Konzepte. Das Ziel ist es, die in der Praxis erreichbaren Laufzeitvorteile zu bestimmen und die Abschätzung des Verbesserungspotenziales aus Kap. 3.2.3 zu untermauern.

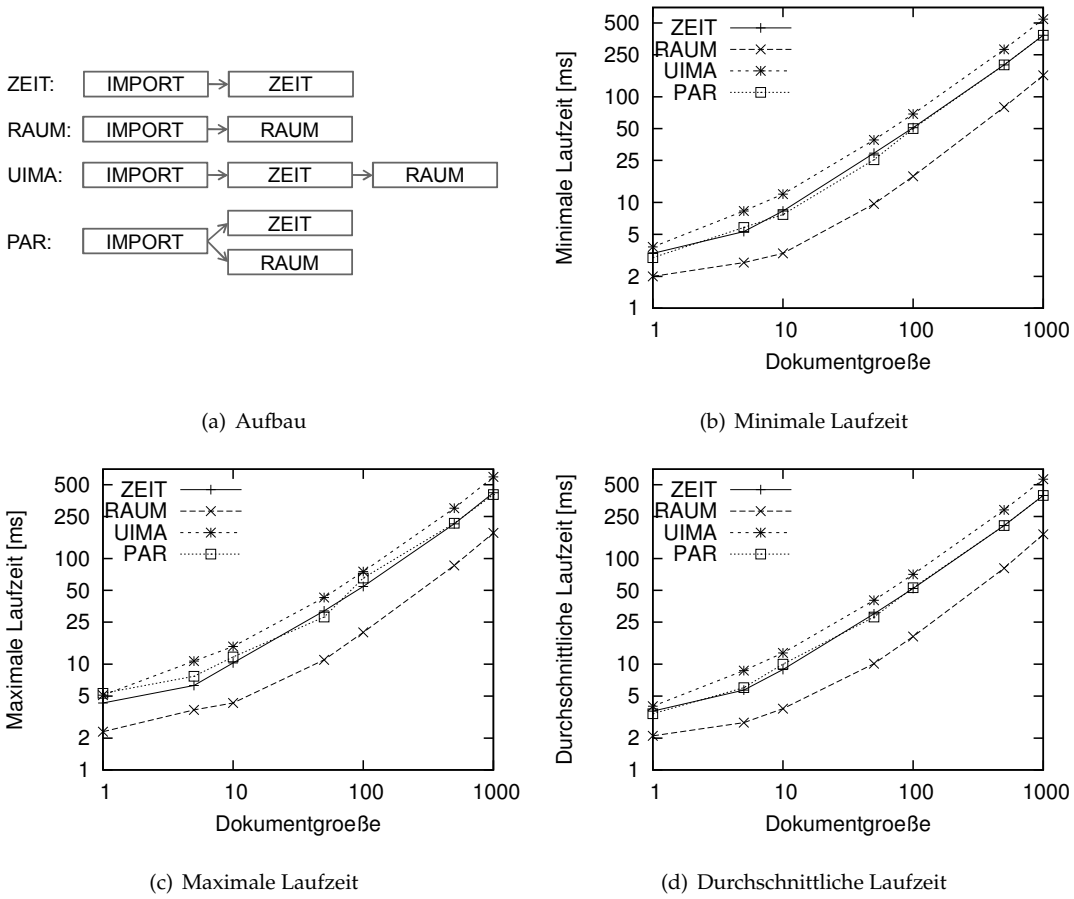
Die Laufzeitverbesserung wird für die in dieser Arbeit untersuchten Situationen evaluiert: die einfache Aufgabenparallelität sowie die Datenparallelität im Kontext direkter und indirekter Datenabhängigkeiten. Zur Illustration des *Overheads*, der durch die Parallelisierung entsteht, wurden Experimente mit variierender Dokumentgröße durchgeführt, die hier die Anzahl der notwendigen Einzeloperationen und damit den möglichen Grad der Parallelisierung bestimmen. Es ist anzumerken, dass nicht die Laufzeitcharakteristiken der einzelnen IE-Module im Vordergrund stehen, die Anwendungs- und Implementations-abhängig sind, sondern die Laufzeiten, die bei der Interaktion der Module erzielt werden. Auf eine Zusammenführung von Operationen oder Optimierung der Ausführungsreihenfolge wird daher verzichtet (siehe Kap. 3.4).

RapidUIM wurde mit Java 6.0[1] implementiert. Die Parallelisierung basiert auf der *Multi-Threading API* von Java. Die Experimente wurden auf einem Desktoprechner mit einem *Intel CoreDuo Quad Q9400* Prozessor mit vier 2,67GHz Kernen und 4GB RAM vorgenommen. Aufgrund der limitierten Ressourcen des Testsystems wurden die Experimente anhand relativ einfacher Extraktionspläne durchgeführt. Die Ergebnisse lassen sich jedoch ohne weiteres auf komplexere Extraktionspläne, die auf Ressourcen-reicherer *Hardware* ausgeführt werden, übertragen.

3.3.1 Aufgabenunabhängigkeit

Das Ziel der Evaluation für eine (einfache) Parallelisierung von Aufgaben ist es, die Annahme aus Kap. 3.2.2 zu untermauern, dass die Gesamtverarbeitungszeit exakt der Laufzeit des kostenintensivsten (parallel ausgeführten) IE-Moduls entspricht. Wir betrachten hier zwei IE-Module, die den gesamten Dokumenttext parallel scannen. Die Experimente basieren auf einem IE-Programm und einem Beispieldokument, die mit *UIMA*[2] ausgeliefert werden. *UIMA*-Module wurden so modifiziert, dass sie nicht blockieren und Ergebnisse unmittelbar weiter propagieren, ohne die eigentliche Extraktionslogik zu verändern. Eine solche Modifikation ist in den meisten Fällen ohne weiteres möglich.

[1] `http://java.sun.com/javase/6/`, Stand: 10.01.2010
[2] `http://incubator.apache.org/uima/downloads.cgi`, Stand: 01.05.2010



(a) Aufbau

(b) Minimale Laufzeit

(c) Maximale Laufzeit

(d) Durchschnittliche Laufzeit

Abb. 3.10: Laufzeit bei Aufgabenunabhängigkeit

Der Extraktionsplan für *RapidUIM* (siehe *„PAR“* in Abb. 3.10(a)) umfasst ein Importmodul, das ein Textfragment je nach Konfiguration des Experiments dupliziert und in das Datenmodell der Plattform abbildet, so dass das Laufzeitverhalten für eine unterschiedliche Anzahl von Einzeloperationen simuliert werden kann. Das verwendete Textfragment für dieses Experiment ist im Anhang auf S. 145 in Bsp. A.1 zu finden.

Der Extraktionsplan *„PAR“* in Abb. 3.10(a) enthält weiterhin ein IE-Modul zur Extraktion von Datums- und Zeitangaben, das fünf Reguläre Ausdrücke für verschiedene Zeit- und Datumsformate auf den Text anwendet, und ein Modul, das zwei Reguläre Ausdrücke zur Extraktion von Raumnummern verwendet. Da kein Modul zur weiteren Dokumentsegmentierung im Extraktionsplan enthalten ist, werden beide IE-Module (parallel) auf den gesamten Dokumenttext angewendet. Eine nebenläufige Verarbeitung von Zwischenresultaten auf Basis einer Datenparallelität findet also nicht statt.

Jedes IE-Modul in *RapidUIM* arbeitet nach der Instantiierung autonom. Für jedes Eingabe- und Ausgabedatum des Zeit- und Raummoduls wird daher ein Zeitstempel erfasst, so dass die Laufzeit der Extraktionsmodule in der parallelen Verarbeitung exakt bemessen werden kann. Relevant für die Evaluation sind der Zeitstempel der ersten Eingabe und der Zeitstempel der letzten Ausgabe der Module *„RAUM“* und *„ZEIT“*. Die in Abb. 3.10 dargestellten Laufzeiten entsprechen der Zeitdifferenz zwischen der frühesten Dateneingabe und der spätesten Ausgabe von Ergebnissen. Die Laufzeit des Importmoduls zur Duplikation des Textes wird nicht berücksichtigt.

In Abb. 3.10 sind zum einen die Laufzeiten der einzelnen IE-Module („*RAUM*“ und „*ZEIT*“) für die Verarbeitung des Dokumenttexts dargestellt. Zum anderen werden die Gesamtlaufzeiten für die sequentielle Ausführung („*UIMA*“) und parallele Ausführung („PAR“) beider Module gezeigt. In Abb. 3.10(b) sehen wir jeweils die minimalen, in Abb. 3.10(d) die maximalen und in Abb. 3.10(d) die durchschnittlichen Laufzeiten für 100 Durchläufe pro Dokumentgröße.

Wir stellen fest, dass die Verarbeitungszeiten für „*ZEIT*“ ca. um einen Faktor $1,5$ größer sind als die von „*RAUM*“, was auf die unterschiedliche Anzahl von ausgewerteten Regulären Ausdrücken zurückzuführen ist. Die Laufzeiten von „*UIMA*“ entsprechen exakt der Summe von „RAUM“ und „ZEIT“. Wir sehen, dass die Laufzeit von „*PAR*“ bei allen in Abb. 3.10 dargestellten Messwerten ungefähr der Laufzeit des kostenintensivsten IE-Moduls entsprechen. Der *Overhead* durch den parallelen lesenden Zugriff auf Eingabedaten (hier auf das gesamte Dokument) ist dementsprechend als unkritisch zu betrachten.

Die Gesamtlaufzeit wird hier lediglich, da keine Datenparallelitäten berücksichtigt werden können, im Vergleich zu „*UIMA*“ im Durchschnitt um den Faktor $1,5$ verbessert, wobei eine maximale Verbesserung um den Faktor 2 für dieses einfache Szenario möglich gewesen wäre. IE-Systeme für Anwendungsfälle in der *Enterprise Search* enthalten jedoch in der Regel viele parallele Extraktionsmodule und werden oft auf Resourcen-reichen Systemen ausgeführt. Das Konzept der Parallelisierung von unabhängigen Extraktionsaufgaben verspricht daher ein großes Potenzial die Laufzeiten von solchen IE-Systemen zu verbessern.

3.3.2 Direkte Datenabhängigkeit

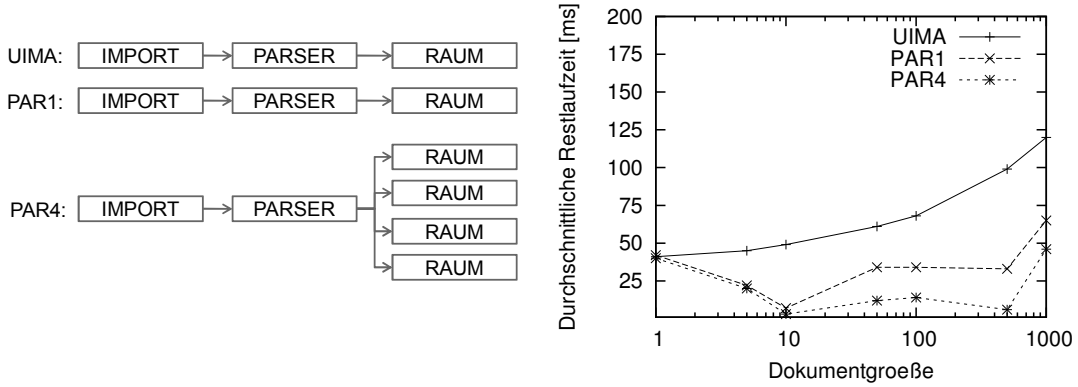
Zwischen IE-Modulen, die in einem Extraktionsplan direkt miteinander verbunden sind, besteht eine direkte Datenabhängigkeit. Das heißt, dass Daten des Vorgängermoduls erst verarbeitet werden können, wenn sie zur Verfügung stehen.

In der Abschätzung der Verarbeitungszeiten in Kap. 3.2.3 wurde angenommen, dass bei unbegrenzten Ressourcen ein Ausgabedatum unmittelbar verarbeitet werden kann, so dass in der Gesamtlaufzeit quasi die Anzahl der notwendigen Einzeloperationen vernachlässigt werden kann. Das heißt, dass sich die Gesamtlaufzeit im optimalen Fall aus der Laufzeit einer Einzeloperation von Modulen im *Kritischen Pfad* ergibt.

Da bei der Verwendung von Warteschlangen eine gewisse Latenzzeit besteht, bis die Daten für das Nachfolgemodul zur Verfügung stehen, viele Einzeloperationen in der Informationsextraktion im Millisekundenbereich liegen und nur begrenzte Ressourcen zur Verfügung stehen, ist die oben genannte Annahme nicht in allen Anwendungsfällen zutreffend. Im ungünstigsten Fall sind die Laufzeiten von zwei verbundenen Modulen derart unterschiedlich, dass ein Vorgängermodul quasi alle Daten verarbeitet, bevor das Nachfolgemodul ein entsprechendes Eingabedatum fertigstellen kann. Wir zeigen im Folgenden zwei Experimente, in denen zum einen die Vorgängermodule eine wesentlich höhere Laufzeit haben, als Nachfolgemodule und zum anderen einen Fall, in dem die Nachfolgemodule wesentlich höhere Laufzeiten haben.

Im ersten Szenario zeigt eine Analyse von Laufzeitdifferenzen zwischen vor- und nachgelagerten IE-Modulen, dass gerade bei aufwändigen Dokumentvorverarbeitungsoperationen der Ansatz der nebenläufigen Verarbeitung von Zwischenergebnissen eine Verbesserung verspricht. Das zweite Szenario zeigt die Anwendbarkeit des Ansatzes bei einer direkten Datenabhängigkeit zwischen drei IE-Modulen, wobei die Laufzeiten von nachgelagerten Modulen die der vorgelagerten um ein Vielfaches übertreffen und mehrere Modulinstanzen zum Ausgleich dieser Differenzen verwendet werden.

Szenario 1. Im ersten Szenario wird als Dokument ein HTML-Fragment verwendet, welches dupliziert wird, um variierende Dokumentgrößen zu simulieren (siehe Abb. A.2 in An-



(a) Aufbau (b) Restlaufzeit Raummodul

Abb. 3.11: Restlaufzeit des Raummoduls relativ zum Parsermodul

hang A.1). Ein XML-Parser[3] wird dazu genutzt, einzelne Teilfragmente (Tabellenzellen) aus dem HTML-Dokument zu extrahieren. Die Teilfragmente dienen als Eingabe für das oben erläuterte IE-Modul zur Extraktion von Raumnummern (siehe auch experimenteller Aufbau in Abb. 3.11(a)). Die Laufzeit des Parser ist wesentlich höher als die des Nachfolgemoduls. Wir zeigen Experimente für *RapidUIM* mit einer Instanz des Raummoduls („*PAR1*“ in Abb. 3.11(a)) und vier Instanzen („*PAR4*“ in Abb. 3.11(a)), die sich eine Eingabewarteschlange teilen. Zum Vergleich dient wiederum die Ausführung der programmatisch kombinierten IE-Module in *UIMA* („*UIMA*“ in Abb. 3.11(a)).

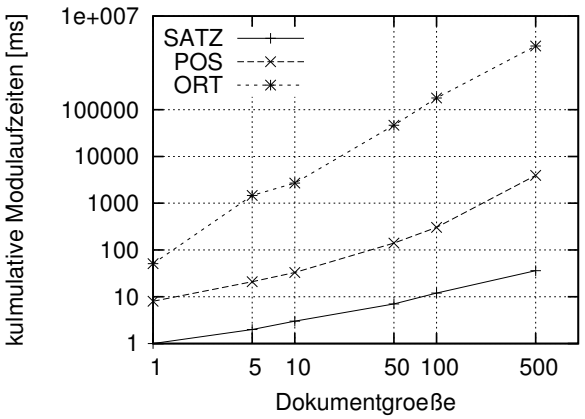
Gemessen wird der Startzeitpunkt des Parser (erstes Eingabedatum) und die Endzeit der Modulinstanzen zur Extraktion der Räume (letztes Ausgabedatum). Zusätzlich wurde die Laufzeit des XML-Parsers ohne nachgelagerte IE-Module gemessen. In Abb. 3.11 sind die Restlaufzeiten des IE-Moduls zur Extraktion von Räumen, also die Differenz zwischen durchschnittlicher Gesamtlaufzeit und durchschnittlicher Laufzeit des XML-Parsers dargestellt.

Nach der Abschätzung der Laufzeiten in Kap. 3.2.3 sollten die Restlaufzeiten für die nebenläufige Verarbeitung wenige Millisekunden betragen, da die Zeitdauer zur Anwendung eines Regulären Ausdrucks auf ein kurzes Textfragment äußerst gering ist. Eine Restlaufzeit von wenigen Millisekunden kann tatsächlich bei einer Dokumentgröße von 10 erreicht werden. Für die Messungen zu Dokumentgrößen < 10 ist zu sehen, dass durch „*PAR1*“ und „*PAR4*“ keine bzw. geringere Verbesserungen im Vergleich zur sequentiellen Verarbeitung mit „*UIMA*“ erreicht werden, da die Latenzzeit der Ergebnisbereitstellung zu groß ist.

Die Restlaufzeiten von „*UIMA*“ steigen streng monoton mit steigender Dokumentgröße. Die Restlaufzeiten für „*PAR4*“ bleiben annähernd konstant bis zu einer Dokumentgröße von 500, wogegen die Restlaufzeiten für „*PAR1*“ leicht ansteigen. Für eine Dokumentgröße von 1000 steigen die Restlaufzeiten für „*PAR1*“ und „*PAR4*“, was darauf zurückzuführen ist, dass in den Warteschlangen der Module eine Art „Stau“ entsteht, der durch die verfügbaren Ressourcen nicht ausgeglichen werden kann. Es ist jedoch anzumerken, dass die Restlaufzeit von „*PAR4*“ bei Dokumentgröße 1000 annähernd der Restlaufzeit von „*UIMA*“ bei einer Dokumentgröße von 1 entspricht. Die Gesamtlaufzeit wird für *PAR4* im Durchschnitt um den Faktor $3,1$ beschleunigt.

Das Experiment zeigt, dass die Annahme, dass die Gesamtverarbeitungszeit aus den Einzeloperationen zur Verarbeitung eines Eingabedatums bestimmt werden kann, in der Praxis in bestimmten Anwendungsfällen zutreffend ist. Insbesondere in Anwendungsszenarien, in denen vorgelagerte Module aufwändige Operationen ausführen, können Zwischenergebnisse durch nachfolgende Module unmittelbar nebenläufig verarbeitet werden.

[3] `http://java.sun.com/j2se/1.4.2/docs/api/javax/xml/parsers/SAXParser.html`, Stand: 10.01.2010



(a) Kumulative Modullaufzeiten

(b) Aufbau

Abb. 3.12: Szenario 2: Kumulative Modullaufzeiten bei Ausführung mit UIMA

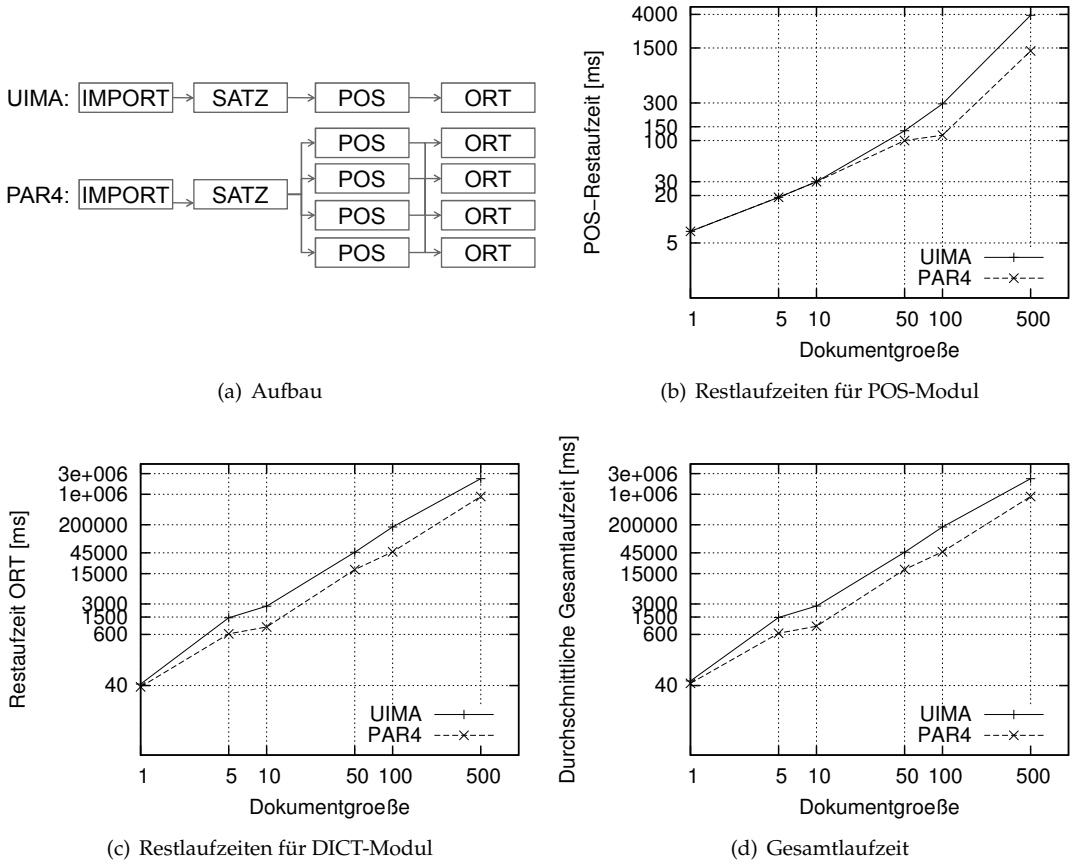
Szenario 2. Es gibt Extraktionspläne, in denen die Kürze der Einzeloperationen vorgelagerter IE-Module eine nebenläufige Verarbeitung von Zwischenergebnissen durch aufwändigere nachgelagerte Operationen fast unmöglich macht, so dass quasi immer ein „Stau“ in den Warteschlangen entsteht. Ein zweites Szenario mit direkter Datenabhängigkeit zwischen drei Modulen soll die Laufzeitverbesserungen in einem solchen Anwendungsfall zeigen. Das Textfragment, das diesem Experiment zugrunde liegt und dupliziert wird, um eine variierende Anzahl von Einzeloperationen zu simulieren, ist im Anhang A.1 in Bsp. A.3 zu finden.

Das Ziel des hier untersuchten IE-Systems ist es, Ortsbezeichnungen aus einem Dokument auf Basis eines approximativen Wörterbuchabgleichs mit ca. 300.000 Ortsbezeichnungen zu extrahieren. Zur Verbesserung der Extraktionsqualität wird eine morphologische Analyse zum Filtern von Substantiven verwendet. Grundlage der morphologischen Analyse sind Sätze. Der IE-Plan besteht daher, wie in Abb. 3.12(b) zu sehen ist, aus Import (*„IMPORT“*), Satzerkennung (*„SATZ“*), morphologischer Analyse (*„POS“*) und fehlertoleranten Abgleich von Substantivgruppen mit dem Wörterbuch (*„ORT“*). Die Module zur Satzerkennung und morphologischer Analyse wurden basierend auf der *OpenNLP Tools API v1.4.0*[4] und der Wörterbuchabgleich mit *Alias-i LingPipe 3.9.0*[5] implementiert. Der Abgleich erfolgt mit einer *Levenshtein-Distanz* von 2.

Die kumulierten Laufzeiten der einzelnen IE-Module in *UIMA* sind in Abb. 3.11 zu sehen. Die Laufzeiten der einzelnen Module unterscheiden sich extrem. Die morphologische Analyse (*„POS“*) benötigt eine vielfache Laufzeit im Vergleich zur Satzerkennung (*„SATZ“*) und der Wörterbuchabgleich (*„ORT“*) ein vielfaches der Laufzeit von *„POS“*, so dass eher die parallele Verarbeitung von Zwischenergebnissen mit mehreren Modulinstanzen und nicht die nebenläufige Ausführung zum Vorgängermodul eine Laufzeitverbesserung verspricht. Wir verwenden für die Experimente mit *RapidUIM* jeweils 4 Modulinstanzen für *„POS“* und 4 Modulinstanzen für *„ORT“* (siehe *„PAR4“* in Abb. 3.13(a)) .

Für die Ausführung mit *UIMA* und *RapidUIM* wurden für jedes Modul Start- und Endzeit gemessen. In Abb. 3.13 sind die Ergebnisse der Laufzeitmessungen dargestellt. In Abb. 3.13(b) sehen wir die Restlaufzeit der morphologischen Analyse (*„POS“*) gegenüber der Satzerkennung (*„SATZ“*). Bis zu einer Dokumentgröße von 10 Sätzen zeigen sich keine Laufzeitdifferenzen, da die Latenzzeiten der Datenbereitstellung im Vergleich zur Gesamtlaufzeit enorm ist. Ab einer

[4] `http://opennlp.sourceforge.net/`, Stand 10.01.2010
[5] `http://alias-i.com/lingpipe/`, Stand 10.01.2010



(a) Aufbau

(b) Restlaufzeiten für POS-Modul

(c) Restlaufzeiten für DICT-Modul

(d) Gesamtlaufzeit

Abb. 3.13: Szenario 2: Laufzeitverbesserung bei direkter Datenabhängigkeit

Dokumentgröße von 10 Sätzen verbessert die Datenparallelität die Restlaufzeiten, die bei der Dokumentgröße von 500 Sätzen einer Verbesserung um den Faktor 3 entspricht.

Der Verarbeitungszeitgewinn bei der morphologischen Analyse verspricht jedoch bei der Betrachtung der Modullaufzeiten für den Wörterbuchabgleich (siehe Abb. 3.12) lediglich eine geringe Verbesserung der Gesamtlaufzeit. In Abb. 3.13(c) ist die Restlaufzeit von *„ORT“* gegenüber *„POS“* und in Abb. 3.13(c) die Gesamtlaufzeit des Extraktionsplans dargestellt. Es ist zu sehen, dass die Einflüsse von den, dem Wörterbuch-basierten Modul vorgelagerten Modulen, auf die Gesamtlaufzeiten marginal sind. Die Beschleunigung der Restlaufzeiten von *„ORT“* liegt zwischen dem Faktor $1,5$ bei einer Dokumentgröße von 1 und $3,8$ bei einer Dokumentgröße von 100. Die Beschleunigung der Gesamtlaufzeit entspricht ungefähr dieser Laufzeitverbesserung, da die Laufzeiten anderer Module im Verhältnis gesehen extrem gering sind, und liegt entsprechend zwischen $1,5$ und $3,8$.

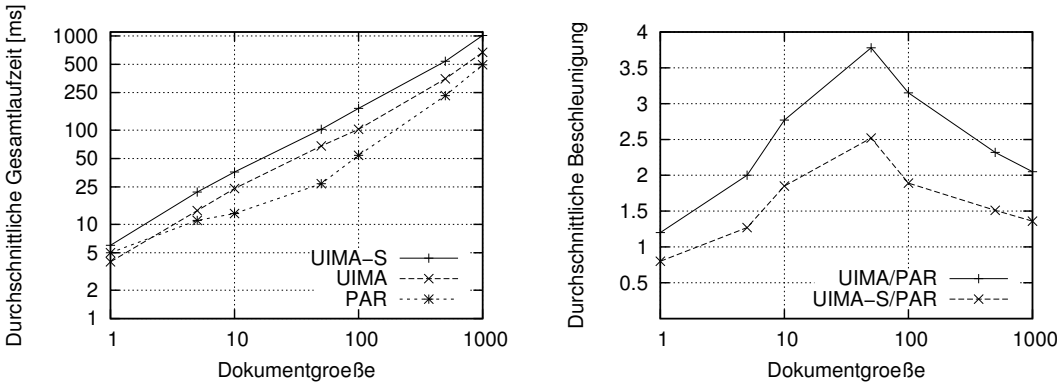
3.3.3 Indirekte Datenabhängigkeit

Für die Messung der Laufzeitverbesserungen bei indirekter Datenabhängigkeit erweitern wir den in Kap. 3.3.1 erläuterten Extraktionsplan zum einen um ein Modul zur Vorsegmentation von Texten in Absätze und zum anderen um ein Modul zur Extraktion von Treffen. Die nebenläufige Extraktion von Treffen ist indirekt von der Fertigstellung aller Operationen für einen Absatz abhängig. Es handelt sich dabei um einen vereinfachten Extraktionsplan im Vergleich zum in Kap. 3.2 diskutierten, der eher den Ressourcen des hier verwendeten Systems entspricht.

Der Extraktionsplan für *RapidUIM* (*„PAR“* in Abb. 3.14(c)) importiert das selbe Textfragment, wie der Plan, der in Kap. 3.2 untersucht wurde, und dupliziert es entsprechend der Konfiguration des Experiments. Das Dokument wir anschließend in Absätze auf Basis eines Regulären Ausdrucks segmentiert– Resultat sind also die ursprünglichen Textfragmente. Absätze werden durch jeweils zwei Modulinstanzen für *„ZEIT“* und *„RAUM“*, wie in Kap. 3.3.1 diskutiert, analysiert. Die Ergebnisse werden auf Basis von gemeinsamen Vorkommen in einem Absatz zu Treffen (*„TREFFEN“* in Abb. 3.14(c)) aggregiert.

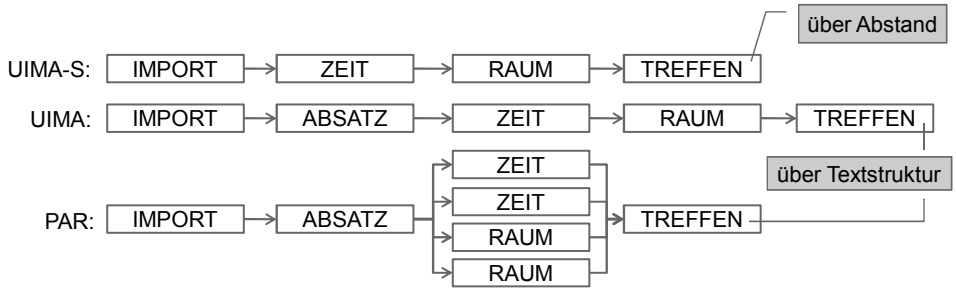
Die *UIMA*-Implementation (*„UIMA“* in Abb. 3.14(c)) ist äquivalent zu *„PAR“*, abgesehen davon, dass die Ausführung nicht parallelisiert erfolgt. Anzumerken ist, dass im Falle von *UIMA*, das *„TREFFEN“*-Modul speziell auf Anwendungsfall angepasst werden musste, so dass die Ergebnisse für *„ABSATZ“*, *„ZEIT“* und *„RAUM“* im Datencontainer adressiert und auf Basis der Postionsindizes im Text einander zugewiesen wurden.

Als weiteren Vergleich zeigen wir die mit *UIMA* ausgelieferte Standardkonfiguration des Extraktionsplans, der lediglich den Abstand von Raum und Zeit und nicht die Textstrukur zur Extraktion von Treffen berücksichtigt (*„UIMA-S“* in Abb. 3.14(c)). Wir merken an, dass es bei einer derartigem Vorgehen zu Extraktionsfehlern, z. B. bei der Analyse von Konferenzankündigungen oder dergleichen kommen kann, wenn ein Raum und eine Zeit nahe beieinander, aber nicht in gleichen Absätzen auftreten. Der Abstand wurde für die Experimente den Textfragmenten angepasst, so dass *„PAR“*, *„UIMA“* und *„UIMA-S“* äquivalente Ergebnisse erzeugen.



(a) Durchschnittliche Laufzeit

(b) Beschleunigungsfaktor



(c) Aufbau

Abb. 3.14: Laufzeitverbesserung bei indirekter Datenabhängigkeit

In Abb. 3.14(a) sind die durchschnittlichen Gesamtlaufzeiten zu den Experimenten *„PAR“*, *„UIMA“* und *„UIMA-S“* für 100 Durchläufe dargestellt. Zusätzlich sind in Abb. 3.14(b) die durchschnittlichen Beschleunigungsfaktoren für die jeweiligen Dokumentgrößen dargestellt.

Wir sehen ein ähnliches Verhalten wie in den Laufzeiten bei direkter Datenabhängigkeit in Abb. 3.11. Die Verbesserung von *„PAR“* gegenüber *„UIMA“* ist bei einer

Dokumentgröße von < 10 marginal, was den Latenzzeiten zur Datenbereitstellung geschuldet ist. Die relative Beschleunigung steigt bis zu einer bestimmten Dokumentgröße an und sinkt durch einen „Stau“ in den Warteschlangen für extrem große Dokumente. Der absolute Laufzeitvorteil steigt streng monoton. Es kann wie im vorherigen Experiment eine maximale Beschleunigung um den Faktor $3,8$ bei 4 Prozessoren erreicht werden.

Der Laufzeitvorteil gegenüber *„UIMA-S“* ist geringer, da die Textsegmentierung aufwändiger als die Überprüfung von Abständen ist. Das Verhalten ist jedoch ähnlich wie beim Vergleich von *„PAR“* und *„UIMA“*. Es kann trotz der zusätzlichen Textoperation ein maximaler Laufzeitvorteil um den Faktor $2,5$ erreicht werden.

3.4 Verwandte Ansätze zur Laufzeitoptimierung

In Kap. 2.2 wurde bereits die Neuartigkeit der vorgestellten Lösung motiviert. Im Folgenden werden Gemeinsamkeiten, Unterschiede und Integrationsszenarien von ausgewählten Ansätzen mit der hier vorgestellten Lösung diskutiert.

Die Laufzeitverbesserung von IE-Methoden ist eine in der Literatur häufig beschriebene Problemstellung. In der Vergangenheit wurden eine Vielzahl von Ansätzen zur Optimierung von Einzelaufgaben entwickelt [Sar08], wie zum Beispiel die Laufzeitoptimierung von Wörterbuchvergleichen. Andere Arbeiten untersuchten die Vorklassifizierung von Dokumenten, um Dokumente, in denen wahrscheinlich keine relevanten Entitäten vorkommen, frühzeitig zu filtern. Beide Problemstellungen sind im Kontext der vorliegenden Arbeit hinsichtlich der Implementation von Importmodulen und anderen IE-Modulen relevant. Für die Parallelisierung spielen sie jedoch keine Rolle.

Enger verwandt zu dem hier vorgestellten Ansatz sind Arbeiten, die ein Zusammenfassen von Operationen für gleiche Merkmale verfolgen, um Laufzeiten zu verbessern. Das Datenmodell von *RapidUIM* unterstützt Metadaten für Typen wie auch für Referenzen auf Wörterbucheinträge oder Regeln. Eine Verwendung von IE-Modulen, die Domänenwissen zur Extraktion verschiedener Entitäten zusammenfassen und gemeinsam evaluieren, ist also ohne weiteres möglich. Die Parallelisierung ermöglicht zusätzlich das Teilen des vereinigten Domänenwissens in Partitionen gleicher Größe und die parallele Evaluation durch mehrere Modulinstanzen und fügt derartigen Ansätzen eine weitere Dimension zur Optimierung von Laufzeiten hinzu.

In Kap. 2.1 wurde *RapidUIM* als eine Extraktionsplan-basierte Plattform zur Entwicklung von anwendungsspezifischen IE-Systemen eingeführt. In Kap. 3.4.1 werden Optimierungsverfahren anderer Extraktionsplan-basierter Plattformen diskutiert und in Beziehung zu den hier vorgestellten Verfahren gesetzt. Eine Gegenüberstellung zu Verfahren, die eine parallele Datenverarbeitung im Bereich der Informationsextraktion unterstützen, werden wir in Kap. 3.4.2 vornehmen.

3.4.1 Optimierung in der deklarativen Informationsextraktion

Ansätze zur Optimierung von deklarativen IE-Systemen basieren, wie der in dieser Arbeit vorgestellte Datenfluss-orientierte Ansatz, auf komplexen Extraktionsplänen. In diesem Bereich sind vor allem Ansätze, die im Kontext des SystemT-Projektes [RRK+08, KLR+08, MKHV09] (dem Nachfolger des AVATAR-Projektes [JKR+06]) und des CIMPLE-Projektes [CDYR08, SDNR07, SDM+08, CBC+07] entstanden sind, relevant. Die zugrundeliegende Annahme der genannten Ansätze ist die Existenz von verschiedenen äquivalenten Ausführungsreihenfolgen. Im Folgenden werden die oben genannten Ansätze beschrieben und auf die Vorteile einer Parallelisierung in derartigen Systemen eingegangen.

Laufzeitoptimierung in SystemT. *SystemT* folgt den Ideen der Anfrageverarbeitung in Relationalen Datenbanken. Es wird ein Operator für Reguläre Ausdrücke und ein Operator für die

Wörterbuch-basierte Extraktion von Entitäten für die Selektion definiert (siehe auch Bsp. 2.1 in Kap. 2.1). Zur Aggregation werden verschiedene Join-Operatoren definiert, unter denen der Blockoperator zur Aggregation von Relationen, z. B. zum Verbund von mit Wörterbüchern extrahierten Tupeln zu Beziehungen oder komplexeren Datensätzen wie Adressfeldern, der relevanteste ist.

Als Verbundprädikat des Blockoperators wird in [RRK+08] ein Kontextfenster definiert, also ein maximaler Abstand von Zeichen, in dem die Bestandteile der im Verbund teilnehmenden Tupel im Text auftreten dürfen. Die Definition von Kontextfenstern ermöglicht weitere Optimierungsmechanismen, auf die im Folgenden noch eingegangen wird. Die anderen Operatoren der Relationalen Algebra verhalten sich wie bekannt.

Bei Anwendung des Blockoperators kann die Ausführung der ihm zugrunde liegenden Selektionsoperationen effektiv auf ein definiertes Kontextfenster eingeschränkt werden. Zum Beispiel kann die Evaluation von Wörterbüchern für Personennamen bei der Extraktion von Adressfeldern auf das Textfenster um eine bereits identifizierte Postleitzahl beschränkt werden oder die Extraktion von Postleitzahlen auf den Kontext eines bereits identifizierten Personennamens. Ein weiterer Vorteil dieses Ansatzes ist es, dass überflüssige Operationen, wie z. B. die Extraktion aller Personennamen in einem Dokument bei der Extraktion von Adressfeldern verhindert wird. Je nach Kosten und Selektivität der einer Blockoperation zugrundeliegenden Operationen führen verschiedene Ausführungsreihenfolgen zu verschiedenen Laufzeiten. Auf Basis einer Kostenabschätzung mit Hilfe von Beispieldokumenten kann der Extraktionsplan mit der potenziell geringsten Laufzeit ausgewählt werden.

Laufzeitoptimierung in XLog-basierten Systemen. Die im Kontext des CIMPLE-Projektes[6] entwickelte IE-Plattform erlaubt im Gegensatz zu *SystemT* eine deklarative Beschreibung von IE-Systemen auf Basis von beliebigen IE-Modulen, die den Annahmen des Systems folgen. Die Interna eines Moduls sind der Plattform unbekannt.

IE-Module werden als Prädikate wie z. B. „extractHouses$(\bar{x}, p, a, h)$“ (siehe Bsp. 2.2 in Kap. 2.1) in einer an Datalog angelehnten Sprache eingebettet [SDNR07], wobei hier $\bar{x}$ eine Eingabevariable und p, a, h Ausgabevariablen sind. Prädikate sind auf Basis von Ein- und Ausgabevariablen verbunden und definieren demnach die Abhängigkeiten zwischen IE-Modulen und Zwischenergebnissen, also den Extraktionsplan. Neben Prädikaten können prozedurale Funktionen zur Evaluation von Nebenbedingungen definiert werden, wie z. B. „distLines$(\bar{x}, \bar{y})$ <3“, wobei $\bar{x}$ und $\bar{y}$ Eingabevariablen und das Ergebnis der Funktion der Abstand der Annotation in Zeilen ist.

In [SDNR07] werden folgende Optimierungstrategien vorgeschlagen: Zur Laufzeitoptimierung können zusätzliche prozedurale Funktionen in einen Ausführungsplan eingebettet werden. Grundlage der Optimierungsstrategie sind Abbildungen zwischen prozeduralen Funktionen und Beziehungen zwischen Annotationen. Zum Beispiel, wenn innerhalb von Sätzen Eigennamen mit der Eigenschaft „groß geschrieben“ extrahiert werden sollen, kann auf Basis der Beziehung, dass Eigennamen in Sätzen enthalten sind (und z. B. nicht mit den Satzgrenzen überlappen) die Evaluation auf Sätze eingeschränkt werden, die großgeschriebene Wörter enthalten. Die Prozedurale Funktion „enthält groß geschriebene Wörter“ kann als zusätzliche Nebenbedingung von Sätzen im Ausführungsplan eingebettet werden. Je nach Komplexität der Namensextraktion kann dieses Verfahren zu einer Laufzeitverbesserung führen.

Wenn prozedurale Funktionen die Konstellation von verschiedenen Daten beschreiben (z. B. „Abstand in Zeilen“ <3), können ähnlich wie in SystemT durch Einschränkungen des Kontextes Laufzeitverbesserungen erzielt werden. Die Verarbeitungsfolge und Eingabevariablen der IE-Module wird dementsprechend im Ausführungsplan angepasst, so dass IE-Module statt des kompletten Textes lediglich die relevanten Textfragmente konsumieren. Die Implementation von prozeduralen Funktionen definiert also neben der Überprüfung von Bedingungen auch

[6] siehe auch `http://pages.cs.wisc.edu/~anhai/projects/cimple/`, Stand: 01.05.2010

Extraktionslogik. Wie auch in *SystemT* werden mögliche Ausführungspläne für einen Extraktionsplan mit Hilfe von Kostenabschätzungen bewertet.

Die in dieser Arbeit vorgestellte *RapidUIM*-Plattform ist ein mächtigeres Werkzeug hinsichtlich der Entwicklung von generischen IE-Modulen, da diese auf beliebige Kontextdaten im Dokument-Konzept-Graph ausgehend von den Eingabedaten zugreifen können, und unterstützt damit insbesondere eine effizientere Entwicklung von IE-Systemen. Dieser Entwicklungsansatz erfordert jedoch, dass Aufgaben, die gemäß Extraktionsplan voneinander abhängig sind, genau in dieser Reihenfolge ausgeführt werden (z. B. dass Sätze identifiziert werden, bevor Entitäten und deren Beziehungen extrahiert werden). Eine Plattform-seitige Optimierung der Ausführungsreihenfolge, wie sie in [RRK+08] oder [SDNR07] vorgeschlagen wurde, ist daher nicht ohne weiteres möglich.

Die hier vorgestellten Prinzipien zur Parallelisierung können jedoch ohne weiteres in andere Extraktionsplan-basierte Systeme adaptiert werden. Zum Beispiel könnte für das laufende Beispiel aus Kap. 3.2, unter Annahme von alternativen Ausführungsreihenfolgen der Operationen, ein Ausführungsplan, dessen Ausführungssequenz in Abb. 3.15 schematisch dargestellt ist, erzeugt werden – wobei im Beispiel die Satzsegmentation zur Einschränkung des Kontexts durch die Operation „Satzkontext“ ersetzt wurde.

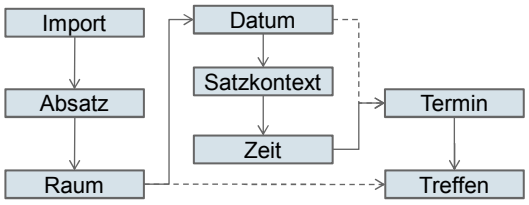


Abb. 3.15: Beispiel für optimierte Ausführungsreihenfolge

Die Datenstrom-artige Parallelisierung der Ausführung verspricht eine weitere Laufzeitverbesserung gegenüber den in [SDNR07, RRK+08] diskutierten Verfahren, da Zwischenergebnisse so früh wie möglich, nebenläufig verarbeitet werden können.

3.4.2 Parallelisierung und Verteilung von Informationsextraktionssystemen

Im Folgenden werden verwandte Arbeiten zur Parallelisierung von Datenverarbeitungssystemen diskutiert. Die Forschung im Bereich der Parallelisierung oder Verteilung von Softwareprogrammen hat eine lange Tradition. Der Fokus von *Hardware-nahen* Ansätzen liegt auf der Abschätzung und Vorhersage von Abhängigkeiten auf Basis des Programmcodes [AN88, AS92, KA96], eine Technik, die insbesondere für das *Pipelining* von kurzen Programmabschnitten auf parallelen Prozessoren, aber nicht für die Parallelisierung ganzer Komponenten geeignet ist.

In der Datenbankforschung wurden eine Reihe von Synchronisationsmechanismen zur Ausführung von konkurierenden Transaktionen [BHG87] entwickelt. Das hier vorgestellte Verfahren ist jedoch stärker mit nicht-blockierenden Verbundoperationen im Kontext von *Pipelining*-Mechanismen im Datenbankbereich [DG92] verwandt und adaptiert dessen Prinzipien in die Informationsextraktion. Hier kann eine nur kurzzeitig blockierende, nebenläufige Datenverarbeitung auf Basis der sequentiellen Struktur von Textdaten erreicht werden.

Andere Ansätze konzentrieren sich auf Beschreibungssprachen für und Ausführung von Systemen für die parallele Datenverarbeitung (siehe z. B. [JHM04] für einen Überblick). Der in Kap. 3.2 vorgestellte Ansatz adaptiert Ideen, wie z. B. die parallele Datenverarbeitung auf mehreren Prozessoren, die Berücksichtigung von Aufgabenabhängigkeiten und der Synchronisation bei Datenabhängigkeiten, die im Bereich der Datenstromsprachen wie StreamIt [TKA02],

Brook [BFH+04], CUDA [Kir07], Dryad [IBY+07], [KM08] oder [ZD10] entwickelt wurden, in den Bereich der Informationsextraktion. Insbesondere fokussiert der in der vorliegenden Arbeit diskutierte Ansatz die effiziente Erhebung von Verarbeitungsdaten und Identifikation von Abhängigkeiten für IE-Operationen, die sich von Abhängigkeiten, die normalerweise in Datenstromsytemen auftreten, unterscheiden.

Im Folgenden wird speziell auf Ansätze zur Parallelisierung im Bereich der Informationsextraktion anhand verschiedener Infrastrukturtypen [DG92] eingegangen:

- **Shared-Nothing Architektur** bezeichnet Systeme, in denen (verteilte) Prozesse mit eigenem Prozessor, Hauptspeicher und Festplatte, die Datenverarbeitung übernehmen.
- **Shared-Disk Architektur** bezeichnet die gemeinsame Nutzung von Festspeicherressourcen (z. B. einem *Network Attached Storage*), auf den von mehreren (verteilten) Prozessen konkurrierend zugegriffen werden kann.
- **Shared-Memory Architektur** bezeichnet die gemeinsame Nutzung eines Teils des Arbeitsspeichers zur Verarbeitung von Daten durch mehrere Prozesse. Alle beteiligten Prozesse können mit normalen Speicherzugriffsoperationen Daten lesen und verändern. Die einfachste Form einer *Shared-Memory Architektur* ist in Mehrkern-Prozessor-Systemen anzutreffen.

In der Informationsextraktion ist in vielen Anwendungsfällen die Verarbeitung eines Dokumentenkorpus notwendig. Der Korpus kann in solchen Fällen in Partitionen (normalerweise Einzeldokumente) geteilt und durch verschiedene Prozesse parallel in *Shared-Nothing Architekturen* verarbeitet werden. Die Grenzen in der Skalierbarkeit solcher Systeme werden nicht von der Extraktionslogik, entsprechende Ressourcen vorausgesetzt, sondern von Beschränkungen beim Zugriff auf Dokumente und der Speicherung von Ergebnisdaten limitiert und sind daher insbesondere für die Informationsextraktion im WWW oder anderer verteilter Dokumentbestände geeignet. Die Verarbeitungszeit für ein Einzeldokument wird durch derartige Ansätze nicht verbessert.

Ansätze zur Informationsextraktion in *Shared-Nothing Architekturen* werden in aktuellen Forschungsarbeiten beschrieben, sind aber auch in *Open Source* und kommerziellen Produkten zu finden. *Business Objects ThingFinder* (basierend auf [Hul00]) und *Semanta* [JPS05] adaptieren zur parallelen Dokumentverarbeitung, die Idee von Dienst-orientierten Architekturen für die Informationsextraktion. *UIMA Asynchronous Scaleout*[7] *(UIMA-AS)* als Erweiterung von *UIMA* [FL04] verwendet eine hochperformante Nachrichtendienst-basierte Verteilung von Dokumenten. In [BTMC04] wird eine ähnliche Erweiterung von *GATE* [CMBT02] vorgeschlagen.

Aktuell wird in der Literatur häufig die Verteilung und Parallelisierung von IE-Prozessen mit Hilfe von *MapReduce* [DG04, Agi05] und darauf aufbauende Sprachen [ORS+08] diskutiert. Zur parallelen Datenverarbeitung werden die einzelnen Dokumente eines Korpus verteilten Instanzen eines IE-Programms zugewiesen. Ansätze, die *MapReduce* zur Informationsextraktion verwenden sind z. B. [LSH08, LM09]. *MapReduce*-basierte Plattformen erleichtern im Falle einer Partitionierung von Dokumentenkorpora hauptsächlich das Management der Infrastruktur im Vergleich zu Dienst-orientierten oder Nachrichtendienst-basierten Ansätzen.

Anzumerken ist, dass [LM09] feingranularere Einheiten (nämlich Absätze) zur Partitionierung des Korpus verwendet, welche durch mehrstufige, verteilte Programminstanzen verarbeitet werden. Der Ansatz ist daher vergleichbar mit der Verwendung von mehreren Modulinstanzen eines Typs in *RapidUIM* im Sinne einer Stapelverarbeitung. Eine Datenstrom-basierte Verarbeitung und die Berücksichtigung von Datenabhänigkeiten wird nicht betrachtet.

Aktuelle Arbeiten [CCA+09] diskutieren die Erweiterung von *MapReduce*-basierten Ansätzen hinsichtlich einer Datenstrom-artigen Verarbeitung. Ziel ist es jedoch, Endresultate z. B. bei der Bestimmung von Termfrequenzen in einem Korpus während der Laufzeit zu approximieren. Eine Adaption für die Informationsextraktion ist nicht ohne weiteres möglich.

[7] `http://uima.apache.org/doc-uimaas-what.html`, Stand: 01.09.2010

In *Shared-Disk Architekturen* konsumieren parallele, verteilte IE-Module Ergebnisse von vorgelagerten IE-Modulen, die im Dateisystem oder Datenbanken vorgehalten werden. Eine *Shared-Disk Architektur* für die Annotation von Dokumenten im Unternehmensumfeld wird für *UIMA-GRID* [ELB07] diskutiert. Im Bereich der Informationsextraktion vom WWW wurde eine *Shared-Disk Architektur* erfolgreich im *WebFountain*-Projekt [GCG+04] adaptiert. *Shared-Disk Architekturen* sind ähnlich wie *Shared Nothing Architekturen* sehr gut zur Analyse großer Dokumentenkorpora, jedoch nicht zur Beschleunigung der Verarbeitung von Einzeldokumenten geeignet. Ein Ansatz zur Konfiguration oder effizienten Komposition solcher Systeme wird in der Literatur nicht diskutiert.

Shared-Memory Architekturen wurden im Umfeld der Informationsextraktion auf Basis von Statistischen Maschinellen Lernverfahren erfolgreich eingesetzt. Diese Verfahren können sehr gut parallelisiert werden, da für verschiedene Textsegmente alternative Ergebnisse berechnet und die Wahrscheinlichkeiten verglichen werden müssen. [SCDZ06] analysiert die Laufzeitverbesserungen für das Verfahren *Conditional Random Fields*. [RRP+07] evaluiert die generelle Anwendbarkeit des *MapReduce*-Prinzips in *Shared-Memory Architekturen* für Statistische Maschinelle Lernverfahren. [HFL+08] adaptiert *Support Vector Machines* für Graphikprozessoren und berichtet über enorme Laufzeitverbesserungen (32- bis 130-fache Beschleunigung). Die Parallelisierung von komplexen IE-Programmen wird in [SCDZ06, RRP+07, HFL+08] nicht untersucht, vielmehr handelt es sich um Ansätze zur internen Parallelisierung von potenziellen IE-Modulen im Kontext von *RapidUIM*.

Prinzipiell kann jede programmatische Parallelisierung von IE-Programmen mit Hilfe von *Multithreading-APIs* in den Bereich der *Shared-Memory Architekturen* eingeordnet werden (siehe z. B. [KK03]). Eine programmatische Parallelisierung erfordert jedoch aufwändig zu implementierende Synchronisationsmechanismen. Bei der Verwendung von *UIMA* [FL04] kann laut Handbuch[8] eine Annotation eines IE-Moduls in ein „Dokument" des Datenmodells umgewandelt und durch weitere Module unmittelbar verarbeitet werden.

Für solch eine Parallelisierung muss die Logik zur Steuerung der verschiedenen *Threads* programmatisch implementiert werden. Eine deskriptive Konfiguration von Abhängigkeiten und automatische Parallelisierung ist nicht möglich. Weiterhin werden nicht alle Abhängigkeiten zwischen Modulen oder Daten modelliert, so dass z. B. eine Synchronisation bei indirekten Datenabhängigkeiten, selbst auf programmatische Weise mit herkömmlichen System- und Datenmodellen, aufwändig zu realisieren ist.

Das in Kap. 3.2 vorgestellte Verfahren parallelisiert einzelne Komponenten von IE-Systemen, die durch Extraktionspläne beschrieben werden. IE-Module werden auf Basis von Aufgaben- und Abhängigkeitsbeziehungen automatisch parallelisiert und wenn nötig synchronisiert. Die Kommunikation zwischen den einzelnen autonom arbeitenden Modulen basiert auf Datenströmen in gemeinsam genutzten Hauptspeicherbereichen.

3.5 Zusammenfassung

In Kap. 3 wurde als Grundlage dieser Arbeit die *RapidUIM*-Plattform vorgestellt. Sie ermöglicht die Entwicklung von generischen und anwendungsspezifischen IE-Modulen auf Basis einer gemeinsamen API und stellt ein generisches Graph-basiertes Datenmodell zur Abbildung von Dokumenten, Entitäten und deren Beziehungen zur Verfügung. Insbesondere zeichnet sich *RapidUIM* dadurch aus, dass IE-Module beliebig miteinander kombiniert werden können. Die Komposition von Modulen zu IE-Systemen geschieht durch graphische Kombination der Module zu Extraktionsplänen. Es fördert damit die Effizienz in der Entwicklung von spezifischen IE-Systemen.

[8] siehe `http://uima.apache.org/downloads/releaseDocs/2.3.0-incubating/docs/html/`, Stand 01.06.2010

Für Extraktionsplattformen wie *RapidUIM*, die Abhängigkeiten von Extraktionsaufgaben und Zwischenergebnissen explizit abbilden, wurde in Kap. 3.2 eine parallelisierte, Datenstrom-artige Dokumentverarbeitung vorgestellt, die die Idee des *Pipelining*, wie sie aus herkömmlichen Datenbanksystemen bekannt ist, in die Dokumentverarbeitung zur Informationsextraktion adaptiert. Die einzelnen IE-Module werden durch Warteschlangen in geteilten Arbeitsspeicherbereichen entkoppelt.

Das Verfahren unterstützt ein hohes Maß an Aufgaben- und Datenparallelität. Aufgaben, die laut Extraktionsplan nicht voneinander abhängig sind, werden parallel ausgeführt. Weiterhin werden Daten durch nachgelagerte IE-Module unmittelbar weiterverarbeitet. Im Falle, dass indirekte Datenabhängigkeiten bestehen, findet eine effiziente Synchronisation des Datenflusses statt. Mehrere Modulinstanzen können weiterhin parallel Daten einer Warteschlange verarbeiten, wodurch eine weitere Effizienzsteigerung erreicht wird.

Die experimentelle Evaluation zeigte die praktische Anwendbarkeit der vorgestellten Methoden und untermauerte eine theoretische Analyse der zu erzielenden Laufzeitverbesserungen. Insbesondere zeigte sich, dass die Verbesserung der Gesamtlaufzeit von der konkreten Komposition der IE-Module abhängt.

Im Falle einer einfachen Aufgabenunabhängigkeit entspricht die Gesamtlaufzeit der maximalen Laufzeit der verschiedenen IE-Module. Bei den Experimenten mit zwei unabhängigen Aufgaben konnte daher lediglich eine Laufzeitverbesserung um den Faktor 1,5 trotz des Einsatzes von zwei Prozessoren im Vergleich zu einer sequentiellen Verarbeitung erzielt werden. Die Experimente für Extraktionspläne, in denen direkte und indirekte Datenabhängigkeiten vorlagen, konnte in einem Fall eine maximale Beschleunigung um den Faktor 3,1 und in zwei Fällen eine maximale Laufzeitverbesserung um den Faktor 3,8 beim Einsatz von vier Prozessoren erreicht werden.

Bisherige Ansätze zur Parallelisierung von IE-Prozessen erhöhen den Dokumentendurchsatz durch eine verteilte Stapelverarbeitung, verbessern jedoch nicht die absolute Verarbeitungszeit für ein Dokument. Die Analyse verwandter Arbeiten zeigte weiterhin, dass die hier vorgestellte Intra-Dokument-Parallelisierung von IE-Modulen mit anderen Optimierungsverfahren sehr gut harmoniert. Die effiziente Implementation von einzelnen IE-Modulen, die Zusammenführung von gleichartigen Operationen und die Kosten-basierte Auswahl von Ausführungsplänen sind orthogonale Strategien zur Laufzeitoptimierung.

Zukünftige Forschungsprobleme sind im Bereich der Kombination von Kosten-basierten Optimierungsverfahren und der vorgestellten parallelen Datenverarbeitung angesiedelt. Es stellt sich die Frage, wie eine Parallelverarbeitung bei der Kosten-basierten Auswahl von Extraktionsplänen berücksichtigt werden kann.

4

Inferenz Regulärer Ausdrücke anhand von Beispielentitäten

Im Allgemeinen können generische Extraktionsmethoden in zwei Klassen eingeteilt werden: *Regel-basierte Verfahren* und *Statistische Maschinelle Lernverfahren* (wie *Hidden Markov Models*, *Maximum Entropy Markov Models* oder *Conditional Random Fields*). Regel-basierte Extraktionsmethoden bieten im Vergleich zu Statistischen Maschinellen Lernverfahren den Vorteil, dass die Extraktionsmechanismen leichter manuell angepasst werden können und die Fehlersuche weniger komplex ist (siehe Kap. 2). Weiterhin ist für die Anwendung von Statistischen Maschinellen Lernverfahren ein großer (annotierter) Dokumentenkorpus notwendig, der für die Modellerzeugung notwendige Kontextdaten und Negativbeisiele zur Verfügung stellt, wogegen Reguläre Ausdrücke auch im Umfeld bekannter Beispielbezeichner (z. B. für Produkte) und einer geringen Anzahl von Testdokumenten entwickelt werden können.

Eine Vielzahl von Entitätstypen im Unternehmensumfeld folgen Bildungsvorschriften, die durch Reguläre Ausdrücke beschrieben werden können. In vielen Anwendungsfällen im Kontext der *Enterprise Search* sind (z. B. historische) Beispielentitäten bekannt, die z. B. in Unternehmensdatenbanken vorliegen, und zur Entwicklung von Regulären Ausdrücken verwendet werden können (siehe Kap. 1.2). In anderen Anwendungsfällen können Beispielentitäten einfach beschafft werden, wogegen ein großer annotierter Dokumentenkorpus häufig nicht verfügbar ist und kostenintensiv erstellt werden müsste.

Die Entwicklung von Regulären Ausdrücken ist jedoch insbesondere aufwändig, wenn Regeln für gleichartige Entitätstypen (z. B. Rechnungsnummern) für verschiedene Unternehmen oder Anwendungskontexte von Grund auf neu entwickelt und bei sich ändernden Anforderungen fortlaufend angepasst werden müssen. In Kap. 1.2.1 betrachteten wir z. B. ein Szenario zur Rechnungsverifikation, in dem eine manuelle Anpassung von Extraktionsmechanismen für jedes einzelne, kundenseitig installierte System schlichtweg nicht praktikabel ist.

Eine automatische Inferenz von Regulären Ausdrücken anhand weniger, gegebener Beispielentitäten hat zur praktischen Umsetzung derartiger Anwendungsfälle einen enormen Mehrwert. In Anwendungsfällen, in denen Beispielentitäten einfach beschafft werden können, kann eine Inferenz von Regulären Ausdrücken den Entwicklungsprozess wesentlich beschleunigen.

Die Konzeption eines IE-Moduls für *RapidUIM*, das auf Basis gegebener Beispielentitäten eine Menge von Regulären Ausdrücken erlernt, steht im Fokus dieses Kapitels (Teillösungen publiziert in [BRBM09]). Einzelne Problemstellungen bezüglich der Inferenz solcher Bildungs-

vorschriften wurden im Kontext von Anwendungsszenarien in Kap. 1.2 diskutiert und Grundlagen zu dem hier verfolgten Zugang mittels einer grammatikalischen Inferenz in Kap. 2.3 dargestellt.

Die in dieser Arbeit entwickelten Konzepte für das maschinelle Erlernen von Regulären Ausdrücken werden in Kap. 4.1 erläutert. Von den gegebenen Beispielinstanzen wird auf Basis verschiedener Merkmalsebenen (Zeichen- und Wortklassen) abstrahiert. Eine Überführung in Präfix- und Suffixautomaten unterstützt eine weitere Abstraktion. Um eine hohe Genauigkeit zu erreichen, werden im Weiteren feingranulare Muster (fixe Zeichenketten) auf Basis der Merkmalsverteilung in den Instanzteilsequenzen ausgewählt. Um deterministische, einfach verständliche Reguläre Ausdrücke zu erzeugen, werden Merkmalskombinationen der verschiedenen Abstraktionsebenen auf Basis des Prinzips der *Minimum Description Length* bewertet und eine hinsichtlich der Regelkomplexität und Tauglichkeit optimierte Lösung selektiert.

Eine experimentelle Evaluation der entwickelten Konzepte in vier Anwendungsfällen für acht verschiedene Typen von Entitäten und ein Vergleich der Resultate zu einem Statistischen Maschinellen Lernverfahren (*Conditional Random Fields*), das nicht mit Beispielentitäten, sondern annotierten Dokumenten trainiert wird, ist in Kap. 4.2 zu finden.

Ein Überblick zu Ansätzen, die im Kontext des maschinellen Lernens für die Informationsextraktion (IE) von Bedeutung sind, wurde bereits in Kap. 2.3 gegeben. In Kap. 4.3 werden verwandte Arbeiten, die zum Teil auch anderen Forschungsgebieten als der Informationsextraktion entstammen, detailliert hinsichtlich der algorithmischen Herangehensweise analysiert und zu dem hier vorgestellten Verfahren abgegrenzt.

4.1 Konzept zur Inferenz von Regulären Ausdrücken anhand von Beispielentitäten

Unser Konzept für die Regelinferenz anhand von Beispielentitäten basiert auf der Annahme, dass die gegebene Menge an Beispielinstanzen – notiert als $\mathcal{I}$ – eine Teilmenge einer Sprache $\mathcal{I} \in L(\mathcal{G})$ beschreibt, die durch eine Grammatik $\mathcal{G}$, der unbekannten Bildungsvorschrift, erzeugt wird. Auf Basis der Instanzbeispiele $\mathcal{I}$ soll eine reguläre Näherungsgrammatik $\mathcal{G}^{\approx}$ erlernt werden, die die ursprüngliche Bildungsvorschrift $\mathcal{G}$ approximiert, so dass in Texten auftretende, unbekannte Instanzen gleichen Typs extrahiert werden können. Formal kann die Regelinferenz demnach durch die Abbildung $\mathit{infer}(\mathcal{I})$ wie folgt ausgedrückt werden:

$$\mathit{infer} : \mathcal{I} \mapsto \mathcal{G}^{\approx}, \mathcal{I} \subseteq \mathcal{L}(\mathcal{G}), \mathcal{G}^{\approx} \approx \mathcal{G}. \tag{4.1}$$

Die Approximation $\mathcal{G}^{\approx} \approx \mathcal{G}$ kann jedoch nicht ohne weiteres definiert werden, da die Bildungsvorschrift $\mathcal{G}$ nicht bekannt ist. Vielmehr sind die Genauigkeit und Vollständigkeit bei der Extraktion von Entitäten mit $\mathcal{G}^{\approx}$ zur Beurteilung einer Annäherung von $\mathcal{L}(\mathcal{G})$ durch $\mathcal{L}(\mathcal{G}^{\approx})$ die maßgeblichen Effektivitätskriterien. Die höchste Genauigkeit kann durch Verwendung der Beispielinstanzen als Wörterbuch erreicht werden. Diese Variante ist aber ganz klar zur Extraktion unbekannter Entitäten ungeeignet.

Eine Abstraktion von den Beispielinstanzen, z. B. durch Substitution von Zeichen oder Wörtern durch Zeichen- oder Wortklassen, ermöglicht eine hohe Vollständigkeit. Zeichenklassen abstrahieren weniger von den konkreten Instanzen als Wortklassen. Es gilt: Je höher der Abstraktionsgrad, desto geringer die Genauigkeit.

Die Problemstellung besteht also darin, eine Bildungsvorschrift $\mathcal{G}^{\approx}$ abzuleiten, die den Aufbau der Beispielinstanzen $\mathcal{I}$ auf verschiedenen höheren Abstraktionsebenen abbildet (Zeichen- und Wortklassen), aber auch signifikante Instanzteilsequenzen als solche berücksichtigt, z. B. wenn ein Zeichen oder Wort an einer bestimmten Position in den Entitäten zu erwarten ist (z. B. ein gemeinsames Präfix).

Überblick: Kapitel 4.1 gliedert sich in zwei Teile: Zu Beginn gehen wir auf die Abstraktion von den Instanzen auf verschiedenen Merkmalsebenen ein. Im zweiten Teil steht eine Spezialisierung durch eine Selektion von signifikanten Instanzteilsequenzen und die Auswahl von geeigneten Abstraktionsebenen im Vordergrund. In Alg. 3 ist der generelle Ablauf der Regelinferenz dargestellt, an dem sich der Aufbau der Unterkapitel orientiert.

Algorithmus 3 infer(): Inferenz von Regulären Ausdrücken anhand positiver Instanzbeispiele

Input: Liste der Beispielinstanzen: I, Merkmalklassen: $\mathcal{K}$
Output: eine Menge Regulärer Ausdrücke: R

1: // **Abstrahiere gemäß Wortklassen $\mathcal{K}_\mathcal{T} \subset \mathcal{K}$.**
2: $C \leftarrow$ *erzeugePartitionen*$(I, \mathcal{K}_\mathcal{T})$ // Klassifikation von Wortsequenzen und Partitionierung.
3: // **Abstrahiere durch Präfix- und Suffixkodierung.**
4: $A_{\text{prefix}} =$ *erzeugeAutomat*(C, I) // Präfixautomat auf Basis der Wortsequenzklassifikationen.
5: $A_{\text{suffix}} =$ *erzeugeAutomat*(C^{-1}, I^{-1}) // Suffixautomat mit entsprechend invertierten Eingaben.
6: // **Erzeuge parallele Abstraktionsebene gemäß Zeichenklassen $\mathcal{K}_\mathcal{Z} \subset \mathcal{K}$.**
7: $A_{\text{prefix}} =$ *ergänzeZeichklassenTransitionen*$(A_{\text{prefix}}, \mathcal{K}_\mathcal{Z})$ // Klassifiziere Wörter mit Zeichenklassen ..
8: $A_{\text{suffix}} =$ *ergänzeZeichklassenTransitionen*$(A_{\text{suffix}}, \mathcal{K}_\mathcal{Z})$ // und füge parallele Abstraktionsebene ein.
9: // **Selektiere fixe Bestandteile anhand der Verteilung von Instanzteilsequenzen.**
10: $A_{\text{prefix}} =$ *fixeBestandteile*(A_{prefix}) // Analysiere Merkmalsverteilung und wähle ..
11: $A_{\text{suffix}} =$ *fixeBestandteile*(A_{suffix}) // signifikante Wörter und Zeichen aus.
12: // **Selektiere die signifikantesten Merkmale aus Zeichen- und Wortmerkmalsebene.**
13: $A_{\text{prefix}} = MDL(A_{\text{prefix}})$ // Bewertung durch „Minimum Description Length (MDL)“ ..
14: $A_{\text{suffix}} = MDL(A_{\text{suffix}})$ // und Auswahl der am besten geeigneten Abstraktionsebenen.
15: // **Erzeuge und bewerte Reguläre Ausdrücke.**
16: **for all** $i \in I$ **do**
17: $(\text{r}_{\text{prefix}}, \text{dl}_{\text{prefix}}) = A_{\text{prefix}}.getRegEx(i)$ // Erzeuge Regulären Ausdruck für Instanz und ..
18: $(\text{r}_{\text{suffix}}, \text{dl}_{\text{suffix}}) = A_{\text{suffix}}.getRegEx(i)$ // speichere die entsprechenden „Description Length“.
19: **if** $\text{dl}_{\text{prefix}} \leq \text{dl}_{\text{suffix}}$ **then**
20: $R \leftarrow \text{r}_{\text{prefix}}$ // Regulärer Ausdruck aus Präfixautomat ist besser geeignet.
21: **else**
22: $R \leftarrow \text{r}_{\text{suffix}}$ // Regulärer Ausdruck aus Suffixautomat ist besser geeignet.
23: **end if**
24: **end for**
25: **return** R

Kapitel 4.1.1: Zunächst betrachten wir grundlegende Abstraktionsschritte von den Instanzen hin zu einer Merkmalsbeschreibung durch konkatenierte Zeichen- und Wortklassen. Wir beschreiben eine initiale Partitionierung der Beispielinstanzen hinsichtlich ihrer syntaktischen Struktur auf Wortebene (Alg. 3.(2)) und gehen auf eine Überführung der syntaktischen Struktur in Präfix- und Suffixautomaten ein (Alg. 3.(4) und Alg. 3.(5)). Die Überführung der Merkmalsbeschreibungen in Präfix- und Suffixautomaten basiert auf der Beobachtung, dass Bezeichner im Unternehmenskontext häufig von rechts nach links oder umgekehrt gebildet werden.

In einem weiteren Schritt werden als Alternative zu jeder Wortklassifikation in einem Automaten eine Menge von parallelen Transitionen auf Zeichenebene erzeugt (Alg. 3.(7) und Alg. 3.(8)). Für alle Wort- und Zeichenklassifikationen, die Transitionsmarken der Automaten sind, werden die zugehörigen Instanzteilsequenzen der Beispielentitäten, die zu der Klassifikation geführt haben, gespeichert. Ergebnis ist jeweils ein Nicht-deterministischer Endlicher Automat (NEA), der alle Eingabeinstanzen I auf unterschiedlichen Abstraktionsebenen akzeptiert.

Kapitel 4.1.2: In Kap. 4.1.2 analysieren wir die Umwandlung der NEAs in Deterministische Endliche Automaten, die einfach in Reguläre Ausdrücke überführt werden können. Dazu ist die Auswahl der signifikantesten Merkmale, die die Instanzen am besten repräsentieren, notwendig. Zunächst wird in Kap. 4.1.2 die Bewertung von feingranularen Mustern diskutiert, also die Selektion von fixen Bestandteilen, die als solche statt Zeichen- bzw. Wortklassen in den Zielausdrücken berücksichtigt werden sollen (siehe Alg. 3.(10) und Alg. 3.(11)). Es entsteht jeweils

ein Automat, der Zeichen, Wörter, Zeichenklassen und Wortklassen als Übergänge zwischen Zuständen definiert.

Um deterministische Reguläre Ausdrücke zu erzeugen, müssen bisher erkannte Muster der verschiedenen Abstraktionsebenen bewertet und miteinander verglichen werden. In Kap. 4.1.2 diskutieren wir das Prinzip der *Minimum Description Length (MDL)* und dessen Anwendung zur Bewertung und Auswahl von alternativen Musterbeschreibungen. Die NEAs werden durch die Merkmalsselektion in Deterministische Endliche Automaten transformiert (siehe Alg. 3.(13) und Alg. 3.(14)).

Ein Deterministischer Endlicher Automat (DEA) kann einfach in Reguläre Ausdrücke überführt werden (siehe Alg. 3.(17) und Alg. 3.(18)). Für jede Instanz wird der Reguläre Ausdruck aus Präfix- oder Suffixautomat in den finalen Regelsatz (insofern noch nicht vorhanden) eingefügt, der die geringste „*Description Length*" (notiert als *dl* in Alg. 3) gemäß der MDL-Berechnung erzeugt.

Die so erlernten Regulären Ausdrücke können ohne weiteres in beliebigen herkömmlichen Regel-basierten Extraktionssystemen eingesetzt werden, wie zusammenfassend in Kap. 4.1.3 beschrieben wird. Wir diskutieren weiterhin mögliche manuelle und automatische Modifikationstechniken, die zur Optimierung der Regulären Ausdrücke angewendet werden können, wenn ein annotierter Trainingskorpus zur Verfügung steht.

4.1.1 Klassifikation, Partitionierung und Automatenerzeugung

Im Folgenden werden Merkmale zur Partitionierung von Instanzen sowie das Verfahren zur induktiven grammatikalischen Inferenz zur Mustererkennung erläutert.

Die meisten Sprachen zur Entwicklung von Regulären Ausdrücken definieren eine Menge von Zeichenklassen, z. B. für Zahlen oder Sonderzeichen, die als Merkmale zur Beschreibung von Bildungsregeln herangezogen werden können, da sie übliche syntaktische Konstrukte repräsentieren. Wortklassen, wie z. B. Wörter, die nur aus Großbuchstaben bestehen, oder Wörter, die sowohl Zahlen als auch Großbuchstaben enthalten, werden daraufhin definiert. Abschließend gehen wir auf die syntaktische Klassifikation, die Partitionierung von Instanzen und die Erzeugung von Präfix- und Suffixautomaten ein.

Merkmalsdefinition

Sei Σ_{Basis} ein Alphabet von Basiszeichen, die in Regulären Ausdrücken eines Extraktionssystems verwendet werden können (Buchstaben, Zahlen, Sonderzeichen, wie z. B. durch *UTF-8*[1] defininiert). Auf Basis von Σ_{Basis} kann durch Alternation von ausgewählten Zeichen eine Menge von Zeichenklassen (notiert als $\mathcal{K}_Z$) definiert werden, die im Hinblick auf die Regelinferenz sinnvoll erscheint. Aufbauend auf den Zeichenklassen werden durch Konkatenation eine Menge von Tokenklassen (notiert als $\mathcal{K}_T$) definiert.

Zeichenklassen: Im Folgenden wird eine Standardkonfiguration für Zeichenklassen $\mathcal{K}_Z$ in der Syntax der Java API für Reguläre Ausdrücke[2] vorgeschlagen, wobei wir die geläufigere „Bereichssyntax" (z. B. $[\mathrm{A}-\mathrm{Z}]$), statt der korrekten Unicode-Variante (wie z. B. $[\backslash\mathrm{p}\{\mathrm{Uppercase\_Letter}\}]$), zur Illustration verwenden. Für spezielle Anwendungsfälle sind andere Konfigurationen vorstellbar. Spitzklammern werden im Folgenden zur Kennzeichnung von Platzhalterklassen verwendet.

[1] ISO/IEC 10646-3:2003

[2] `http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html`, Stand 01.05.2010

Die Standardkonfiguration für Zeichenklassen $\mathcal{K}_Z$ ist wie folgt definiert:

$$\begin{aligned}
\mathcal{K}_Z &:= \{\langle \text{WT}\rangle, \langle \text{SC}\rangle, \langle \text{NB}\rangle, \langle \text{UC}\rangle, \langle \text{LC}\rangle\} \\
L(\langle \text{WT}\rangle) &:= \backslash \text{s} \\
L(\langle \text{SC}\rangle) &:= (-|_|\S|\$|\backslash|\%|\&|/|*|\#|.|,|;|\backslash|\ldots|?) \\
L(\langle \text{UC}\rangle) &:= [\text{A}-\text{Z}] \\
L(\langle \text{NB}\rangle) &:= [0-9] \\
L(\langle \text{LC}\rangle) &:= [\text{a}-\text{z}]
\end{aligned}$$

Leerzeichen und andere Arten von Freiräumen $\langle \text{WT}\rangle$ (*„Whitespace“*) begrenzen Wörter und sind daher ein wichtiges Merkmal hinsichtlich der Struktur von Wortfolgen. Sonderzeichen $\langle \text{SC}\rangle$ sind ein sehr signifikantes Merkmal im Hinblick auf Bezeichner im Unternehmensumfeld. Zahlenfolgen $\langle \text{NB}\rangle$ und Folgen von Großbuchstaben $\langle UC\rangle$ sind ein weiteres Merkmal und insbesondere interessant, wenn sie nicht am Beginn eines Wortes auftreten. Aber auch Kleinbuchstaben $\langle \text{LC}\rangle$ können in Kombination mit anderen Merkmalen ein aussagekräftiges Muster darstellen.

Tokenklassen: Wörter werden im Folgenden als Token und Wortklassen als Tokenklassen bezeichnet. Wir definieren Token wie folgt, erlauben jedoch die Konfiguration von alternativen Tokenisierungsverfahren.

Definition 4.1.1 (*Token*)

Als Token werden im Weiteren Sonderzeichen $\langle \text{SC}\rangle$ und Leerzeichen $\langle \text{WT}\rangle$, sowie die durch sie begrenzten Zeichenketten aufgefasst.

Diese Art der Tokenbildung ist, abgesehen von der Berücksichtigung von Leerzeichen, konform mit vielen in der Praxis eingesetzten Tokenisierungsverfahren.

Tokenklassen $\mathcal{K}_T$ basieren auf den Zeichenklassen $\mathcal{K}_Z$ und werden wie folgt definiert:

Definition 4.1.2 (*Tokenklassen*)

Die Menge der Tokenklassen wird durch die Potenzmenge von $\mathcal{K}_Z$ abzüglich der leeren Menge definiert: $\mathcal{K}_T := 2^{\mathcal{K}_Z} \backslash \emptyset$. Eine Menge in der Menge $2^{\mathcal{K}_Z} \backslash \emptyset$ charakterisiert eine Tokenklasse.

Im Folgenden werden Tokenklassen durch die Alternation der zugrunde liegenden Zeichenklassen, ergänzt um ein Kleene Plus, zur Illustration notiert, wobei die Reihenfolge der Zeichenklassen in der Notation keine Rolle spielt. Zum Beispiel ergeben sich mit $\mathcal{K}_Z := \{\langle UC\rangle, \langle NB\rangle\}$ die Tokenklassen $\mathcal{K}_T := \{\langle \text{UC}\rangle^+, \langle \text{NB}\rangle^+, \langle \text{UC}|\text{NB}\rangle^+\}$.

Die Sprache $L(\kappa_T), \kappa_T \in \mathcal{K}_T$ von einfachen Tokenklassen, die durch genau eine Zeichenklasse gebildet werden ($\kappa_T \in \mathcal{K}_T, |\kappa_T| = 1$), ergibt sich durch die positive Kleensche Hülle der entsprechenden Zeichenklasse: $L(\kappa_T) = \kappa_Z^+, |\kappa_T| = 1, \kappa_Z \in \kappa_T$. Statt die durch Kombination verschiedener Zeichenklassen entstandenen Tokenklassen, wie z. B. $\langle \text{UC}|\text{NB}\rangle^+$ als Verallgemeinerung der Sprache einfacherer Tokenklassen, wie z. B. $\langle \text{NB}\rangle^+$ anzusehen, werden komplexere Tokenklassen als spezielleres Merkmal angesehen. Diese Vorgehensweise reflektiert, dass im Allgemeinen Token der Art *„A0aa“* oder *„a0A“* als spezifischer angesehen werden als z. B. *„Aaaa“* und verhindert in der initialen Partitionierung, dass derartige Token als „ähnlich“ klassifiziert werden.

Formal kann die Spezialisierung von komplexeren Tokenklassen durch eine „Subtraktion“ der Sprachen einfacherer Tokenklassen von der verallgemeinerten Sprache, die alle Zeichenklassen berücksichtigt, ausgedrückt werden. Auf eine exakte formale Definition wird

an dieser Stelle verzichtet. Ein Beispiel soll die Differenzbildung zur Definition von Tokenklassensprachen illustrieren: Angenommen $\mathcal{K}_Z := \{\langle \text{UC} \rangle, \langle \text{NB} \rangle\}$ sei die Menge aller Zeichenklassen, so ist die Menge der Tokenklassen definiert als $\mathcal{K}_T := \{\langle \text{UC} \rangle^+, \langle \text{NB} \rangle^+, \langle \text{UC}|\text{NB} \rangle^+\}$. Die Sprache der Tokenklasse $L(\langle \text{UC}|\text{NB} \rangle^+)$ wird demnach durch den Regulären Ausdruck $L(\langle \text{UC}|\text{NB} \rangle^+) := [A-Z0-9]^+ \backslash ([A-Z]^+ \cup [0-9]^+)$ definiert, so dass z. B. „AA“ $\in L(\langle \text{UC} \rangle^+)$ und „$A0A$“$\in L(\langle \text{UC}|\text{NB} \rangle^+)$ aber „AA“ $\notin L(\langle \text{UC}|\text{NB} \rangle^+)$ gilt.

Initiale Partitionierung

Ziel der initialen grammatikalischen Inferenz ist es, aufbauend auf dem Alphabet $\Sigma = \mathcal{K}_Z \cup \mathcal{K}_T$, eine Grammatik zu erlernen, die den abstrakten syntaktischen Aufbau der Instanzmenge I beschreibt. Die Grammatik wird durch Automaten dargestellt, deren Transitionen die sequentielle Abfolge von Merkmalen repräsentieren. Zuvor werden Duplikate in den Instanzbeispielen entfernt, da diese keinen Mehrwert hinsichtlich der grammatikalischen Inferenz von Bildungsvorschriften bieten. Anzumerken ist, dass wir in einem später erläuterten Spezialisierungsschritt das dem Automaten zugrundeliegende Alphabet um Σ_{Basis} erweitern.

Jede Instanz $i \in I$ wird, wie in Alg. 4 dargestellt, tokenenisiert und syntaktisch auf Basis von Tokenklassen $\mathcal{K}_T$ klassifiziert. Ergebnis ist eine Partitionierung der Instanzen nach Tokenmerkmalssequenzen. Die Tokenmerkmalssequenz, die den Partitionsschlüssel repräsentiert, beschreibt die gemeinsame Sprache der zugehörigen Instanzen auf Tokenebene.

Algorithmus 4 erzeugePartitionen() – siehe auch Alg. 3

Input: Liste der Instanzdaten: I , Tokenklassen: $\mathcal{K}_\mathcal{T}$
Output: Partitionen gemäß Klassifikationssequenz: C

```
1: C ← ∅
2: for all i ∈ I do
3:     c_i ← ∅    // Klassifikationssequenz für i
4:     for all token ∈ tokenisiere(i) do
5:         c_i ⊕ klassifiziere(token, K_T)
6:     end for
7:     C ← (c_i, i)    // Partitionsschlüssel ist c_i
8: end for
```

In Tab. 4.1 ist ein Beispiel für die Partitionierung ausgewählter Instanzen des Anwendungsbeispiels für Notebookbezeichner aus Kap. 1.2.1 dargestellt. Die Klassifikationen der Partitionen könnten bereits zu diesem Zeitpunkt in Reguläre Ausdrücke überführt werden, die jedoch zu einer äußerst geringen Genauigkeit bei der Extraktion führen würden.

Tab. 4.1: Initiale Klassifikation und Partitionierung von Produktnamen

Initiale Klassifikation $c^i \in C$	**Instanz** $i \in I$
$\langle \text{LC}|\text{NB} \rangle^+$	m8100y m8000z dv8000 zd8000 zd7000
$\langle \text{LC}|\text{NB} \rangle^+ \langle \text{WT} \rangle^+ \langle \text{UC} \rangle^+$	d5100t ATX d5000t ATX
$\langle \text{LC}|\text{NB} \rangle^+ \langle \text{WT} \rangle^+ \langle \text{LC}|\text{UC} \rangle^+ \langle \text{WT} \rangle^+ \ldots$	tx1000z Tablet PC tx2000z Tablet PC
$\langle \text{LC}|\text{NB} \rangle^+ \langle \text{WT} \rangle^+ \langle \text{LC}|\text{UC} \rangle^+ \langle \text{WT} \rangle^+ \ldots$	dv7 Artist Edition
$\langle \text{LC}|\text{UC} \rangle^+ \langle \text{WT} \rangle^+ \langle \text{NB} \rangle^+ \langle \text{WT} \rangle^+ \ldots$	Mini 1000 Mi Mini 1099 Eg
$\langle \text{LC}|\text{UC} \rangle^+ \langle \text{WT} \rangle^+ \langle \text{NB} \rangle^+ \langle \text{WT} \rangle^+ \ldots$	Mini 1000 Mobile Broadband Wireless

Erzeugung von Präfix- und Suffixautomaten

Die initiale Klassifikation und Partitionierung ist Ausgangspunkt für weitere Generalisierungs- und Spezialisierungsheuristiken. Bei typischen Bezeichnern im Bereich der *Enterprise Search* kann oft davon ausgegangen werden, dass sie von rechts nach links oder umgekehrt gebildet werden. Daher ist eine Generalisierung durch eine gemeinsame Mustersuche in Anfangs- bzw. Endsequenzen von Instanzen über Partitionsgrenzen hinweg sinnvoll.

Es wird jeweils ein Präfix- und Suffixautomat erzeugt, der die Beispielinstanzen mit syntaktisch ähnlichem Aufbau zusammenführt. Im Folgenden wird lediglich auf die Erzeugung von Präfixautomaten eingegangen. Die Vorgehensweise für Suffixautomaten ist äquivalent, abgesehen davon, dass die Beispielinstanzen und die Tokenklassifikationsequenz invertiert betrachtet wird (siehe auch Alg. 3.(4) und Alg. 3.(5)).

Auf den Tokenklassifikationsequenzen wird von links nach rechts iteriert und für jede Tokenklasse eine neue Transition im Automaten erzeugt (insofern nicht vorhanden). Als alternative Merkmalsbeschreibung werden für jede Transition alle Token im Automaten gespeichert, die ursprünglich in den Beispielinstanzen an dieser Position vorkamen. Ein Beispiel für einen Präfixautomaten für die Beispielinstanzen aus Tab. 4.1 ist in Abb. 4.1 dargestellt.

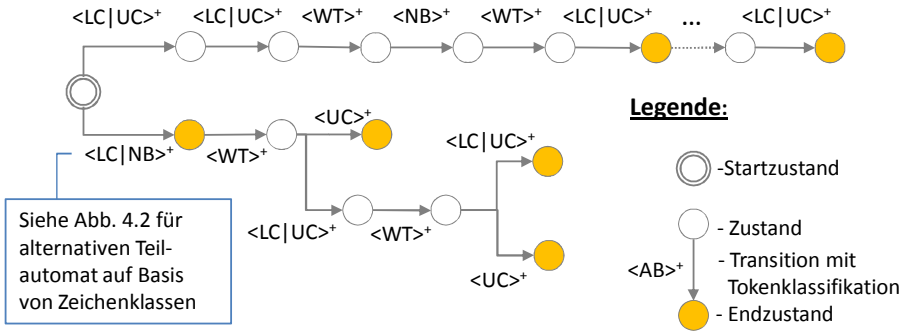


Abb. 4.1: Präfixautomat für die Beispielinstanzen aus Tab. 4.1 auf Tokenebene

Alternative Teilautomaten auf Zeichenebene

Nachdem Tokenklassen analysiert wurden, werden als weitere Abstraktionsebene die Zeichenklassensequenzen der einzelnen Token betrachtet, die den Transitionen im Automaten zugrundeliegen (siehe Alg. 3.(7) und Alg. 3.(8)). Das Vorgehen ist ähnlich der Klassifikation von Beispielentitäten (siehe Alg. 4): Statt der kompletten Instanzen werden jeweils die Menge an Token, die einer Transition auf Tokenebene zugrundeliegt, als Eingabe verwendet. Token werden in Zeichen zerlegt und gemäß der Zeichenklassen $\mathcal{K}_Z$ klassifiziert.

Für die Klassifikationssequenzen auf Zeichenebene soll ähnlich wie im Falle der Tokenklassifikationen ein präfixkodierter Teilautomat erzeugt werden, der aber als Alternative zu einer Transition auf Tokenebene im eigentlichen Präfixautomat eingebettet wird. Da verschiedene Token, die derselben Tokenklassifikation zugrunde liegen, potenziell verschiedenen Bildungsregeln folgen, unterscheidet sich der Algorithmus zur Partitionierung der Instanzteilsequenzen. Statt jede Klassifikationsmarke einzeln zu betrachten, werden Blöcke gleicher Zeichenklassifikationen herangezogen.

Zur Identifikation von so genannten Zeichenklassenblöcken wird jeder einzelne Token, der einer Transition im Automaten zugrunde liegt, auf Basis der Zeichenklassen $\mathcal{K}_Z$ syntaktisch klassifiziert. Klassifikationssequenzen werden in einer Matrix, wie in Abb. 4.2.(1) dargestellt, linksbündig ausgerichtet und die Zeilen entsprechend der Klassifikationssequenzen sortiert.

Die Token, die der Beispielmatrix zugrunde liegen sind ganz links in Abb. 4.2.(1) dargestellt (siehe auch Abb. 4.1 und Tab. 4.1).

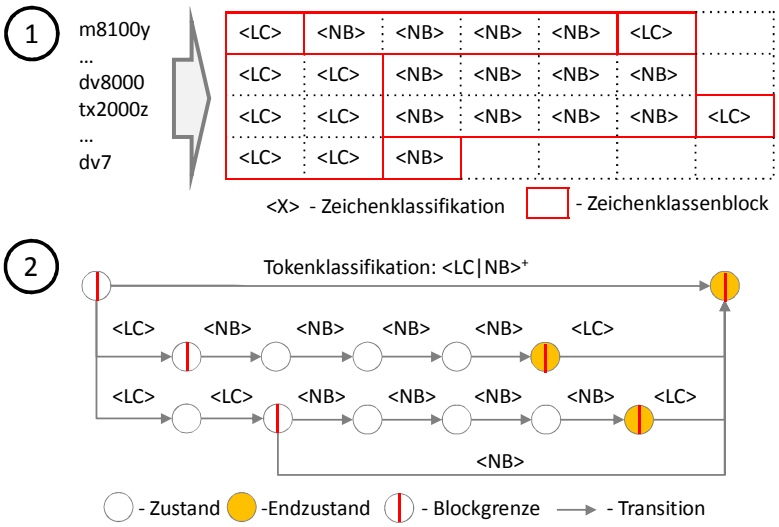


Abb. 4.2: Erzeugung eines alternativen Teilautomaten auf Zeichenebene

Die erzeugte Matrix wird in eine Blockmatrix überführt, wobei ein Block durch gleiche Zeichenklassen, wie in Abb. 4.2.(1) dargestellt, gekennzeichnet ist. Sequenzen gleicher Zeichenklassen werden dabei zuerst innerhalb einer Zeile detektiert und dann zeilenübergreifend zu Blöcken zusammengefasst.

Eine spaltenweise Ausrichtung von Blöcken mit gleicher Zeichenklassensequenz (z. B. beide Blöcke mit der Sequenz ⟨NB⟩⟨NB⟩⟨NB⟩⟨NB⟩ in Abb. 4.2.(1)) und erweiterte Zusammenfassung wäre für das gegebene Beispiel vorstellbar. Ein solches Vorgehen vereinigt jedoch in der Praxis häufig syntaktisch nicht zusammengehörige Blöcke verschiedener Bildungsvorschriften und führt zu einer geringeren Güte der Muster in späteren Spezialisierungsphasen.

Als alternative Darstellungsform einer Transition auf Tokenebene werden im vorher erzeugten Präfixautomaten Sequenzen von Transitionen für Zeichenklassen eingefügt. Für jeden Block in der Blockmatrix wird demnach eine Sequenz von Transitionen entsprechend der erkannten Zeichenklassen erzeugt. Die Blocksequenz wird ebenfalls präfix-kodiert abgebildet (siehe Abb. 4.2.(2)). Es entsteht demnach ein Teilautomat, der mit den Zuständen der entsprechenden Transition auf Tokenebene verbunden ist. Es werden wiederum die einer Transition zugrunde liegenden Bestandteile der Beispielinstanzen für spätere Spezialisierungsschritte gespeichert.

Als Resultat der bisherigen Inferenzschritte entsteht ein Nicht-deterministischer Endlicher Automat, der mögliche Muster der Beispielinstanzen auf parallelen Abstraktionsebenen darstellt. In einem späteren Verarbeitungsschritt werden die Transitionen der verschiedenen Abstraktionsebenen bewertet und die am besten geeigneten Merkmalsrepräsentationen selektiert. Zuvor werden jedoch alle Transitionen aller Abstraktionsebenen auf feingranulare Muster hin analysiert, also bewertet, ob ein Zeichen oder Token an einer bestimmten Position der Zeichen- oder Tokenklasse vorzuziehen ist.

4.1.2 Musterbewertung und -selektion

In Abb. 4.3 ist der aktuelle Verarbeitungszustand für einen Teilabschnitt des Präfixautomaten aus Abb. 4.1 ausschnittsweise dargestellt, wobei „␣“ ein Leerzeichen notiert. Token- und Zeichenklassen stellen verschiedene Abstraktionsebenen zur Beschreibung der Instanzteilse-

quenzen dar. Zusätzlich werden zu jeder Klassifikation die Menge von Instanzteilsequenzen gespeichert, die als weitere Alternative zur Musterbeschreibung in Frage kommen.

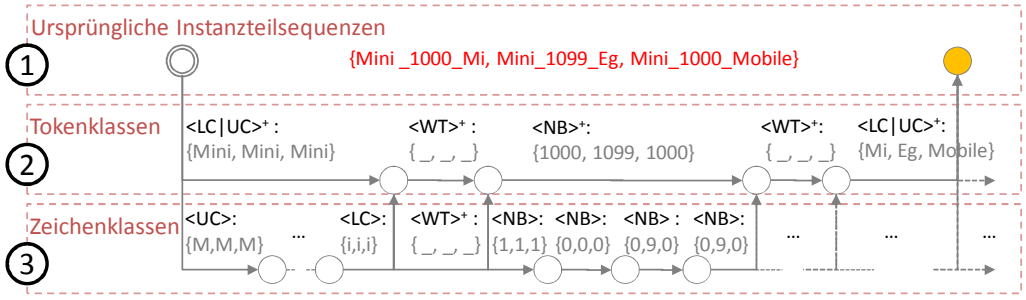


Abb. 4.3: Alternative Transitionen auf verschiedenen Abstraktionsebenen

Im Folgenden werden Heuristiken zur Suche von fixen Zeichenketten (feingranularen Mustern) zur Verbesserung der Genauigkeit und Heuristiken zur Bewertung der alternativen Musterbeschreibungen auf verschiedenen Abstraktionsebenen vorgestellt. Ziel ist es, den NEA in einen DEA umzuwandeln, der die signifikantesten Muster der Beipielinstanzen repräsentiert und in einen oder mehrere Reguläre Ausdrücke umgewandelt werden kann.

Zu Beginn werden Kriterien für die Bewertung und Auswahl von feingranularen Mustern definiert (siehe auch Alg. 3.(10) und Alg. 3.(11)). Die Bewertung der verschiedenen Abstraktionsebenen für Teile des NEA wird im darauf folgenden Unterkapitel diskutiert (Alg. 3.(13) und Alg. 3.(14)). Zum Abschluss dieses Kapitels wird die finale Musterselektion erläutert, durch die der NEA in einen DEA transformiert wird, und die Erzeugung von Regulären Ausdrücken beschrieben (Alg. 3.(16) – Alg. 3.(25)).

Identifikation von feingranularen Mustern

Ein feingranulares Muster kann im Allgemeinen als eine Enumeration von Token oder Zeichen beschrieben werden, die an einer bestimmten Position in den Instanzen zu erwarten ist. Bei einem feingranularen Muster handelt es sich also um ein lokal begrenztes Phänomen, das die Unbestimmtheit der Musterbeschreibung im NEA verringert und die Genauigkeit verbessert.

Feingranulare Muster sind zum einen dadurch gekennzeichnet, dass sie einen kleineren Wertevorrat im Vergleich zur entsprechenden Zeichen- oder Tokenklasse haben, so dass z. B. nur gewisse Zahlen statt aller Zahlen, oder bestimmte statt beliebiger Wörter bei der Extraktion berücksichtigt werden. Für Zeichenklassen $\mathcal{K}_Z$ ist der Wertevorrat vordefiniert. Sei $W : \mathcal{K} \mapsto N$ die Abbildung einer Zeichen- oder Tokenklasse $\mathcal{K}$ auf die Mächtigkeit des Wertevorrats, so gilt z. B. $W(\langle \text{NB} \rangle) = 10$.

Für Tokenklassen kann der Wertevorrat nicht so einfach bestimmt werden. Für englische Texte ist z. B. eine Konfiguration $W(\langle \text{LC} \rangle) \approx 1.000.000$ entsprechend der Wortanzahl in der Englischen Sprache [MCM03] vorstellbar, aber wenig sinnvoll. Vielmehr ist hier die Lesbarkeit der final erzeugten Regulären Ausdrücke zu betrachten, so dass eine maximal erlaubte Anzahl von Token für eine Transition konfiguriert werden kann (z. B. $W(\kappa) = 100, \kappa \in \mathcal{K}_T$ für alle Tokenklassen). Es ist anzumerken, dass in den hier evaluierten Anwendungsfällen, trotz einer hohen Anzahl von Beipielentitäten, aufgrund des zweiten, später erläuterten Auswahlkriteriums kein Fall auftrat, in dem mehr als 10 Token als feingranulares Muster in Frage kamen.

Das erste Kriterium zur Selektion von feingranularen Mustern wird durch einen konfigurierbaren Schwellwert $\alpha \in [0, 1]$ für das zulässige Verhältnis von individuellen Instanzteilsequenzen zum Wertevorrat der Zeichen- oder Tokenklasse definiert. Sei I'_t die Menge der Instanzteilsequenzen, die einer Transition t zugrunde liegen, und $\kappa_t \in \mathcal{K}$ die entsprechende Klassifikations-

marke von t, so ergibt sich das erste Kriterium zur Klassifikation von Instanzteilsequenzen I'_t als feingranulares Muster wie folgt:

$$\frac{|I'_t|}{W(\kappa_t)} \leq \alpha. \tag{4.2}$$

Wir verwenden im Folgenden die Standardkonfiguration $\alpha = 0,5$, die sich im Allgemeinen hinsichtlich der leichten Interpretierbarkeit der Resultate als sinnvoll erwiesen hat. Für den einfachen Automatenausschnitt aus Abb. 4.3 ist die Bedingung 4.2 für alle dargestellten Transitionen erfüllt.

Ein maßgeblicheres Kriterium für die Auswahl ist eine Beurteilung dahingehend, ob eine Menge von Instanzteilsequenzen I'_t an einer bestimmten Stelle der Beispielinstanzen zu erwarten ist. Durch die Klassifikation von Zeichen und Zeichenketten und die Präfixkodierung der Klassifikationssequenzen wurden potenziell ähnliche Zeichenketten gruppiert.

Signifikante feingranulare Muster zeichnen sich nun dadurch aus, dass die Mächtigkeit der Menge der Instanzteilsequenzen einer Transition $|I'_t|$ ins Verhältnis gesetzt zu deren absoluter Anzahl – notiert als $A(I'_t)$ – relativ gesehen gering ist. Im Beispiel aus Abb. 4.3 ergibt sich z. B. für die Transition mit den Token {Mini, Mini, Mini} das Verhältnis $|I'_t|/A(I'_t) = 1/3$, für das Beispiel {1000, 1099, 1000} ein $|I'_t|/A(I'_t) = 2/3$ und für {Mi, Eg, Mobile} ein $|I'_t|/A(I'_t) = 1$.

Wie das Beispiel zeigt, ergeben sich, wenn die unbekannte Bildungsvorschrift tatsächlich einem präfix-artigen Bildungsschema wie in Abb. 4.3 folgt, niedrige Verhältnisse in der Nähe des Startzustandes und höhere Verhältnisse nahe der Endzustände. Wenn es sich um suffix-artiges Bildungsschema handelt, kann dieses Verhalten im Suffixautomaten beobachtet werden.

Die Verteilungen der beschriebenen Verhältnisse in den einzelnen Pfaden des Präfixautomaten weichen zum Teil stark voneinander ab. Daher ist die Konfiguration eines absoluten Schwellwertes für $|I'_t|/A(I'_t) \in (0,1]$ als Vergleichskriterium nicht geeignet. Vielmehr ist als relatives Vergleichskriterium der Pfad, also die Sequenz von Transitionen zwischen dem Startzustand und möglichen Endzuständen, der eine untersuchte Transition t enthält, heranzuziehen. Weiterhin sind lediglich die Transitionen in dem Pfad mit derselben Abstraktionsebene wie die aktuell untersuchte Transition zu betrachten.

Wir verwenden als Vergleichskriterium für $|I'_t|/A(I'_t)$ den Mittelwert der beschriebenen Verhältnisse im zur Transition zugehörigen Pfad im Präfixautomaten. Sei V die Menge aller Vorgängertransitionen eines Abstraktionsgrades, die durch rückwärtige Traversion des Automaten von t in Richtung des Startzustandes erreichbar sind und N die Menge aller Nachfolgertransitionen, die durch Traversion des Automaten in Richtung Endzustände erreichbar sind, so ergibt sich der genannte Mittelwert der Verhältnisse, den wir im Folgenden als $E(|I'_t|/A(I'_t)) \in (0,1]$ notieren, im Kontext einer Transition t durch:

$$E\Big(\frac{|I'_t|}{A(I'_t)}\Big) = \frac{\sum_{v \in V} |I'_r| + |I'_t| + \sum_{n \in N} |I'_n|}{\sum_{v \in V} A(I'_v) + A(I'_t) + \sum_{n \in N} A(I'_n)} \ , \tag{4.3}$$

wobei $v \in V$ und $n \in N$ jeweils eine Vorgänger- oder Nachfolgertransition notiert und I'_v und I'_n die entsprechend zugrundeliegende Menge an Instanzteilsequenzen. Für den relativ einfachen Ausschnitt des Präfixautomaten aus Abb. 4.3 ergibt sich bei Betrachtung der Transition mit den Token {1000,1099,1000} (ohne Berücksichtigung nicht dargestellter Transitionen) ein $E(|I'_t|/A(I'_t)) = (1+1+2+1+3)/(3+3+3+3+3) = 8/15$.

Wie bereits erläutert, steigt, wenn eine präfix-artige Bildungsvorschrift vorliegt, das Verhältnis $|I'_t|/A(I'_t)$, je weiter eine Transition t vom Startzustand entfernt ist. Da nun immer alle Pfade von t zu allen möglichen Endzuständen bei der Berechnung von $E(|I'_t|/A(I'_t))$ betrachtet werden, ergibt sich im Falle einer komplexen präfix-artigen Syntax der erwünschte Effekt, dass je weiter entfernt eine Transition t vom Startzustand liegt (und dementsprechend weniger Endzustände betrachtet werden), desto geringer ist das Verhältnis $E(|I'_t|/A(I'_t))$. Beide Verhältnisse

hängen demnach von der Position und dem Kontext der betrachteten Transition im Präfixautomaten ab und präferieren zueinander ins Verhältnis gesetzt feingranulare Muster am Beginn der Beispielinstanzen, wenn eine präfix-artige Bildungsvorschrift vorliegt.

Im Falle, dass für eine Transition $|I'_t|/A(I'_t) \leq E(|I'_t|/A(I'_t))$ gilt, soll die Klassifikationsmarke κ_t der Transition t durch I'_t ersetzt werden. Nutzern soll die Möglichkeit gegeben werden, die Berücksichtigung von feingranularen Mustern substanziell zu beeinflussen. Hierzu führen wir einen konfigurierbaren Schwellwert $\beta \in [-1, 1]$ ein und definieren die zweite Ungleichung für die Selektion von feingranularen Mustern als:

$$\frac{|I'_t|}{A(I'_t)} + \beta \leq E\Big(\frac{|I'_t|}{A(I'_t)}\Big). \tag{4.4}$$

Wenn ein $\beta = 1$ gewählt wird, dann wird selbst bei $|I'_t|/A(I'_t) \approx 0$ die Auswahl als feingranulares Muster unterdrückt. Im Falle, dass ein $\beta = -1$ konfiguriert wird, werden für alle Transitionen die Instanzteilsequenzen als feingranulares Muster klassifiziert, eine Generalisierung also komplett unterbunden. Beides sind Konfigurationen mit geringer praktischer Bedeutung. Wir berücksichtigen im Folgenden zwei Standardkonfigurationen: $\beta = -0,1$ für sehr genaue Extraktionsregeln und $\beta = 0,1$ für eine hohe Vollständigkeit bei der Extraktion.

Wenn die Ungleichungen 4.2 und 4.4 erfüllt sind, wird die Klassifikationsmarke κ_t der Transition t durch die Alternation der individuellen Instanzteilsequenzen I'_t ersetzt. In Abb. 4.4 ist ein Beispiel mit $\beta = -0,1$ als Forstsetzung des Beispiels aus Abb. 4.3 dargestellt. Modifizierte Transitionsmarken sind hervorgehoben. Die Notation „(X|Y)“ beschreibt die Alternation von Instanzteilsequenzen.

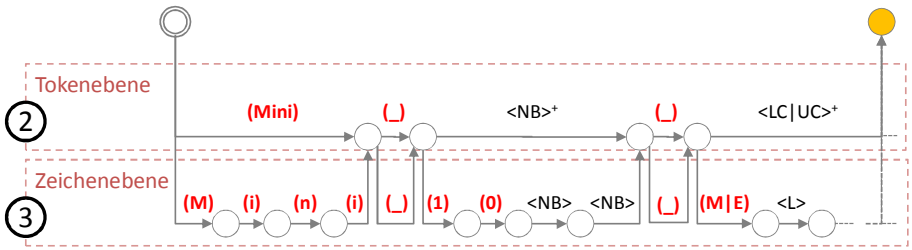


Abb. 4.4: Auswahl von feingranularen Mustern

In Abb. 4.3 ist weiterhin zu sehen, dass auch Leer- und Sonderzeichen bei der Bewertung berücksichtigt werden. Dieser Umstand hat den erwünschten Nebeneffekt, dass je mehr Leer- oder Sonderzeichen in den Beispielsequenzen auftreten, desto geringer ist die Tendenz, feingranulare Muster für andere Zeichen oder Token zu berücksichtigen.

Der Grund dafür ist, dass Leer- oder Sonderzeichen aneinander ausgerichtet werden und damit an einer solchen Position zumeist ein Verhältnis von $|I'_t|/A(I'_t) = 1/A(I'_t)$ vorherrscht. Somit fällt der Wert von $E(|I'_t|/A(I'_t))$ für andere Transitionen im selben Pfad geringer aus. Daraus folgt wiederum, dass je mehr Token die Beispielinstanzen enthalten, also umso spezifischer ihre Struktur als solche ist, desto weniger oft werden feingranulare Muster ausgewählt.

Der Automat ist nach wie vor nicht deterministisch, da parallele Musterbeschreibungen auf verschiedenen Abstraktionsebenen vorhanden sind, und kann nicht ohne weiteres in deterministisch Reguläre Ausdrücke, die von Nutzern einfach verstanden und modifiziert werden können, transformiert werden. Eine Selektion von am besten geeigneten Merkmalsbeschreibungen für Teilabschnitte des Automaten ist demnach notwendig, um z. B. zu entscheiden, ob $\langle \text{NB}^+ \rangle$ oder $(1)(0)\langle \text{NB} \rangle \langle \text{NB} \rangle$ zur Beschreibung der Instanzteilsequenzen $\{1000, 1099, 1000\}$ aus Abb. 4.4 geeigneter ist.

Bewertung von Merkmalen verschiedener Abstraktionsebenen

Im nächsten Verarbeitungsschritt werden die verschiedenen Abstraktionsebenen für Teilabschnitte des Automaten hinsichtlich ihrer Komplexität und Güte mit dem Ziel, die signifikantesten Muster zu selektieren, bewertet. Es wird versucht, das menschliche Vorgehen bei der Mustererkennung zu imitieren. Bei der manuellen Entwicklung von Regulären Ausdrücken anhand gegebener Beispielinstanzen verfolgen Nutzer im Allgemeinen den Ansatz durch eine möglichst geringe Anzahl an Symbolen, die Instanzen so genau wie möglich zu umschreiben.

Das Prinzip der *Minimum Description Length (MDL)* [Grü04] formalisiert eine solche Heuristik. Das MDL-Prinzip ist eng verwandt mit der Kolmogorow-Komplexität [LV97], die als Maß für die Strukturiertheit einer Zeichenkette die Länge des kürzesten Programms, das diese Zeichenkette erzeugt, vorschlägt. Andere Gebiete, in denen das MDL-Prinzip erfolgreich genutzt wurde, sind z. B. das Erzeugen von Entscheidungsbäumen [MP94] oder die Inferenz XML-Schemata von gegebenen Beispielen [GGR$^+$00].

Minimum Description Length. Die Idee von MDL kann wie folgt beschrieben werden. Gegeben sei eine Menge von Daten D, die durch ein endliches Alphabet beschrieben werden kann. Regelmäßigkeiten der Daten können verwendet werden, um die Daten zu komprimieren. Das MDL-Prinzip beschreibt den Umstand: umso besser ein Modell M die Regelmäßigkeiten der Daten D abbildet, desto geringer ist der Speicherbedarf der Daten D bei einer Kompression auf Basis des Modells M. Verschiedene Modelle können durch die Berechnung des Speicherbedarfs verglichen werden.

Das MDL-Prinzip schafft im Allgemeinen einen Ausgleich zwischen der Tauglichkeit und Komplexität bei der Auswahl aus alternativen Modellen, so dass eine über- oder untermäßige Generierung auf natürliche Weise verhindert wird. Formal kann die Kodelänge („*Description Length*"), auf der das MDL-Prinzip basiert, wie folgt ausgedrückt werden:

Definition 4.1.3 (*Kodelänge*)

Sei $\mathcal{L}(M)$ der Speicherbedarf, der benötigt wird, um ein Modell M zu beschreiben und $\mathcal{L}(D|M)$ der Speicherbedarf, der notwendig ist, um die gegebenen Daten D mit M zu kodieren, dann ist die Kodelänge $\mathcal{L}(D, M)$ definiert als: $\mathcal{L}(D, M) = \mathcal{L}(M) + \mathcal{L}(D|M)$.

Im Allgemeinen werden in der Literatur im Kontext des MDL-Prinzips zwei Varianten zur Bestimmung von Kodelängen beschrieben. Einige Ansätze gehen von einer verlustbehafteten Kompression aus und berücksichtigen eine Art Kompressionsfehler bei der Berechnung von $\mathcal{L}(D|M)$ (z. B. [RH01, YZHL05]). Unser Ansatz adaptiert die Ideen von MDL-Ansätzen, die von einer verlustfreien Kompression ausgehen (z. B. [GGR$^+$00]), in die hier dargestellte Problemstellung.

Adaption des Prinzips der Minimum Description Length. Die verschiedenen Modelle sind in der hier beschriebenen Problemstellung durch Kombinationen von Transitionen der verschiedenen Abstraktionsebenen zur Kodierung einer Menge von Beipielinstanzen I gegeben. Die Kodelänge für das Modell $\mathcal{L}(M)$ beschreibt den Speicherbedarf für die Symbole zur Kodierung von Transitionen und Transitionsmarken. Die Kodelänge für Daten $\mathcal{L}(D|M)$ beschreibt die Kompression, die bei der Kodierung der Daten auf Basis einer gegebenen Menge an Transitionen erreicht werden kann.

Im Folgenden notiere T eine beliebige Menge an Transitionen, wobei jede Tranisition $t \in T$ durch ihre Position im Automaten und die Transitionsmarke charakterisiert wird, und I_T die zu T zugehörige Menge von Instanzteilsequenzen. Die Problemstellung reduziert sich darauf, eine geeignete Kodelängendefinition $\mathcal{L}(I_T, T)$ für eine beliebige Transitionsmenge T zu definieren und potenzielle Kombinationen von Tranistionen verschiedener Abstraktionsebenen zu vergleichen.

Die Kodelängendefinition $\mathcal{L}(I_T, T)$ sollte folgende Heuristik abbilden: Ein hohes Abstraktionsniveau sollte ohne Berücksichtigung von feingranularen Mustern zu einem geringen $\mathcal{L}(T)$ und ein komplexeres Modell auf geringerer Abstraktionsebene zu einem vergleichsweise hohem $\mathcal{L}(T)$ führen. Hingegen sollte $\mathcal{L}(I_T|T)$ höhere Werte für höhere Abstraktionsebenen und geringere Werte für niedrigere Abstraktionsebenen annehmen. Die Präsenz von feingranularen Mustern in Transitionsmarken verschiebt bildlich gesprochen einen Teil der Daten in das Modell und erhöht den Speicherbedarf für das Modell $\mathcal{L}(T)$, aber verbessert die Kompression der Instanzteilsequenzen und verringert damit $\mathcal{L}(I_T|T)$.

Kodelänge für das Modell. Im Allgemeinen werden zur Datenkompression detektierte Muster in den Daten durch „Symbole“ repräsentiert und in einer Art Wörterbuch gespeichert. Ein wichtiger Bestandteil der Kodelänge $\mathcal{L}(T)$ ist daher das „Wörterbuch“, das zur Kompression verwendet wird.

Zur Bewertung der Kodelänge für die „Symbole“ ergeben sich zwei Einflussfaktoren: der Speicherbedarf für die Merkmale (Instanzteilsequenzen bei feingranularen Mustern bzw. Zeichen- und Tokenklassen) und der Speicherplatz, der für die Kodewörter der Symbole benötigt wird, die für die Kompression der Daten zur Verfügung stehen. Bevor auf die Bewertung eingegangen wird, soll der Term „Symbol“ in diesem Kontext definiert werden.

Sei T eine Menge von zusammenhängenden Transitionen, die das Modell zur Kodierung darstellen, I_T die Menge der zugehörigen zu kodierenden Beispielinstanzen und I'_t die Menge an Instanzteilsequenzen, die durch eine Transition $t \in T$ abgebildet wird. Es notiere weiterhin $S(t)$ die Menge der Symbole einer Transition t und $S(T) = \bigcup_{t \in T} S(t)$ die Menge aller Symbole für T, so sind zwei Fälle zu berücksichtigen: Zum einen wird, wenn eine Transition durch eine Klassifikationsmarke repräsentiert wird ($t \equiv \kappa_t$), genau ein Symbol im Modell berücksichtigt ($S(t) \leftarrow \kappa_t$). Zum anderen werden, wenn ein feingranulares Muster durch die Transition repräsentiert wird ($t \equiv I'_t$), alle individuellen Instanzteilsequenzen jeweils als Symbol im Modell dargestellt (also pro Instanzteilsequenz ein Symbol: $S(t) \leftarrow I'_t$).

Eine optimale Entropiekompression angenommen, kann der Speicherbedarf für Symbole $S(T)$ durch die Kodewortlänge $H(s) = -\log_2 p(s), s \in S(T)$, die auf der Auftrittswahrscheinlichkeit $p(s)$ eines Symbols $s \in S(T)$ basiert, multipliziert mit der Merkmalslänge, die im Folgenden als $L(s), s \in S(T)$ notiert wird, approximiert werden.

Zur Berechnung der Merkmalslänge $L(s)$ für feingranulare Muster gehen wir davon aus, dass pro Schriftzeichen einer Instanzteilsequenz genau ein Byte (= 2^3 bit) benötigt wird. Der Speicherbedarf berechnet sich demnach anhand der Anzahl der Schriftzeichen in der entsprechenden Instanzteilsequenz: $L(i'_t) = |i'_t| \cdot 2^3 \text{ bit}, i'_t \in S(T), t \in T$. Für Klassifikationsmarken wird, um eine Vergleichbarkeit zu gewährleisten, der Speicherbedarf von einem Schriftzeichen angenommen: $L(\kappa_t) = 2^3 \text{ bit}, \kappa_t \in S(T), t \in T$.

Die Anzahl der Bits, die zur Speicherung der Symbole $S(T)$ im „Wörterbuch“ benötigt wird, kann demnach zusammenfassend aus der Kodewortlänge $H(s), s \in S(T)$ und der Merkmalslänge $L(s), s \in S(T)$ wie folgt approximiert werden:

$$\mathcal{L}(T) = \sum_{s \in S(T)} H(s) + L(s). \tag{4.5}$$

Bei der Gegenüberstellung von möglichen Transitionskombinationen für das Beispiel aus Abb. 4.4 ergibt sich z. B. für die Kodierung der dargestellten Instanzteilsequenzen $I_T = \{\text{Mini\_1000\_Mi}, \text{Mini\_1099\_Eg}, \text{Mini\_1000\_Mobile}\}$ auf Basis der Transitionen auf Tokenebene mit $S(T) = \{(\text{Mini}), (\_), \langle \text{NB} \rangle^+, \langle \text{LC|UC} \rangle^+\}$ ein wesentlich geringerer Speicherbedarf für das Modell, als z. B. für die Kombination von Transitionen aus Zeichen- und Tokenebene mit $S(T) = \{(\text{Mini}), (\_), (1), (0), \langle \text{NB} \rangle, (\text{M}), (\text{E}), \langle \text{LC} \rangle\}$. Die letztere Variante verspricht jedoch eine höhere Genauigkeit. Die höhere Komplexität von Tranistionen auf Zeichenebene kann sich unter Umständen durch eine bessere „Kompression“ der Daten amortisieren, die im Folgenden erläutert wird.

Kodelänge für Daten. Die Kodelänge für Daten $\mathcal{L}(I_T|T)$ definiert hier den Speicherbedarf, der zur verlustfreien Kompression der Daten I_T mit dem Modell T notwendig ist und erlaubt damit eine Beurteilung der Güte des Modells.

Wenn eine Menge von Transitionen auf Zeichenebene wie im Beispiel aus Abb. 4.4 existiert, die den syntaktischen Aufbau der Instanzteilsequenzen sehr gut beschreibt, soll sich eine geringere Kodelänge im Vergleich zur Kodierung auf Tokenebene ergeben. Im Falle, dass feingranulare Muster in den Daten vorhanden sind, soll sich eine geringe Kodelänge für $\mathcal{L}(I_T|T)$ im Vergleich zur Kodierung mit Klassifikationsmarken ergeben. Weiterhin ist anzumerken, dass feingranulare Muster in Beispielentitäten sich mit steigendem β zuerst auf Zeichenebene „durchsetzen", bevor sie auf Tokenebene berücksichtigt werden, wodurch die „Kompressionsrate" auf Zeichenebene häufig höher ausfällt.

Zur Kompression von Instanzteilsequenzen I'_t für eine Transition t ist, im Falle von feingranularen Mustern ($t \equiv I'_t$), lediglich die Kodewortlänge für jedes einzelne Auftreten einer Instanzteilsequenz zu berücksichtigen. Sei $A(i'_t)$ die absolute Anzahl des Auftretens einer Instanzteilsequenz $i'_t \in I'_t$ in den ursprünglichen Instanzteilsequenzen und $H(i'_t)$ die Kodewortlänge des entsprechenden Symbols im Modell (siehe oben), so ergibt sich der Speicherbedarf für $t \equiv I'_t$ durch:

$$L(I'_t|t) = \sum_{i'_t \in I'_t} A(i'_t) \cdot H(i'_t), \quad t \equiv I'_t, i'_t \in S(T). \tag{4.6}$$

Wenn eine Tranistion t mit Klassifikationsmarke κ_t vorliegt, ist jede einzelne Instanzteilsequenz unabhängig vom Modell zu kodieren, da das Modell keine geeigneten Muster zur verlustfreien Kompression der Daten beschreibt. Die Unbestimmtheit des Modells kann durch die Annahme, dass in den zugrunde liegenden Instanzteilsequenzen I'_t keine Duplikate auftreten, ausgedrückt werden und entspricht damit für $t \equiv \kappa_t$:

$$L(I'_t|t) = \sum_{i'_t \in I'_t} A(i'_t) \cdot L(i'_t) \cdot 2^3 \text{ bit}, \quad t \equiv \kappa_t, \kappa_t \in S(T), \tag{4.7}$$

wobei $L(i'_t)$ die Länge einer Instanzteilsequenz in Zeichen und $A(i'_t)$ deren absolute Anzahl notiert.

Gleichung 4.6 und 4.7 definieren die Anzahl der Bits, die zur Kompression der Daten I'_t mit einer Transition t notwendig sind. Für eine zusammenhängende Menge von Transitionen T und zugehörige Instanzteilsequenzen I_T ergibt sich unter Berücksichtigung von Symbolen zur Beschreibung des sequentiellen Aufbaus von T ein Speicherbedarf von:

$$\mathcal{L}(I_T|T) = (|T| - 1) \cdot 2^3 \text{ bit} + \sum_{t \in T, I'_t \in I_T} \mathcal{L}(I'_t|t), \tag{4.8}$$

wobei $(|T| - 1) \cdot 2^3$ bit den Speicherbedarf zur Kodierung der Transitionssequenz und $\mathcal{L}(I'_t|t)$ die oben diskutierten Kodelängen für Transitionen mit feingranularen Mustern und Klassifikationsmarken beschreibt.

Kombinierte Kodelänge für Transitionen. Der Speicherbedarf bei der Kodierung der gegebenen Menge von Beispielentitäten I auf Basis einer möglichen Kombination von Transitionen T ergibt sich unter Zusammenfassung der Gleichungen 4.5 und 4.8 durch:

$$\mathcal{L}(I, T) = \mathcal{L}(T) + \mathcal{L}(I|T). \tag{4.9}$$

Es handelt sich dabei um eine Abschätzung des Speicherbedarfs, um die Eignung der Merkmale, die durch die Transitionen der verschiedenen Abtraktionsebenen repräsentiert werden, vergleichen zu können.

Selektion von Merkmalen zur Erzeugung von Regulären Ausdrücken

Für die Selektion von Merkmalen werden alle möglichen zulässigen Kombinationen von Transitionssequenzen getestet, die die gegebenen Beispielentitäten I beschreiben. Der Anzahl der Transitionskombinationen ist dabei überschaubar, da lediglich für jede Transition auf Tokenebene eine alternative Beschreibung auf Zeichenebene in Frage kommt. Sei $\mathcal{T}$ die Menge aller möglichen Transitionskombinationen und $T_{Tok} \in \mathcal{T}$ die Menge von Transitionen auf Tokenebene, so ergibt sich die Anzahl an zu überprüfenden Kombinationen durch $1 + \sum_{k=1}^{|T_{Tok}|} k = 1 + (|T_{Tok}| \cdot (|T_{Tok}| + 1))/2$.

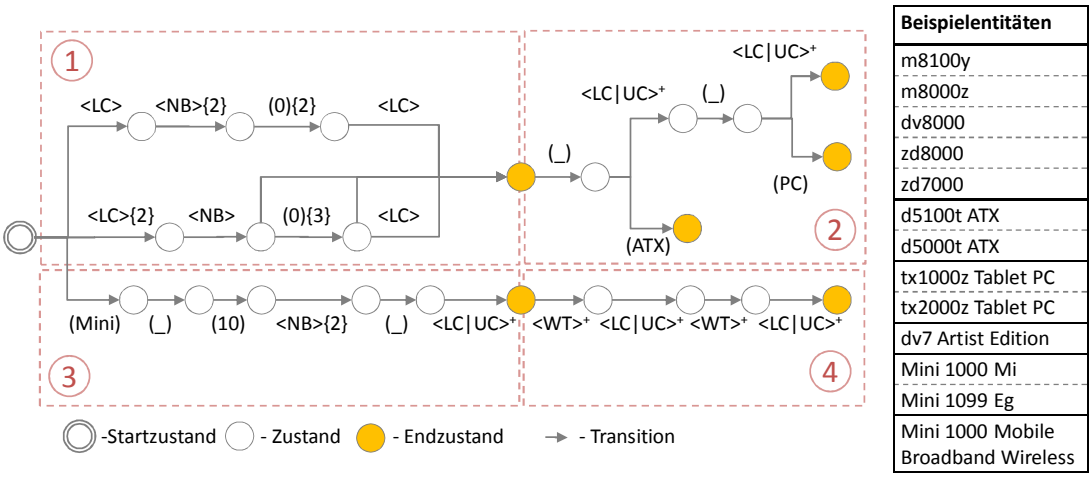
Die minimale Kodelänge, die sich durch die verschiedenen Transitionskombinationen in $\mathcal{T}$ ergibt, kann durch:

$$\mathcal{L}_{min}(I, \mathcal{T}) = \min_{(T \in \mathcal{T})} \left[\mathcal{L}(T) + \mathcal{L}(I|T) \right] \tag{4.10}$$

ausgedrückt werden. Es wird die Kombination von Transitionen $T \in \mathcal{T}$ zur Beschreibung der Beispielinstanzen I ausgewählt, die $\mathcal{L}_{min}(I, \mathcal{T})$ erzeugt.

Als nächster Schritt zur Transformation des NEA in einen DEA werden die Ausdrücke im Automaten vereinfacht. Sequenzen gleicher Transitionsmarken werden zusammengefasst und mit den entsprechenden Kardinalitäten gekennzeichnet (notiert als $\{x\}$). Wenn Abfolgen von feingranularen Mustern auftreten, die jeweils nur einen Token oder ein Zeichen enthalten, werden diese ebenfalls zur besseren Lesbarkeit zusammengeführt.

In Abb. 4.5 ist der finale Präfixautomat für das laufende Beispiel dargestellt. Es resultiert ein DEA, der die wesentlichsten Muster der Beispielinstanzen beschreibt, die rechts in Abb. 4.5 zu sehen sind. Die Einteilung der Beispielentitäten entspricht dabei der initialen Partitionierung aus Tab. 4.1.



Beispielentitäten
m8100y
m8000z
dv8000
zd8000
zd7000
d5100t ATX
d5000t ATX
tx1000z Tablet PC
tx2000z Tablet PC
dv7 Artist Edition
Mini 1000 Mi
Mini 1099 Eg
Mini 1000 Mobile Broadband Wireless

Abb. 4.5: Finaler Deterministischer Automat zur Musterbeschreibung

In Abb. 4.5.(1) ist ein Block von Merkmalssequenzen dargestellt, in dem eine Menge von Transitionen auf Zeichenebene, die teilweise feingranulare Muster und Klassifikationsmarken enthalten, ausgewählt wurde. Die Muster in Abb. 4.5.(2) werden auf Tokenebene am besten repräsentiert. Zum Teil wurden signifikante Token am Ende der Beispielinstanzen erkannt.

Der Bereich in Abb. 4.5.(3) beschreibt Muster auf Basis einer Kombination von Transitionen auf Token- und Zeichenebene. In der letzten Transition von Abb. 4.5.(3) wurde die Klassifikationsmarke auf Tokenebene: $\langle$LC|UC$\rangle^+$ anstatt der Alternative auf Zeichenebene: $(M|E)\langle LC\rangle\{2,6\}$ ausgewählt (vgl. Abb. 4.4), da in diesem Fall (im Gegensatz zu dem Bereich in Abb. 4.5.(1))

die Komplexität des Ausdrucks auf Zeichenebene in keinem günstigen Verhältnis zur erzielten Kompressionsrate steht.

Für den Bereich in Abb. 4.5.(4) lag pro Transition lediglich eine Instanzteilsequenz vor. Da vorherige Transitionen bereits wertvolle Musterbeschreibungen enthalten, wird in diesem Bereich der höchste Abstraktionsgrad ausgewählt. Anzumerken ist, dass in Fällen, in denen eine Beispielinstanz keinerlei Gemeinsamkeiten mit anderen Instanzen hat, durch die Bedingung für feingranulare Muster in Ungleichung 4.4 sichergestellt wird, dass keinerlei Abstraktion stattfindet.

Zur Überführung des DEA in Reguläre Ausdrücke werden zur Reduktion der Komplexität und besseren Verständlichkeit mehrere Reguläre Ausdrücke erzeugt. Pro Endzustand im DEA wird ein Regulärer Ausdruck durch rückwärtige Traversion der Transitionen in Richtung Startzustand erzeugt. Marken der Transitionssequenzen werden durch deren Konkatenation im erzeugten Regulären Ausdruck repräsentiert. Einzelne Transitionssequenzen von komplexen Teilautomaten auf Zeichenebene (siehe z. B. Abb. 4.5.(1)) werden durch deren Alternation in die Regulären Ausdrücke abgebildet. Für Abb. 4.5.(1) ergibt sich z. B. der Ausdruck $(\langle \text{LC} \rangle \langle \text{NB} \rangle \{2\} (0) \{2\} \langle \text{LC} \rangle | \langle \text{LC} \rangle \{2\} \langle \text{NB} \rangle (0) \{3\} ? \langle \text{LC} \rangle ?)$, in dem zur weiteren Vereinfachung optionale Teilausdrücke, die sich aus der Struktur des Automaten ergeben, entsprechend markiert wurden.

In Kap. 4.1.1 wurde erläutert, dass jeweils ein Präfix- und ein Sufixautomat generiert wird. Zur Vereinigung der Ergebnisse von Präfix- und Suffixautomat werden für jeden Regulären Ausdruck die Referenzen zu den Beispielinstanzen, die ihm zugrunde liegen, und die Kodelänge $\mathcal{L}(I_T, T)$ für die ursprünglich zum Ausdruck gehörigen Transitionen T gespeichert. Für jede Beispielinstanz werden die Kodelängen der Regulären Ausdrücke, die aus Präfix- und Suffixbaum entstanden sind, verglichen (siehe Alg. 3.(16) – Alg. 3.(25)). Es wird der Reguläre Ausdruck mit der geringsten Kodelänge (wenn noch nicht vorhanden) in den finalen Regelsatz übernommen.

Als Ergebnis steht eine Menge von wenig komplexen Regulären Ausdrücken zur Verfügung, die durch Experten einfach verstanden und modifiziert werden kann. Durch eine entsprechende Konfiguration von Zeichenklassen können Reguläre Ausdrücke für beliebige Regel-basierte Extraktionssysteme erlernt werden, die eine Negation zur Beschreibung von Tokenklassen unterstützen. Wenn eine Negation nicht unterstützt wird oder nicht die in Kap. 4.1.1 vorgestellte Art und Weise zur Erzeugung von Tokenklassen verwendet werden soll, können diese manuell konfiguriert werden. Es ist weiterhin eine Erweiterung um höherwertige Abstraktionsebenen auf Basis von Tokenklassen für zukünftige Anwendungsfälle vorstellbar.

4.1.3 Zusammenfassung

In Kap. 4.1 wurde eine Methode beschrieben, die die Inferenz von Regulären Ausdrücken aus einer Menge von Beipielentitäten unterstützt. Bei der Regelinferenz werden für die Instanzmenge ein Präfix- und Suffixautomat erzeugt, die von der Syntax der Entitäten auf Basis von Zeichen- und Tokenklassen abstrahieren.

Zur Selektion von feingranularen Mustern (fixen Zeichen oder Token) wurde das Verhältnis von individueller zu absoluter Anzahl von Instanzteilsequenzen im Vergleich zu dem entsprechenden mittleren Verhältnis in Vorgänger und Nachfolgertranistionen vorgeschlagen. Die Selektion von am besten geeigneten Abstraktionsebenen zur Repräsentation von Merkmalen basiert auf dem Prinzip der „*Minimum Description Length*“. Die Übersetzung des Automaten in mehrere Reguläre Ausdrücke stellt eine Menge von Regeln zu Verfügung, die in beliebigen Regel-basierten Extraktionssystemen verwendet werden können.

Im Falle, dass Beispielinstanzen (wie z. B. Rechnungsnummern oder Telefonnummern) einer klaren Bildungsvorschrift folgen, sind relativ wenige Beispiele für die Regelinferenz notwendig. Wenn komplexere Bildungsvorschriften (wie im Fall der diskutierten Notebookbezeichner) vorliegen, wird eine entsprechend größere Anzahl an Beispielen zur Inferenz von charakteristischen Mustern benötigt. Es ist anzumerken, dass das hier vorgestellte Verfahren für

Anwendungsfälle, in denen die Instanzen extrem heterogen sind, also quasi keiner Bildungsvorschrift folgen, und extrem große Beispieldatensätze, z. B. zum Erlernen von charakteristischen Token (wie z. B. übliche Vor- und Nachnamen von Personen) verwendet werden müssen, nicht ohne weiteres geeignet ist.

Die durch das vorgestellte Verfahren erzeugten Regulären Ausdrücke schaffen (in Abhängigkeit von zwei Parametern) einen Kompromiss zwischen den Zielkriterien: Vollständigkeit und Genauigkeit. Es werden deterministische Regeln alleine basierend auf Charakteristiken der vorgegebenen positiven Beispielinstanzen ohne die Notwendigkeit eines annotierten Trainingskorpus erlernt. Wir untersuchen die Extraktionsqualität in Abhängigkeit von der Anzahl an Beispielentitäten in Kap. 4.2 und vergleichen die Ergebnisse zu einem Statistischen Maschinellen Lernverfahren, das mit einem Trainingskorpus trainiert wird und Kontextinformationen berücksichtigt.

Experten können die durch das hier vorgestellte Verfahren erzeugten Regulären Ausdrücke ohne weiteres modifizieren. Die einfachste Modifikationsmethode stellt das Pflegen von Negativlisten dar, die im Allgemeinen ein hohes Verbesserungspotenzial verspricht. Die automatische Optimierung von Regulären Ausdrücken mit Hilfe eines Trainingskorpus wurde in [LKR+08] z. B. durch automatische Erzeugung von Negativlisten und Transformation von Zeichenklassen entlang einer vordefinierten Hierarchie vorgeschlagen. Weiterhin ist es möglich, die erzeugten Regeln, so denn ein großer Trainingskorpus zur Verfügung steht, als Merkmale in Statistischen Maschinellen Lernverfahren einzusetzen, wodurch z. B. approximierte Genauigkeiten einzelner Regeln oder deren Kombinationen bestimmt werden können [MKHV09].

4.2 Experimentelle Evaluation

Die Güte erlernter Regulärer Ausdrücke kann grundsätzlich durch eine Äquivalenz- bzw. Ähnlichkeitsprüfung zu manuell erstellten Regulären Ausdrücken evaluiert werden. Eine empirische Evaluation mit Hilfe eines annotierten Dokumentenkorpus ist jedoch besser geeignet, um den praktischen Nutzen des Verfahrens darzustellen.

Der Überblick über verwandte Arbeiten in Kap. 2.3 zeigte, dass ein direkt vergleichbares Verfahren, welches Reguläre Ausdrücke oder andere deterministische Regeln für eine gegebene Liste an Beispielentitäten erzeugt, im Forschungsbereich der Informationsextraktion bisher nicht beschrieben wurde. Insbesondere unterstützen bisherige Verfahren zur Regelinferenz keine abhängigen Merkmale unterschiedlicher Granularität.

Die Möglichkeit der Kombination von abhängigen Merkmalen unterschiedlicher Granularität ist ein fundamentaler Vorteil von Statistischen Maschinellen Lernverfahren wie *Conditional Random Fields (CRF)* [LMP01] gegenüber bisherigen Regellernverfahren in der Informationsextraktion. Um eine quantitative Aussage für die Effektivität der hier beschriebenen Methode zu erhalten, vergleichen wir unser System mit einem CRF-Ansatz, der im Unterschied zu dem hier vorgestellten Verfahren mit annotierten Beispieldokumenten trainiert wird. Als Vergleichskriterium werden die Resultate, die mit einem exakten Wörterbuchvergleich erreicht werden können, gezeigt.

Der Vergleich zu einem *CRF*-Modell soll darstellen, ob das hier vorgestellte Verfahren eine ähnliche Extraktionsqualität für Entitäten, die bestimmten Bildungsvorschriften folgen, aufweist, obwohl es lediglich Beispiellisten für Entitäten benötigt und von Experten einfach zu modifizierende Reguläre Ausdrücke erzeugt.

Die einzelnen Verfahren werden mit der selben Teilmenge der Beispielentitäten eines annotierten Dokumentenkorpus trainiert. Das CRF verwendet immer alle Dokumente für das Training, wobei Entitäten und der Kontext von Entitäten, die nicht für das Training ausgewählt wurden, während der Lernphase ausgeblendet werden. Das hier vorgestellte Verfahren benötigt für die Inferenz von Bildungsvorschriften lediglich die Beispielentitäten. In der Beschreibung der Datensätze in Kap. 4.2.1 wird darauf eingegangen, ob es sich bei den verwendeten Beispielen-

titäten um Bezeichner handelt, die in gleicher Form in strukturierten Referenzdaten auftreten, oder ob sie in die Kategorie „einfach zu beschaffende Beispielentitäten“ eingeordnet werden können.

Es werden Experimente für die Verwendung von 1%, 2%, 4%, ..., 64% und 100% der Beispielentitäten betrachtet, da insbesondere Anwendungsfälle, in denen wenige Trainingsdaten verfügbar sind, interessant sind. In mehreren Durchläufen eines Experiments wurden zufällig Beispielentitäten ausgewählt, die Resultate für alle Verfahren mit diesen Beispieldaten ermittelt und zum Abschluss das arithmetische Mittel der Ergebnisse berechnet.

Das CRF wurde mit einer häufig eingesetzten Merkmalskombination[3] konfiguriert, die mit den Merkmalen, die in Kap. 4.1 beschrieben wurden, vergleichbar sind: Wörter (Token), der vorherige und nachfolgende Kontext im Abstand von bis zu zwei Token, N-Gramme mit einer maximalen Länge von 4 sowie deren Sequenz und der Wortaufbau, welcher mit den Tokenklassen aus Kap. 4.2.1 vergleichbar ist (Sonderzeichen werden jeweils einzeln betrachtet).

Das in dieser Arbeit beschriebene Verfahren wurde in zwei Versionen als Modul für die *RapidUIM*-Plattform zur Verfügung gestellt. Das Ziel dieser alternativen Implementationen war es, nachzuweisen, dass das Verfahren mit verschiedenen Regel-basierten Systemen eingesetzt werden kann. Die erste Version basiert auf CGUL, der Regelbeschreibungssprache von *BusinessObjects Text Analysis*[4] (basierend auf [Hul00]), die zweite Variante auf der Java-API für Reguläre Ausdrücke. Beide Implementationen erzeugen äquivalente Ergebnisse. Für die hier vorgestellten Experimente wurde das Modul, das auf der Java-API basiert, verwendet. Als CRF-Verfahren wurde die Implementation des *Stanford NLP Toolkits* [FGM05] verwendet.

Wir betrachten vier verschiedene Szenarien und acht verschiedene Typen von Entitäten. Im ersten Anwendungsfall wird das Szenario I aus Kap. 1.2.1 aufgegriffen. Es wird die Effektivität hinsichtlich der Extraktion von Rechnungsnummern, Telefonnummern und SWIFT-Codes dargestellt. Zur Beurteilung der Qualität bei der Extraktion von Produktnamen werden Bezeichner von Notebooks aus HTML-Seiten extrahiert.

Weiterhin wird Szenario II aus Kap. 1.2.2 aufgegriffen und die Qualität bei der Extraktion von Java-Fehlermeldungen diskutiert. Als letzter Anwendungsfall wird ein Dokumentenkorpus, der durch die Verfasser von [LKR+08] zur Verfügung gestellt wurde, zur Evaluation herangezogen. Wir beurteilen die Extraktion von Lehrveranstaltungsbezeichnern aus HTML-Seiten, die als besondere Art von Produktnamen im Kontext der Enterprise Search betrachtet werden können. Weiterhin gehen wir auf die Extraktion von Softwarebezeichnern und die Extraktion von Telefonnummern aus diesem Korpus ein.

Evaluationsmetriken und verwendete Datensätze werden in Kap. 4.2.1 eingeführt. In Kap. 4.2.2 stellen wir die Ergebnisse der Experimente vor und werten diese aus.

4.2.1 Evaluationsmetriken und Datensätze

Bevor wir auf die verwendeten Datensätze eingehen, betrachten wir die Evaluationsmetriken *Genauigkeit* (*Precision*), *Vollständigkeit* (*Recall*) und *F-Measure*. Die Metriken werden wie in der Serie der *Conference on Natural Language Learning (CoNLL)* [TKSDM03] verwendet, wobei nur eine exakt extrahierte Entität als korrekt gewertet wird und z. B. nicht, ob der Typ innerhalb der Satzgrenzen korrekt bestimmt wurde [GS96].

Die drei relevanten Mengen zur Beurteilung der Ergebnisse sind: *Wahr Positiv* (WP) – korrekt extrahierte Entitäten, *Falsch Positiv* (FP) – fehlerhaft extrahierte Entitäten und *Falsch Negativ* (FN) – nicht extrahierte Entitäten. Die *Vollständigkeit* R und *Genauigkeit* P sind wie folgt definiert:

[3] `http://nlp.stanford.edu/software/crf-faq.shtml`, Stand: 01.07.2010

[4] `http://www.sap.com/solutions/sapbusinessobjects/large/eim/textanalysis/index.epx`, Stand: 01.08.2010

$$R = \frac{|WP|}{|WP \cup FN|}, \quad P = \frac{|WP|}{|WP \cup FP|} \tag{4.11}$$

Im Folgenden gehen wir lediglich auf das harmonische Mittel aus Genauigkeit und Vollständigkeit, dem so genannten *F-Measure* ein, welches als $F = (2 \cdot P \cdot F)/(P + F)$ definiert ist. Eine detaillierte Auflistung der Einzelergebnisse für Genauigkeit und Vollständigkeit und Beispiele für erzeugte Reguläre Ausdrücke sind im Anhang A.3 zu finden. Im Folgenden werden die (manuell annotierten) Dokumentenkorpora, die in der Evaluation verwendet wurden, vorgestellt. Beispieldokumente für den jeweiligen Korpus sind in Anhang A.2 dargestellt.

Dokumentenkorpus „Rechnungen". Zum Test von SAP-Systemen werden Anwendungsszenarien zusammengestellt, die Daten und Dokumente von realen Kunden widerspiegeln. Dabei kann zwischen Daten unterschieden werden, die von automatischen Tests erzeugt werden und realen Daten zum Testen von User-Interface und Usability. Zur Evaluation des Szenarios I aus Kap. 1.2.1 wurde ein realer Rechnungskorpus akquiriert, der für die Tests des in Kap. 1.2.1 durch Abb. 1.4 dargestellten SRM-Systems verwendet wurde.

Nach einer Deduplizierung standen 103 Dokumente zur Verfügung. Es handelt sich um durch eine OCR-Software digitalisierte Rechnungen von Unternehmen aus verschiedensten Bereichen wie dem Automobilhandel, Hotelgewerbe, IT-Bereich etc. Ein Beispieldokument ist im Anhang A.2 dargestellt (siehe Bsp. A.4 auf S. 146). Zusätzliche strukturierte Daten, wie Rechnungsnummern, für die keine Dokumente vorliegen, werden nicht berücksichtigt.

Für die Evaluation wurden die in Tab. 4.2 dargestellten Entitätstypen: Swift-Code, Rechnungsnummer und Telefonnummer herangezogen. In Tab. 4.2 sind weiterhin Beispiele für Entitäten und „schwierige Fälle" (Negativbeispiele) mit Erklärung (insofern notwendig) dargestellt.

Tab. 4.2: Entitäten im Dokumentenkorpus „Rechnungen"

Entitätstyp	Beispiel	Negativbeispiel
Swift-Code	SMBCJPJT CRESCHZZ80A GENODE TDOMCATT	ADDRESSES CH20080606 (Kostenstelle) TRANSFER REMITTANCE
Rechnungsnummer	11-093 #545465 2008-035465 0909.08338	001-412 (Teil von Telefonnummer) #708274774 (Kundennummer) 2008-16 (Teil von Datum) 2008.09 (Teil von Datum)
Telefonnummer	650-429-2500 0531/ 593333 +49 171/ 3363540 (0621)71846	0210-0002-1 (Wohnungsnummer) 040/ 116 (Teil von Steuernummer) +27.06.08 (Datum: OCR-Fehler) (60620)0 (Datum: OCR-Fehler)

Die syntaktischen Muster von SWIFT-Codes sind relativ einfach (Großbuchstaben und Zahlen, variable Länge). Das Muster ist einfach herzuleiten, jedoch wenig diskriminativ. Im vorgestellten Dokumentenkorpus existieren eine Vielzahl von anderen Wörtern, die wie SWIFT-Codes durchgängig groß geschrieben sind und zusätzlich Zahlen enthalten. Wir merken an, dass SWIFT-Codes in strukturierten Daten meist exakt so erfasst werden, wie sie in Rechnungen auftreten. Das in Kap. 4.1 vorgestellte Verfahren könnte also auch auf Basis von (historischen) Beispielentiäten eines RDBMS trainiert werden.

Rechnungsnummern (siehe Tab. 4.2 für Beispiele) sind im Vergleich zu SWIFT-Codes heterogener strukturiert. Auch wenn die Anzahl der Besipielentitäten im Dokumentenkorpus relativ klein ist, stehen für die verschiedenen syntaktisch ähnlich aufgebauten Entitäten mehrere Beispiele zur Verfügung. Die syntaktische Struktur ist im Vergleich zu SWIFT-Codes relativ diskriminativ. Der Aufbau der Rechnungsnummern ähnelt dennoch anderen Konzepten im Trainingskorpus, wie in Tab. 4.2 zu sehen ist, wodurch eine präzise Extraktion erschwert wird.

Rechnungsnummern werden wie SWIFT-Codes zumeist in Datenbanksystemen so erfasst, wie sie in den Dokumenten vorkommen.

Telefonnummern werden in Dokumenten zur besseren Lesbarkeit im Allgemeinen anders dargestellt als in strukturierten Daten. Es handelt sich in diesem Szenario um einen Anwendungsfall, in dem Beispielentitäten aus Texten akquiriert werden müssen. Der Aufbau von Telefonnummern ist komplexer als der von vorherigen Entitätstypen, so dass die syntaktischen Muster relativ eindeutig sind. Es existieren jedoch wiederum eine Vielzahl von Negativbeispielen mit ähnlicher Struktur.

Dokumentenkorpus „Notebookbewertungen". Zur Evaluation der Extraktion von Produktbezeichnungen wurde ein Dokumentenkorpus für Notebookbezeichner zusammengestellt. Die Dokumente entstammen einer Bewertungsplattform für Produkte[5] im WWW. 90 Dokumente, die jeweils eine Liste von 20 Notebookbezeichner enthalten, standen zur Evaluation zur Verfügung. Der tabellenartige Aufbau der Dokumente ähnelt z. B. Rechnungen, die in HTML-Form verschickt werden. In den Navigationsmenüs der Seite sind zusätzliche Notebookbezeichner hinterlegt, so dass der gesamte HTML-Quelltext für die Extraktion verwendet wurde.

Im Anhang A.2 ist ein Beispieldokument des Korpus abgebildet (siehe Abb. A.1 auf S. 147). In Tab. 4.3 sind in Ergänzung zu dem in Kap. 4.1 diskutierten Beispielen weitere Entitäten samt Negativbeispielen dargestellt.

Tab. 4.3: Entitäten im Dokumentenkorpus „Notebookbewertungen"

Beispiel	Negativbeispiel
HDX18t	HBX2 (in eingebetteten Java Script)
A945US	UA283S (in eingebetteten Java Script)
Mini Note	Mini Tower (PC-Typ)
Touchsmart tx2z	Touchscreen PC 19 (PC-Typ mit Monitorgröße)

Dokumentenkorpus „SAP Community Network". Zur Evaluation des Szenarios II aus Kap. 1.2.2, in dem die Extraktion von Java-Fehlermeldungen notwendig ist, wurde ein Dokumentenkorpus von 1.500 Foreneinträgen des SCN-Forums verwendet. Die Dokumente wurden mit Hilfe von manuell erstellten Regulären Ausdrücken für Java-Sprachkonstrukte ausgewählt. Im Anhang A.2 ist ein Beispiel für einen Foreneintrag dargestellt (siehe Abb. A.2 auf S. 147).

In Tab. 4.4 sind Beispiele für Entitäten und „schwierige Fälle" dargestellt. Das größte Problem ist in diesem Anwendungsfall die Unterscheidung zwischen normalem Java-Quelltext und Java-Fehlermeldungen. Insbesondere ist hier das Suffix der Beispielentitäten relevant („Exception", "Error" etc.), da die Präfixe normalen Java-Sprachkonstrukten entsprechen. Für diesen recht speziellen Anwendungsfall wurde weiterhin die Tokenisierung der verwendeten Verfahren dahingehend modifiziert, dass der Übergang von Klein- zu Großschreibung eine Tokengrenze definiert.

Tab. 4.4: Entitäten im Anwendungsfall „SAP Developer Network"

Beispiel	Negativbeispiel
OutOfMemoryError	Ec.SetError (Variablenaufruf)
java.lang.NoClassDefFoundError	AddoCompany.GetLastError (kein Java/ Methode)
System.IO.FileNotFoundException	Net.WebException (kein Java)
java.sql.SQLException	Manager.raiseException (Methodenenaufruf)
com.sap.aii.[..].ConfigException	com.sap.hcm.[..].SteamCourse (Java-Klasse)

[5] `http://www.notebookreview.com`, Stand: 01.08.2009

Dokumentenkorpus „ReLIE". Der Dokumentenkorpus der durch die Autoren von [LKR+08] („Regular Expression Learning for Information Extraction"- ReLIE) zur Verfügung gestellt wurde, umfasst ca. 80.000 Textfragemente. Der Ursprungskorpus sind nach [LKR+08] 50.000 Dokumente aus dem Intranet eines Unternehmens, 50.000 Webseiten der Universität Michigan und 10.000 Emails des Enron-Datensatzes. Jedes Textfragment beschreibt den Kontext im Abstand von 200 Buchstaben, um das mögliche Auftreten einer Entität vom Typ Telefonnummer, Lehrveranstaltung oder Softwarebezeichnung, die mit Regulären Ausdrücken, die eine hohe Vollständigkeit erzielen, selektiert wurden. Ein Beispiel für Textfragemente des Korpus ist auf S. 148 in Bsp. A.5 dargestellt.

Textfragmente überlappen sich nicht. Es wurde jeweils das erste mögliche Auftreten einer Entität in einem Textfragment des Korpus als korrekt oder falsch annotiert. Für die Evaluation im Kontext dieser Arbeit wurde zufällig eine Teilmenge der Textfragmente (ca. 1.500 pro Entitätstyp) ausgewählt und die Annotationen von Entitäten manuell vervollständigt (zusätzlich zum ersten Auftreten). Beispiele für zu extrahierende Entitätstypen und „schwierige Fälle" sind in Tab. 4.5 dargestellt.

Tab. 4.5: Entitäten im Dokumentenkorpus „ReLIE"

Entitätstyp	Beispiel	Negativbeispiel
Telefonnummer	734-936-6395 (313) 593-5131 847.494.5626 212/949-8058	2002-11-15 (Datum) (1863-1865) (Jahreszahlen) 141.211.90 (IP-Adresse) 1511/1695 (Seitenreferenz in Buch)
Lehrveranstaltung	CompCourse 006 Nursing 03 AOSS 585 SPANISH 436	WordPerfect 9 (Software) Winter 05 (Semester) ISSN 003 (Teil von Buchreferenz) VOTETEXT 1052 (in HTML-Navigation)
Softwarebezeichnung	Fireworks 4.0 Microsoft Frontpage 2000 Fireworks MX Microsoft Word 97	Html 3.0 (Standard) Fiscal Year 2000 (Zeitspanne) Building II (Ort) Pitcher Place 97 (Adresse)

Telefonnumern sind in diesem Korpus anders formatiert als in dem Rechnungskorpus. Die Bezeichner von Lehrveranstaltungwn sind recht einfach strukturiert, wodurch deren Extraktion erschwert wird. Softwarebezeichnungen, die mit einer Versionsnummer der Art „2.0" gekennzeichnet sind, können häufig sehr genau extrahiert werden. Andere Merkmale sind jedoch nicht sehr diskriminativ und überschneiden sich mit vielen anderen Konzepten im Text. Die Problemstellung, eine gute Genauigkeit in Kombination mit einer akzeptablen Vollständigkeit zu erzielen, gestaltet sich in diesem Anwendungsfall besonders schwierig.

Anzumerken ist, dass ein direkter Vergleich zu dem Verfahren, das in [LKR+08] vorgestellt wurde, nicht möglich ist, da es von Nutzern vorgeschlagene Reguläre Ausdrücke hinsichtlich eines Trainingskorpus optimiert. Das hier vorgestellte Verfahren und der Ansatz von [LKR+08] sind daher eher als komplementär zu betrachten.

4.2.2 Ergebnisse und Auswertung

In Abb. 4.6 ist das *F-Measure* für alle evaluierten Verfahren in den einzelnen Anwendungsfällen dargestellt (siehe Anhang A.3 für Details zu Vollständigkeit und Genauigkeit sowie für Beispielregeln). Es werden Experimente für die zwei Standardkonfigurationen unseres Verfahrens gezeigt: die Konfiguration mit $\beta = -0,1$ führt im Allgemeinen zu einer hohen Genauigkeit und eine Verwendung von $\beta = 0,1$ zu einer hohen Vollständigkeit bei der Extraktion. Erstere Konfiguration wird in Abb. 4.6 als *PAT-.1* und zweitere mit *PAT+.1* dargestellt. Das Wörterbuchbasierte Verfahren, das Textfragmente und Beipielentiäten über einen exakten Zeichenkettenvergleich aufeinander abbildet, wird durch *DICT* illustriert. Das Vergleichsverfahren *Conditional Random Fields* ist als *CRF* gekennzeichnet. Die Abszisse der Grafiken in Abb. 4.6 stellt den prozentualen Anteil der für das Training verwendeten Beispielentitäten dar.

Das in Kap. 4.1 vorgestellte Verfahren kann als effektiv bezeichnet werden, wenn es ähnliche Ergebnisse wie *CRF* erzielt, da *CRF* zusätzlich Kontextdaten und Negativbeispiele betrachtet. Von vergleichbaren Ergebnissen wird bei einem maximalen Unterschied von 0,15 im F-Measure ausgegangen. *CRF* liefert im Allgemeinen eine hohe Genauigkeit. Wir beginnen daher die Auswertung mit dem Vergleich von *CRF* und *PAT-.1*.

In den zwei Anwendungsfällen zur Extraktion von Telefonnummern (siehe Abb. 4.6(a) und Abb. 4.6(f)) sowie bei der Extraktion von Lehrveranstaltungen (siehe Abb. 4.6(g)) sind die Ergebnisse von *PAT-.1* bei wenigen Trainingsdaten wesentlich besser als die von *CRF*. In Abb. 4.6(a) beträgt die Differenz 0,24 und in Abb. 4.6(f) 0,34 bei 4% der Trainingsdaten. Mit zunehmenden Trainingsdaten stellt sich *CRF* auf den entsprechenden Korpus ein und liefert zu *PAT-.1* vergleichbare Ergebnisse.

Im Anwendungsfall der Lehrveranstaltungen in Abb. 4.6(g) produziert *CRF* schlechte Ergebnisse mit der gegebenen Merkmalskombination. Nachforschungen ergaben, dass der Klassifikator ähnliche Wahrscheinlichkeiten für eine Klassifikation als Entität und als Nicht-Entität liefert, was auf die heterogene Merkmalsverteilung im Korpus zurückzuführen ist. Die Verwendung anwendungsspezifischer Merkmalskombinationen verspricht hier eine Verbesserung. Durch Variation der hier verwendeten Merkmale (z. B. der N-Gramm-Länge) und deren Rekombination (z. B. nur Token und Wortaufbau) konnten die Ergebnisse jedoch nicht verbessert werden. *PAT-.1* liefert im Gegensatz zu *CRF* für das Szenario der Extraktion von Lehrveransaltungen sehr gute Ergebnisse, so dass bereits bei Verwendung von 2% der Trainingsdaten ein F-Measure von 0,4 vorliegt.

In zwei Fällen sind die Ergebnisse von *PAT-.1* und *CRF* ähnlich und im Vergleich zu *DICT* sehr gut: Bei der Extraktion von Rechnungsnummern (siehe Abb. 4.6(c)) erzielen *CRF* und *PAT-.1* wesentlich bessere Ergebnisse im Vergleich zu *DICT* (maximaler Unterschied von 0,34 für *CRF* und 0,44 für *PAT-.1* bei 4% der Trainingsdaten), abstrahieren also sehr gut von den konkreten Instanzen. Die Resultate von *PAT-.1* übertreffen die Ergebnisse von *CRF* um 0,02 bis 0,14. Bei der Extraktion von Notebookbezeichner beträgt die Differenz von *CRF* und *PAT-.1* zwischen 0,02 und 0,07. Die Extraktionsqualität beider Verfahren ist in den beschriebenen Experimenten also vergleichbar.

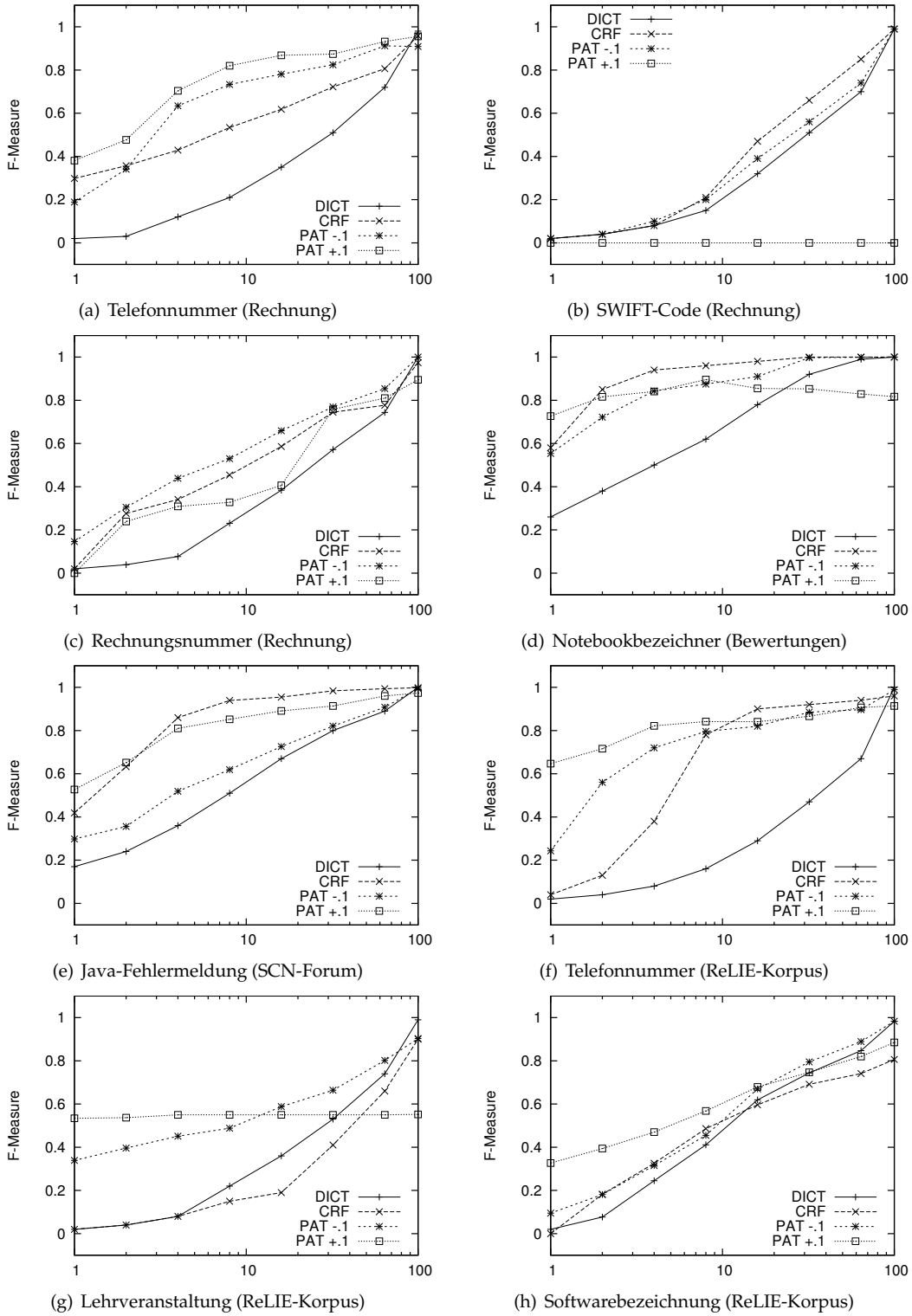
In zwei Fällen (Abb. 4.6(b) und Abb. 4.6(h)) ist die Extraktionsqualität von *CRF* und *PAT-.1* bei der Verwendung von wenigen Trainingsdaten im Vergleich zu *DICT* nicht so hoch, wie in vorherigen Szenarien: Bei der Extraktion von SWIFT-Codes (siehe Abb. 4.6(b)) beträgt der Unterschied zu *DICT* bei 4% der Trainingsdaten für beide Verfahren ca. 0,1. Die wenig signifikante Struktur von SWIFT-Codes erschwert deren Extraktion enorm (siehe auch detaillierte Ergebnisse zur Vollständigkeit in Abb. A.4 auf S. 148). Ab einem Training mit mehr als 8% der Annotationen übertrifft *CRF* die Ergebnisse *PAT-.1* um bis zu 0,1 durch die zusätzliche Ausnutzung der Kontextinformationen. Der maximale Unterschied von *CRF* zu *DICT* bei der Extraktion von Nootebookbezeichnern (siehe Abb. 4.6(h)) liegt bei 0,08 (bei 8% Trainingsdaten) und der von *PAT-.1* zu *DICT* bei 0,1 (bei 1% Trainingsdaten).

In Abb. 4.6(h) kann wie in Abb. 4.6(g) beobachtet werden, dass die Parameter des *CRF* nicht optimal bestimmt werden können, so dass die Extraktionsqualität z. B. bei Verwendung von 64% der Trainingsdaten wesentlich geringer ist, als z. B. von *PAT-.1*. Diese Problematik führt im Falle des *CRF* dazu, dass bei 100% der Trainingsdaten aufgrund einer geringen Vollständigkeit (siehe Abb. A.10 auf S. 154) lediglich eine F-Measure von 0,81 erreicht wird.

Im Falle von Java-Fehlermeldungen (siehe Abb. 4.6(e)) liefert *PAT-.1* deutlich schlechtere Ergebnisse als *CRF* (0,34 Unterschied bei 4% Trainingsdaten), da eine nicht zufriedenstellende Vollständigkeit vorliegt (siehe auch Abb. A.7 auf S. 152). Der Grund dafür ist, dass der CRF-Ansatz die sich wiederholende Sequenz von Token und Punkten innerhalb der Java-Fehlermeldungen implizit abbildet, da es immer den Kontext von aktuell beobachteten Token zur Bewertung heranzieht.

Die Konfiguration *PAT+.1* liefert jedoch für Java-Fehlermeldungen ähnliche Ergebnisse wie *CRF*. In den Anwendungsfällen, in denen Telefonnummern extrahiert werden (siehe

Abb. 4.6(a) und Abb. 4.6(f)) und bei der Extraktion von Softwarebezeichnungen werden sogar die besten Ergebnisse mit der Konfiguration *PAT+.1* erzielt, da der syntaktische Aufbau der Entitäten auch mit wenigen berücksichtigten fixen Zeichenketten sehr diskriminativ ist.



(a) Telefonnummer (Rechnung)

(b) SWIFT-Code (Rechnung)

(c) Rechnungsnummer (Rechnung)

(d) Notebookbezeichner (Bewertungen)

(e) Java-Fehlermeldung (SCN-Forum)

(f) Telefonnummer (ReLIE-Korpus)

(g) Lehrveranstaltung (ReLIE-Korpus)

(h) Softwarebezeichnung (ReLIE-Korpus)

Abb. 4.6: Extraktionsqualität (F-Measure) in Abhängigkeit von den Trainingsdaten [%]

In den anderen Anwendungsfällen neigt *PAT+.1* zur übermäßigen Verallgemeinerung. Das führt dazu, dass selbst bei 100% der Trainingsdaten oft nicht annähernd eine 100%ige Genauigkeit erreicht wird. *PAT+.1* liefert aber insbesondere bei extrem wenigen Trainingsdaten eine hohe Vollständigkeit. Die Graphen für *PAT+.1* sind daher in Abb. 4.6 häufig relativ flach (siehe insbesondere Abb. 4.6(g)). Bei der Extraktion von Swift-Codes erzeugt *PAT+.1* z. B. einen Regulären Ausdruck, der alle durchgängig groß geschriebenen Wörter beliebiger Länge als Entität klassifiziert (siehe Beispiele für Reguläre Ausdrücke in Tab. A.2 im Anhang A.3 auf S. 150). Die Genauigkeit dieses Ausdrucks geht gegen 0, wogegen die Vollständigkeit bereits bei Verwendung von 1% der Trainingsdaten 1 ist (siehe Abb. A.4 auf S. 149).

Die Versuchsergebnisse zeigen, dass das in Kap. 4.1 vorgestellte Verfahren für die hier betrachtete Klasse von Entitäten und der Konfiguration *PAT-.1* eine ähnliche gute Extraktionsqualität bietet wie CRF, obwohl weder der Entitätenkontext, noch Negativbeispiele, sondern lediglich Beispielentitäten für das Training notwendig sind. In vielen Anwendungsfällen übertrifft unsere Methode (mit $\beta = -0,1$ oder $\beta = 0,1$) das CRF-Verfahren, welches mit einer vergleichbaren Merkmalskombination initialisiert wurde, im Kontext von wenigen Trainingsdaten um wesentlich mehr als $0,15$ im F-Measure. Dieses Verhalten ist insofern nicht überraschend, da das vorgestellte Verfahren explizit für Entitäten, die einer Bildungsvorschrift folgen, entwickelt wurde.

Insbesondere bietet das in dieser Arbeit entwickelte Verfahren den Vorteil, dass einfach zu verstehende sowie leicht modifizierbare Reguläre Ausdrücke erzeugt werden, so dass die Extraktionsmechanismen einfach nachvollziehbar und Fehler leichter entdeckt und beseitigt werden können. Eine manuelle Feinabstimmung der Regulären Ausdrücke verspricht eine leichter zu erzielende Qualitätssteigerung gegenüber Statistischen Maschinellen Lernverfahren im praktischen Einsatz.

4.3 Verwandte Arbeiten in der Mustererinferenz

In Kap. 2.3.1 wurde bereits eine grobe Einschätzung von bekannten Ansätzen hinsichtlich ihrer Anwendbarkeit zur Inferenz von Bildungsvorschriften für Entitäten gegeben. Im Folgenden gehen wir auf die Gemeinsamkeiten und Unterschiede des in Kap. 4.1 vorgestellten Ansatzes zu ausgewählten, verwandten Verfahren aus unterschiedlichen Bereichen vom Blickwinkel der algorithmischen Herangehensweise ein.

Zu Beginn werden Ansätze im Bereich der Datenintegration und -bereinigung diskutiert (siehe Kap. 4.3.1). Im Speziellen gehen wir auf die Umwandlung von unstrukturierten in strukturierte Datensätze ein – ein häufiges Problem im Bereich der Datenintegration. Die Problemstellung ist dahingehend verwandt, als dass verschiedene Attribute in unstrukturierten Datensätzen jeweils einem bekannten Attributtyp (z. B. auf Basis von Bildungsvorschriften) zugewiesen werden müssen. Im Kontext dieser Betrachtungen greifen wir noch einmal Ansätze aus dem Bereich der Statistischen Maschinellen Lernverfahren auf. Weiterhin diskutieren wir einen Ansatz zur Musterinferenz, der zur Beurteilung von fehlerhaften Datenbanktupeln im Kontext der Datenbereinigung entwickelt wurde.

Auf eine gesonderte Diskussion von Statistischen Maschinellen Lernverfahren wird an dieser Stelle verzichtet, da diese in Kap. 2.3.1 hinreichend abgegrenzt wurden. Sie unterscheiden sich zu dem hier vorgestellten Verfahren insbesondere dadurch, dass sie Negativbeispiele für das Training benötigen. Wir betrachten im Weiteren Ansätze, die wie das in Kap. 4.1 vorgestellte Verfahren Regeln anhand von Beispielinstanzen erlernen.

In Kap. 4.3.2 werden Arbeiten aus zwei anwendungsorientierten Forschungsbereichen im Kontext der grammatikalischen Inferenz von positiven Beispielinstanzen diskutiert. Wir greifen Algorithmen, die in der Inferenz von DTD-Schemata verwendet werden und wie das in dieser Arbeit vorgeschlagene Verfahren Reguläre Ausdrücke erzeugen, auf und betrachten die Mustererkennung im Bereich der Wrapper-Induktion.

Abschließend betrachten wir Ansätze zur Musterinferenz, die im Bereich von Regellernverfahren in der Informationsextraktion eingesetzt werden. In Kap. 2.3.1 wurden Arbeiten aus diesem Bereich bereits dahingehend abgegrenzt, dass sie annotierte Dokumentenkorpora zum Training benötigen und lediglich Merkmale einer Tokenebene unterstützen. Mit einer detaillierten Betrachtung der verwendeten Algorithmen und Bewertungsverfahren in Kap. 4.3.3 bekräftigen wir diese Abgrenzung.

4.3.1 Verwandte Arbeiten im Bereich der Datenintegration und -bereinigung

Im Bereich der Datenintegration [BN08] steht die Integration von heterogenen Datenbeständen im Mittelpunkt. Zur Bestimmung von Korrespondenzen zwischen verschiedenen heterogenen strukturierten Datenbestände ist in bestimmten Anwendungsfällen eine syntaktische Analyse von Instanzdaten hilfreich [NHT+02]. Es handelt sich dabei jedoch um eine entfernt verwandte Problemstellung, da Merkmale von strukturierten Daten verglichen werden.

Im Kontext dieser Arbeit sind Ansätze, die im Bereich des Parsens von heterogenen Einzeldatensätzen mit verschiedenen Attributtypen, wie z. B. textuelle Adress- oder Logdaten entwickelt wurden, relevanter. Anzumerken ist, dass derartige Methoden in der Literatur häufig im Bereich der Informationsextraktion eingeordnet werden, aber gerade im Bereich der Datenintegration in vielen Anwendungsfällen von besonderer Bedeutung sind [EIV07].

Die Problemstellung der Überführung von unstrukturierten in strukturierte Datensätze lässt sich sehr gut mit Statistischen Maschinellen Lernmethoden wie *Hidden Markov Models (HMM)* lösen (siehe z. B. [BDS01]), wenn annotierte Beispieldatensätze vorhanden sind. [MS06] stellen eine Erweiterung von [BDS01] auf Basis von *Conditional Random Fields (CRF)* vor, die zusätzlich Referenzdaten eines RDBMS für das Training verwendet. Ein Training ohne Dokumentenkorpus ist (im Gegensatz zur hier entwickelten Methode) jedoch nicht möglich.

Die Referenzdaten des RDBMS dienen in [MS06] unter anderem als „Merkmalsquelle“. Eine Reihe von häufigen Tokenklassen wie $[A-Z]\backslash.$ oder $[A-Z][a-z]^+$ für Personennamen werden vorkonfiguriert und jeder Token in jedem Tupel des RDBMS auf Übereinstimmung überprüft. Konkatenierte Merkmalsbeschreibungen wie z. B. $[A-Z].[A-Z][a-z]^+$ für „*R. Stevenson*“ werden als zusätzliche Merkmale beim Training des CRF hinzugezogen. Das in Kap. 4.1 vorgestellte Verfahren kann im Kontext des Ansatzes von [MS06] detailliertere Merkmalsbeschreibungen generieren, die weit über einfache konkatenierte Tokenmuster hinausgehen.

Ein Verfahren, das ähnlich wie der hier vorgestellte Ansatz für Entitäten entwickelt wurde, die einer klaren Bildungsvorschrift folgen, ist [AHS+09]. Es werden Zeichen-N-Gramm-Signaturen zur Klassifikation von Attributtypen in unstrukturierten Datensätzen vorgeschlagen. Eine generelle Kritik an der Verwendung von Zeichen-N-Grammen ist, dass keinerlei Abstraktion von den konkreten Instanzdaten vorgenommen wird, wie dies in der hier vorgestellten Methode der Fall ist. Da nicht signifikante N-Gramme der Instanzen in Signaturen einfach ausgeblendet werden, erfordert eine Klassifikation von Attributtypen mit [AHS+09] (im Gegensatz zu unserer Methode), dass die Grenzen von Attributen in den Textdaten bekannt sind.

Im Bereich der Datenbereinigung [RD00] ist die Inferenz von Bildungsvorschriften hilfreich, wenn Anomalien in einer gegebenen Menge von textuellen Instanzwerten gesucht sind. Im Fall, dass ein Datentupel nicht der oder den erwarteten Bildungsvorschriften folgt, ist es wahrscheinlich, dass es sich um einen fehlerhaften Wert handelt. „Potters Wheel“ [RH01] verwendet z. B. nutzerdefinierte Domänen wie Zahl („⟨number⟩“), Zeit („⟨time⟩“) oder Wort („⟨word⟩“) zur Inferenz von genau einer Bildungsvorschrift für eine Datenbankspalte. Für eine Menge von Instanzen der Art „19 January 06:45“ wird z. B. die Bildungsvorschrift „⟨number⟩⟨word⟩⟨time⟩“ erlernt.

Merkmale auf Zeichenebene werden in [RH01] nicht diskutiert und sind auch nicht notwendig, da eine abstrakte Musterumschreibung gesucht ist. Weiterhin sind feingranulare Muster aus

demselben Grund in [RH01] nicht von Bedeutung. Selbiges gilt für heterogene Instanzen, die verschiedenen Bildungsvorschriften folgen, wie sie im Kontext der Inferenz von Bildungsvorschriften für die Informationsextraktion relevant sind. Da Instanzbeispiele im Anwendungsgebiet von [RH01] relativ homogen strukturiert sind und Abweichungen gesucht sind, ist die Erzeugung von genau einer Bildungsvorschrift in diesem Kontext sinnvoll.

Wie auch das in dieser Arbeit vorgestellte Verfahren, basiert der Ansatz von [RH01] auf dem Prinzip der *Minimum Description Length (MDL)*. Es gibt jedoch signifikante Unterschiede in der Verwendung des MDL-Prinzips, die auf die grundverschiedenen Anwendungsklassen zurückzuführen sind. So sind z. B. die Anzahl von definierten Domänen und die Wahrscheinlichkeit, dass Attributinstanzen durch eine untersuchte, potenzielle Bildungsvorschrift nicht abgedeckt werden, wichtige Einflussfaktoren in der Musterbewertung von [RH01]. Feingranulare Muster spielen, da sie nicht betrachtet werden, auch bei der Definition der *„Description Length"* keine Rolle. Es werden keine komplex strukturierten Musterbeschreibungen auf verschiedenen Abstraktionsebenen, sondern eher einfach strukturierte, sequentielle Konkatenationen von Domänen untersucht.

4.3.2 Inferenz von Regulären Grammatiken anhand positiver Beispielinstanzen

Für Problemfälle, in denen positive und negative Beispielinstanzen gegeben sind, sind stochastische Grammatiken (wie z. B. *Hidden Markov Models*) und entsprechende Lernverfahren geeignet [GT07]. Im Folgenden betrachten wir Ansätze im Bereich der Inferenz von regulären Grammatiken auf Basis positiver Beispielinstanzen in den Gebieten der Inferenz von DTD-Schemata (z. B. [GGR$^+$00, BNST06, HNW06, Fer09, BNSV10]) und der Web-Wrapper-Induktion (siehe z. B. [TAC06, CKGS06, BGG09]).

Grundlage vieler Ansätze [Fer09, GGR$^+$00, BNSV10] zur Inferenz von DTD-Schemata ist die Darstellung von XML-Elementen in Beispieldokumenten als Zeichenketten (z. B. *abab* für ⟨a⟩⟨/b⟩⟨/a⟩⟨a⟩⟨/b⟩⟨/a⟩). Auf Basis der Zeichenketten werden Reguläre Ausdrücke (z. B. $(ab)*$) abgeleitet, die wiederum in eine DTD-Schemabeschreibung überführt werden können.

Die Ansätze von [GGR$^+$00] und [BNSV10] adaptieren, wie das in dieser Arbeit entwickelte Verfahren, das MDL-Prinzip. In [GGR$^+$00] werden zur Beschreibung der Kodelänge des Modells für einen Ausdruck der Form $(a|b)^*$ die Anzahl der Struktursymbole im Regulären Ausdruck verwendet (4 im Fall von $(a|b)^*$ – Klammern werden nicht gezählt). Zur Berechnung der Kodelänge für Daten wird, wenn ein Kleene Konkatenation (*) vorliegt, die Anzahl der Wiederholungen, wie oft der Teilausdruck angewendet wird, und ein Einflussfaktor, der bei Alternationen das selektierte Symbol kennzeichnet, berücksichtigt. [BNSV10] verwendet das MDL-Prinzip in ähnlicher Weise, abstrahiert jedoch von der konkreten Regelbeschreibungssprache (siehe [AV06] für eine formale Diskussion).

Diese Art von MDL-Definition ist für die Bewertung von Sequenzmustern sehr gut geeignet, aber nicht für die Selektion von Merkmalen aus verschiedenen abhängigen Abstraktionsebenen, wie sie in dieser Arbeit diskutiert wurde.

Das Problem der Wrapper-Induktion [TAC06, CKGS06] ist ein verwandtes Anwendungsgebiet der grammatikalischen Inferenz [Sak97]. Dokumente können als Beispielinstanzen und das Ergebnis der Induktion als eine Bildungsvorschrift, die variable und fixe Bestandteile von Beipielinstanzen abbildet, angesehen werden. Wenn ein annotierter Dokumentenkorpus zum Training verwendet wird, sind im Unterschied zu dem in dieser Arbeit diskutierten Problem variable Bestandteile durch die Annotationen in den Trainingsdokumenten bekannt. Die algorithmische Herangehensweise ist in diesem Fall eher mit der Inferenz von DTD-Schemata vergleichbar, da Sequenzmuster wie Tabellenstrukturen, ineinander geschachtelte Dokumentstrukturen oder Permutationen von Inhalten, die in Dokumenten auftreten können, in eine Grammatik abzubilden sind.

Eine weitere Parallele zwischen dem Verfahren aus Kap. 4.1 und Ansätzen in der Wrapper-Induktion lässt sich zeichnen, wenn man bedenkt, dass auch in Dokumenten verschiedene Abstraktionsebenen zu berücksichtigen sind (z. B. der tabellarische Aufbau eines HTML-Dokuments und die Textstruktur innerhalb von HTML-Elementen). Lediglich [CL01, CK04, ZSWG09] unterstützen eine Inferenz für Muster auf verschiedenen Abstraktionsebenen, die jedoch vom Nutzer manuell angewiesen werden muss. Eine voll-automatische Selektion von geeigneten Abstraktionsebenen wird nicht diskutiert.

Es kann also festgestellt werden, dass es verschiedene Gemeinsamkeiten zwischen der in dieser Arbeit betrachteten Inferenz von Bildungsvorschriften und der Induktion von Web-Wrappern gibt. In den Prioritäten in den Inferenzschritten unterscheiden sie sich jedoch stark: Bei der Induktion von Wrappern liegt das Hauptproblem in dem Erkennen von Wiederholungen und anderen strukturellen Sequenzmustern. Die Berücksichtigung von verschiedenen Abstraktionsebenen wird in der Literatur diskutiert, kann aber manuell im Kontext einer Nutzerinteraktion geschehen. Für die Inferenz von Bildungsvorschriften für Entitäten ist die automatische Bewertung und Selektion aus verschiedenen Abstraktionsebenen essentiell, wogegen Wiederholungen oder Permutationen von Symbolfolgen von geringerer Bedeutung sind.

4.3.3 Regellernverfahren in der Informationsextraktion

In bestimmten Anwendungsfällen (in denen z. B. ein Nutzereingriff wünschenswert ist) sind Regellernverfahren eine Alternative zu Statistischen Maschinellen Lernverfahren [Sar08]. Die Voraussetzung für deren Einsatz ist die Verfügbarkeit von großen (annotierten) Dokumentkopora.

In Kap. 2.3.1 wurde bereits erläutert, dass traditionelle Regellernverfahren zur Inferenz von Bildungsvorschriften anhand von Beispielentitäten nicht ohne weiteres einsetzbar sind. Im Folgenden gehen wir auf die verwendeten Klassen von Lernverfahren ein und zeigen Ähnlichkeiten und Unterschiede zu dem in Kap. 4.1 diskutierten Ansatz auf.

Die in traditionellen Regellernverfahren verwendeten Regelsprachen orientieren sich an den Prinzipien der Aussagen- oder Prädikatenlogik [TAC06]. Die eingesetzten Lernverfahren orientieren sich entsprechend an Paradigmen zur automatischen Erzeugung von logischen Programmen in dem entsprechenden Gebiet [TAC06]: der Propositionalen Inferenz (Aussagenlogik) oder dem Relationalen Lernen (Prädikatenlogik).

Innerhalb beider Paradigmen können die Verfahren in *Top-Down*- und *Bottom-Up*-Verfahren eingeteilt werden [Sar08]. *Bottom-Up*-Verfahren erstellen für jedes Beispiel im Trainingskorpus eine Regel. Diese werden iterativ zusammengefasst und z. B. bezüglich ihrer Abdeckung und Genauigkeit bei der Extraktion aus dem Trainingskorpus bewertet. *Top-Down*-Verfahren beginnen mit der Erzeugung einer äußerst allgemeinen Regel für die Beispielinstanzen, die Schritt für Schritt durch Spezialisierung hinsichtlich der Vollständigkeit und Genauigkeit für den annotierten Dokumentenkorpus optimiert wird.

Im Folgenden gehen wir auf traditionelle Regellernverfahren ein und diskutieren die zugrunde liegenden algorithmischen Ideen, wobei wir uns an [TAC06, CKGS06, NS07, Sar08] orientieren. Im Anschluss diskutieren wir aktuellere Verfahren, die eine Art von Regulären Ausdrücken zur Extraktion von Beziehungen erlernen. Eine Bewertung und Abgrenzung wird am Ende des Kapitels vorgenommen.

Propositionale Inferenz. AutoSlog [Ril93], PALKA [KM95], CRYSTAL [Sod99] und TIMES [CBG99] sind Systembeispiele, die das Paradigma der Propositionalen Inferenz adaptieren. AutoSlog [Ril93], PALKA [KM95] und CRYSTAL [Sod99] erlernen Regeln auf Basis linguistischer Merkmale (grammatikalische Satzstrukturen) und erwarten daher linguistisch annotierte Fließtexte als Eingabe. Sie benötigen weiterhin ein manuell erstelltes *linguistisches Templates* wie „⟨Subjekt⟩ Verb(passiv)“ zur Merkmalsdefinition und erlernen konkrete Ausprägungen wie et-

wa: „⟨Subjekt⟩ Verb(passiv)=was acquired“ durch Spezialisierung des Templates anhand von Beispielsätzen.

AutoSlog [Ril93] unterstützt keinerlei Abtraktionsmechanismen. PALKA [KM95] erlaubt als Erweiterung semantische Annotationen für Entitäten und eine Generalisierung von semantischen Typen in Hierarchien (z. B. die Hierarchie {„Unternehmen“, “Universität“} < “Organisation“), so dass Regeln der Art „⟨Organisation⟩ Verb(passiv)=was acquired“ erlernt werden können, wenn in den Beispielsätzen speziellere Ausprägungen des Entitätstyps identifiziert wurden.

CRYSTAL [Sod99] verwendet zur Verallgemeinerung von linguistischen Regeln einen *Bottom-Up Covering*-Algorithmus für semantische Typen und morphologisch-syntaktische Merkmale. Ähnliche Einzelregeln werden zusammengeführt, wobei jeweils die Schnittmenge der Merkmale weiterverwendet wird. So denn eine erzeugte Kombinationsregel keine falschen Ergebnisse im Trainingskorpus erzeugt, werden die zugrunde liegenden Regeln verworfen, andernfalls werden alternative Regelkombinationen getestet.

Das TIMES-System [CBG99] verwendet WordNet-Synonyme und die WordNet-Typhierarchie (siehe [Mil95]) zur Verallgemeinerung. Regelkombinationen werden zufällig mit dem Ziel der Generalisierung gebildet und alle Kombinationsregeln gespeichert, die einen vordefinierten Schwellwert für die Genauigkeit bei der Extraktion aus dem Trainingsdatensatz überschreiten.

Relationales Lernen. Systeme, deren Regelsprachen Konzepte der Prädikatenlogik adaptieren, unterstützen die Inferenz von Beziehungen zwischen Merkmalen und benötigen daher keine nutzergenerierten *linguistischen Templates*. LIEP [Huf95] basiert wie die oben genannten Systeme auf grammatikalischen Satzstrukturen. Bei der Untersuchung einer noch nicht betrachteten Beispielannotation wird versucht, die aktuell beobachteten Token in bekannte Regeln zu integrieren, so dass die Anzahl der Regeln, z. B. im Vergleich zu AutoSlog [Ril93] minimiert wird. Ist dies nicht möglich, wird eine neue Extraktionsregel erzeugt.

RAPIER [CM99] oder $(LP)^2$ [Cir01] sind Beispiele für Systeme im Bereich des Relationalen Lernens, die eine „echte“ Verallgemeinerung von Regeln unterstützen. Sie verwenden einen *Bottom-Up Covering*-Algorithmus. Die Vorgehensweise unterscheidet sich dabei nicht wesentlich von dem Algorithmus, der in CRYSTAL [Sod99] verwendet wird (siehe oben).

SRV [Fre00] verwendet einen *Top-Down Covering*-Algorithmus. Es adaptiert ein Lernverfahren der *Induktiven Logischen Programmierung*– den *First Order Logic Induction*-Algorithmus (FOIL). Eine leere Regel wird Schritt für Schritt mit Merkmalen angereichert, solange bis alle Beispiele im Trainingskorpus extrahiert werden können. In jedem Schritt wird ein noch nicht berücksichtigtes Merkmal der Annotationen ausgewählt, das den maximalen Informationsgewinn verspricht (berechnet auf Basis der positiven und negativen Beispiele im Trainingskorpus).

Inferenz von Regulären Ausdrücken zur Extraktion von Beziehungen. Einige aktuellere Regellernverfahren erzeugen eine Art von Regulären Ausdrücken zur Extraktion von Beziehungen. Wir betrachten die verwendeten Lernverfahren im Folgenden.

Der Lernprozess von WHISK [Sod99] ist interaktiv: Dem Nutzer werden Dokumente zur manuellen Annotation vorgeschlagen. Er hat daher zu jedem Zeitpunkt die Möglichkeit, die Inferenz zu beeinflussen. Tokenklassen, wie *„(Number)“* werden ebenso wie Wörterbücher von dem Nutzer zur Annotationszeit festgelegt. Merkmale sind ebenso, wie in bisher beschriebenen Ansätzen, auf eine Ebene von Tokenrepräsentationen begrenzt. Die Regelinferenz in WHISK [Sod99] beginnt mit einer allgemeinen, leeren Regel, wie z. B. *„*(*)*(*)*“*, die alle bekannten positiven Beispiele abdeckt, wobei *„(*)“* einen Platzhalter für eine Entität und *„*“* einen Platzhalter für Kontextinformationen repräsentiert. Es wird ein *Top-Down Covering*-Algorithmus verwendet. In jedem Schritt wird vorgeschlagen, der aktuellen Regel das Merkmal hinzuzufügen, das das Verhältnis von fehlerhaft zu korrekt extrahierten Instanzen im Trainingskorpus minimiert, bis ein vorkonfigurierter Grenzwert des Verhältnisses erreicht wird.

SPIES [YZHL05] erlernt Regeln der Art „⟨***Protein***⟩⟨*Verb, 3. Person*⟩⟨*Präposition*⟩⟨***Protein***⟩“ zur Extraktion von Protein-Interaktionen aus bio-medizinischen Texten. Auf Basis von Annotationen im Traingkorpus werden zufällig Merkmale zu Regeln kombiniert und zu Regelsätzen zusammengefasst. Potenzielle Regelsätze, die in einem vorgegebenen Zeitfenster erzeugt wurden, werden anhand einer MDL-Definition bewertet. Die MDL-Definition von [YZHL05] berücksichtigt die Anzahl der verwendeten Merkmale und die Vollständigkeit und Genauigkeit bei Anwendung des Regelsatzes auf den Trainingskorpus.

AliBaba [PHL05, HSL07, NTL10] erlernt spezifische Ausprägungen von *Linguistischen Frames*, wie z. B. „⟨*Protein*⟩ Word* ⟨*Verb*⟩ Word* ⟨*Protein*⟩“ (siehe Kap. 2.3.1.1) und berücksichtigt Tokenmerkmale, wie z. B. morphologische Merkmale, charakteristische Wörter und die Wortanzahl für Platzhalter (z. B. für „Word*“ im Beispiel). Der Ablauf der Regelinferenz ist wie folgt: Zuerst werden positive Satzbeispiele aus einem Korpus extrahiert, diese werden auf ihren „Kern“ reduziert (z. B. nur der relevanten Worttypen zwischen Entitäten) und Merkmale auf verschiedenen Abstraktionsebenen von Token erfasst. Als Resultat steht eine Menge von initialen Regeln zur Verfügung, die Tokenabstraktionen verschiedener Ebenen berücksichtigt.

Im Weiteren werden die initialen Regeln zur Verallgemeinerung *geclustert* und die Regeln in einem *Cluster* zu generellen Regeln zusammengefasst, die die Tokenmerkmale der verschiedenen Abstraktionsebenen vereinigen. Zur Bestimmung der Wortanzahl für Platzhalter wird ein genetischer Optimierungsalgorithmus vorgeschlagen (siehe [PHL05]), wobei als Optimierungsziel entweder die Genauigkeit, die Vollständigkeit oder das F-Measure von potenziellen Regelsätzen herangezogen wird. In [NTL10] werden weitere Heuristiken diskutiert, die die Extraktionsqualität der erzeugten Regelsätze wesentlich verbessern. Beispiele für Heuristiken sind das Filtern von extrem komplexen Regeln oder das Entfernen von Regeln mit einer geringen Genauigkeit. Alle Abstraktionsebenen werden bei der Extraktion berücksichtigt und auf Basis von Statistiken des Trainingkorpus bewertet.

Untereinander abhängige syntaktische Merkmalstypen unterschiedlicher Granularität (z. B. Token- und Zeichklassen) zur Beschreibung von Entitäten werden in den oben diskutierten Ansätzen nicht betrachtet. Die Selektion von geeigneten Abstraktionsebenen für Merkmale von Entitäten beschränkt sich auf semantische Typhierarchien. Eine syntaktische Analyse von Entitäten wird lediglich auf Basis von Tokenmerkmalen unterstützt. Die Anwendung bisheriger Verfahren in den hier beschriebenen Szenarien ist daher nicht Erfolg versprechend, da komplexe syntaktische Merkmale (Bildungsvorschriften) für Entitäten nicht im vollen Umfang erlernt werden können.

Eine syntaktische Verallgemeinerung geschieht in bisherigen Ansätzen ausschließlich für Kontextmerkmale durch das Vereinigen von Merkmalen verschiedener Beispiele bzw. das Ausblenden von nicht signifikanten Merkmalen. Der in der vorliegenden Arbeit entwickelte Ansatz ist insofern als komplementär zu bisherigen Regellernverfahren zu betrachten, als dass sie auf die Musterinferenz zur Beschreibung von Kontextinformationen spezialisiert sind, wobei die Betrachtung genau einer Tokengranularität (meist Wörter) in vielen Anwendungsfällen völlig ausreichend ist.

Insofern im Kontext von Entitäten wertvolle Muster zu erwarten sind, ist eine Kombination von dem in Kap. 4.1 vorgestellten Verfahren mit traditionellen Regellernverfahren vorstellbar. Häufig ist jedoch der syntaktische Aufbau, bei den in dieser Arbeit betrachteten Typen von Entitäten, äußerst diskriminativ, wogegen die in Kap. 4.2 betrachteten Dokumenttypen zumeist wenig relevante Kontextinformation beinhalteten.

4.4 Zusammenfassung

In Kap. 4.1 wurde eine Methode beschrieben, die die Inferenz von Bildungsvorschriften in Form von Regulären Ausdrücken anhand einer Menge von (positiven) Beispielentitäten ermöglicht. Bei der Regelinferenz werden für die Instanzmenge ein Präfix- und Suffixautomat erzeugt, die von der Syntax der Entitäten auf Basis von Zeichen- und Tokenklassen ab-

strahieren. Zur Selektion von feingranularen Mustern (fixen Zeichen oder Token) wurde das Verhältnis von individueller zu absoluter Anzahl von Instanzteilsequenzen im Vergleich zum entsprechenden mittleren Verhältnis in Vorgänger und Nachfolgertransitionen vorgeschlagen.

Die Selektion von am besten geeigneten Abstraktionsebenen zur Repräsentation von Merkmalen basiert auf dem Prinzip der „*Minimum Description Length*“. Die Übersetzung des Automaten in mehrere Reguläre Ausdrücke stellt eine Menge von Regeln zur Verfügung, die in beliebigen Regel-basierten Extraktionssystemen verwendet werden können.

Eine detaillierte Analyse verwandter Arbeiten zeigte, dass in der Literatur kein vergleichbares Verfahren beschrieben wurde, das Beispielentitäten auf verschiedenen abhängigen Abstraktionsebenen unterschiedlicher Granularität analysiert, potenzielle Bildungsvorschriften ableitet und deterministische Regeln zur Extraktion erzeugt. Prinzipiell unterstützen Statistische Maschinelle Lernverfahren wie *Conditional Random Fields (CRF)* die Kombination von abhängigen Merkmalen unterschiedlicher Granularität. Zum Training solcher Verfahren ist jedoch ein annotierter Trainingskorpus zur Bestimmung von Modellparametern notwendig. Eine Intervention durch Nutzer zur Fehlerbeseitigung und Feinjustierung von Extraktionsmechanismen ist weiterhin bei dem hier vorgestellten Verfahren wesentlich einfacher als bei der Verwendung von Statistischen Maschinellen Lernverfahren.

Die experimentelle Evaluation zeigte die praktische Anwendbarkeit des vorgestellten Verfahrens und eine vergleichbare Extraktionsqualität zu einem CRF-Ansatz mit einer ähnlichen Merkmalskonfiguration, der jedoch anhand von Beispieldokumenten trainiert wurde und Kontextmerkmale berücksichtigte. Mehr noch zeigte sich, dass das hier vorgestellte Verfahren bei wenigen gegebenen Beispielentitäten, die einer relativen klaren Bildungsvorschrift folgen, einem CRF-Ansatz in vielen Fällen überlegen ist.

Künftig ist die Integration des hier vorgestellten Verfahrens mit Statistischen Maschinellen Lernverfahren und/ oder bisherigen Regellernverfahren interessant, so dass auch Anwendungsfälle, in denen annotierte Trainingskorpora mit wertvollen Kontextinformationen zur Verfügung stehen, in vollem Umfang unterstützt werden können. Für zukünftige Anwendungsfälle ist weiterhin eine Erweiterung um höherwertige Merkmalsebenen denkbar.

5

Identifikation von Entitäten auf Basis Graph-strukturierter Referenzdaten

Die im vorherigen Kapitel vorgestellten Methoden unterstützen die Extraktion von unbekannten Entitäten auf Basis erlernter Bildungsvorschriften. In diesem Kapitel untersuchen wir das Problem der Identifikation von Entitäten in Dokumenten, die in Graph-strukturierten Referenzdaten vordefiniert sind (siehe Def. 2.3.2). Teilprobleme und -lösungen der hier vorgestellten Methoden wurden in [BLD08, BBH+09, BHH+10] veröffentlicht.

Im Kontext der *Enterprise Search* existieren zahlreiche Anwendungen, die durch eine Identifikation von Entitäten auf Basis von Graph-strukturierten Referenzdaten profitieren (siehe Kap. 1.2): Eine (semi-)automatische Verifikation von Rechnungen (*Szenario I* in Kap. 1.2.1) erfordert unter anderem die Zuordnung von Rechnungen zu Bestellungen, wobei erstere in Textform vorliegen und letztere in strukturierter Form in einem RDBMS gespeichert sind. Anhand der durch eine Bestellung im RDBMS subsumierten Entitäten, wie z. B. Kunden und Bestellpositionen, soll der Zusammenhang zwischen Rechnung und Bestellung hergestellt werden, auch wenn die Bestellnummer nicht im Rechnungstext enthalten ist. Zur Verifikation von einzelnen Rechnungsposten ist eine Disambiguierung von direkten Übereinstimmungen zwischen Rechnungstext und RDBMS notwendig.

In Kap. 1.2.2 wurde als *Szenario II* ein System zur semantischen Dokumentsuche im Unternehmensforum *SAP Community Network (SCN)* betrachtet. Die Problemstellung besteht darin, Anfragen und Dokumente auf einen gegebenen Graph-strukturierten Referenzdatensatz abzubilden – in dem konkreten Anwendungsfall die SAP Terminologie (SAPTerm) –, also die relevanten Teilgraphen zu bestimmen und die Ähnlichkeit zwischen den entstehenden Anfrage- und Dokumentgraphen zur Relevanzbewertung der Suchresultate heranzuziehen.

Häufig ist die Identifikation von Entitäten mit Graph-strukturierten Referenzdaten problematisch. Die Ursachen hierfür sind vielfältig (siehe auch Kap. 2.3.2): Zum Ersten ist die Identifikation von Entitäten über Text-Referenzdaten-Übereinstimmungen problematisch und durch bisherige Ansätze nicht zufriedenstellend gelöst. Zum einen werden Attribute von Entitäten in Texten häufig nicht in der gleichen Art und Weise in Texten erwähnt, wie sie in Referenzdaten gepflegt werden. Zum anderen wird eine Entität häufig über mehrere Attribute im Referenzdatensatz beschrieben, was bei der Identifikation von Entitäten berücksichtigt werden muss.

Weiterhin ist die Ausnutzung der Struktur der Referenzdaten problematisch. Zum Beispiel ist, im Falle von typischen, über mehrere Ebenen spannenden Produkt-Teil-Hierarchien, die Se-

lektion der relevantesten Entität, die einen Text beschreibt, besonders schwierig. Selbst wenn Text-Referenzdaten-Übereinstimmungen ohne weiteres bestimmt werden könnten, ist die Selektion der relevantesten Entität für einen Text nicht trivial, wenn die Übereinstimmungen in beliebigen Hierarchieebenen der Referenzdaten auftreten.

Das vorliegende Kapitel ist wie folgt aufgebaut: In Kap. 5.1 beschreiben wir die Konzepte zur Identifikation und Disambiguierung von Entitäten. Einführend wird in Kap. 5.1.1 auf naive Methoden zur Identifikation eingegangen. Wir erläutern Problemstellungen, die bei Experimenten mit einem Dokumentenkorpus von 120.000 Foreneinträgen des SCNs identifiziert wurden.

In Kap. 5.1.2 stellen wir die entwickelten Konzepte zur Herstellung und Bewertung von direkten Text-Referenzdaten-Übereinstimmungen vor. Wir beginnen mit der Betrachtung von Vorverarbeitungsschritten: Zum einen werden die zu analysierenden Texte für den Abgleich zwischen Text und Referenzdaten vorsegmentiert und zum anderen Attributinhalte von Entitäten zur Beschleunigung des Abgleichs indexiert. Am Ende von Kap. 5.1.2 wird detailliert auf die Bewertung von Übereinstimmungen zwischen Textsegmenten und den verschiedenen Attributen von Entitäten eingegangen.

In Kap. 5.1.3 diskutieren wir die Ausnutzung der Graphstruktur der Referenzdaten zur Bewertung von direkt und indirekt im Text referenzierten Entitäten. Es wird die Relation der vorgestellten Konzepte zu dem in Kap. 3.1.3 diskutierten Meta-Datenmodell von *RapidUIM* erläutert und auf die Bewertung von Entitäten im Dokumentkontext eingegangen. Abschließend greifen wir die verschiedenen Anwendungsszenarien aus Kap. 1.2 auf und diskutieren die Verwendung der Resultate z. B. zur Bewertung von Dokumenten im Kontext der semantischen Dokumentsuche.

In Kap. 5.2 werden die entwickelten Methoden mit ausgewählten, verwandten Ansätzen in drei Anwendungsszenarien experimentell verglichen. Wir betrachten Dokumentenkorpora für *Szenario II* und *Szenario III* aus Kap. 1.2 und verwenden die SAP Terminologie zur Identifikation von Entitäten. Der Rechnungskorpus, der in Kap. 4.2 beschrieben wurde, kann nicht zur Evaluation der hier beschriebenen Konzepte für *Szenario I* aus Kap. 1.2 herangezogen werden, da zugehörige Referenzdaten die Entitäten der Texte nur partiell widerspiegeln. Der Grund dafür ist, dass die Daten aus einem Testsystem zur Evaluation der *Usability* akquiriert wurden und z. B. Einzelpositionen der Bestellungen nur teilweise in den Referenzdaten gepflegt wurden.

Zur Evaluation eines zu *Szenario I* verwandten Problems verwenden wir den Referenzdatensatz *Internet Movie Database (IMDB*[1]*)* mit einem Dokumentenkorpus aus nutzergenerierten Filmbewertungen. Die Problemstellung ist insofern verwandt, als dass die Filmbeschreibungen in der IMDB als Produkte (z. B. DVD-Titel) angesehen werden können. Unternehmen und Personen stehen wie in anderen Unternehmensdatenbanken mit Produkten in Relation und können zur Identifikation herangezogen werden. Auch werden verschiedene „schwache“ Attribute hinsichtlich der Identifizierbarkeit von Entitäten, wie das Erscheinungsjahr der Filme in der IMDB gepflegt.

In Kap. 5.3 werden verwandte Arbeiten analysiert. Wir vergleichen ausgewählte Bewertungsmethoden vorheriger Ansätze detailliert und stellen Nachteile der Methoden im Vergleich zu dem in Kap. 5.1 vorgestellten Verfahren heraus.

5.1 Konzept zur Identifikation und Disambiguierung von Entitäten

Der nachfolgende Rückblick auf das Szenario der Forensuche aus Kap. 1.2.2 veranschaulicht beispielhaft die Komplexität der Problemstellung. Es werden Beobachtungen, die bei der Verwendung von naiven Ansätzen wie einem exakten Wörterbuchabgleich zur Identifikation von Entitäten in Forenbeiträgen auftreten, erläutert. Weiterhin gehen wir darauf ein, inwiefern die

[1] htttp://imdb.com

dargestellten Beobachtungen auf andere Datensätze übertragbar sind. Im Anschluss gehen wir in Kap. 5.1.2 und Kap. 5.1.3 auf die hier entwickelten Konzepte ein.

5.1.1 Naive Ansätze zur Identifikation von Entitäten im SCN

Bei den Inhalten des SAP Community Network (SCN) handelt es sich in der Regel um Informationen zu SAP-Technologien. Nutzer des SCN suchen und diskutieren Lösungen zu Softwareproblemen, Informationen zu Produkten, Downloads etc. (siehe Kap. 1.2.2). Es ist daher anzunehmen, dass beinahe jedem Forumeintrag mindestens einem Softwaremodul, das in der SAP Terminologie (SAPTerm) spezifiziert ist, zugeordnet werden kann. Die in Kap. 1.2.2 vorgestellte Studie zu Suchintensionen im SCN untermauert diese Vermutung unter der Annahme, dass Suchbegriffe die Inhalte des Forums widerspiegeln. Von den 400 untersuchten Anfragen konnten 316 genau einer Entität in SAPTerm zugeordnet werden. In den anderen Fällen war die Zuordnung nicht eindeutig.

Softwaremodule und technische Konzepte in SAPTerm (siehe Kap. 1.2.2, Abb. 1.5) sind ein typisches Beispiel für ein Fachvokabular mit begrifflicher Ordnung. Die Bezeichnungen von Softwaremodulen in SAPTerm richten sich zum Beispiel nach ihrer Funktion wie „*Customer Invoice Processing*“. Die Konzepte werden in Texten auf unterschiedlichste Weise referenziert. Zum Beispiel adressiert ein Text der Art „*I want to process invoices from my customers automatically*“ implizit oben genanntes Softwaremodul.

Bezeichnungen für technische Konzepte wie „*business object*“ („*Geschäftsobjekt*“), die in Verbindung zu Softwaremodulen gespeichert werden, treten in verschiedenen Bedeutungen auf. Bei Abkürzungen stellt sich das selbe Problem. „*XI*“ hat z. B. die Bedeutungen: „*SAP Exchange Infrastructure*“ oder „*Business Objects Extreme Insight*“, wobei hier „*Business Objects*“ als Name des SAP-Unternehmens und nicht im Sinne von „*Geschäftsobjekten*“ verwendet wird.

Der Dokumentenkorpus, der für die im Folgenden beschriebenen Versuche verwendet wurde, besteht aus 120.000 Forenbeiträgen des SCN vom Anfang des Jahres 2008. Forumeinträge dieses Korpus haben eine durchschnittliche Länge von 300 Zeichen oder 3,6 Sätzen. Wir gehen zunächst auf Probleme im Bezug auf die Vollständigkeit und anschließend auf die Genauigkeit von Identifikationsergebnissen ein. Abschließend diskutieren wir die Übertragbarkeit der Beobachtungen auf andere Anwendungsfälle.

Vollständigkeit

Das wesentliche Ziel in diesem Anwendungsfall ist die Zuordnung von Foreneinträgen zu relevanten Softwaremodulen. Mit Hilfe eines Wörerbuchabgleichs unter Verwendung der ca. 16.000 Lang- und Kurznamen der Softwaremodule wurden insgesamt 37.000 exakte Treffer in den 120.000 Forumeinträgen erzielt. Da jeder Forumeintrag mindestens einem Softwaremodul zugeordnet werden soll, ist diese Anzahl an Übereinstimmungen unbefriedigend.

Unter der Annahme, dass eine große Menge von Softwarekomponenten auf Grund von fehlerhaften Schreibweisen und anderer leichter syntaktischer Hetereogenitäten nicht identifiziert werden konnte, wurde ein approximatives Abgleichverfahren basierend auf der Levenshtein Distanz getestet. Mit einer Levenshtein Distanz von 1 stieg die Trefferrate um den Faktor 3,3– also im Durchschnitt auf einen Treffer pro Forumeintrag. Wenn man bedenkt, dass im Extremfall in einem einzelnen Forumeintrag über hundert Treffer erzeugt wurden, da sich z. B. Abkürzungen stark ähneln, ist auch dieses Resultat nicht akzeptabel.

Es konnte beobachtet werden, dass 10% der Treffer genau ein Wort umfassten, wogegen jedoch 83% der Softwaremodulbezeichner aus mehreren Wörtern zusammengesetzt sind – ein Missverhältnis, da ganz klar die Popularität der Softwaremodule im Forum in keinster Weise mit der Länge der Softwaremodulbezeichner zusammenhängt. Eine Identifikationsmethode für Entitäten, die auch (im Text verteilte) Teilsequenzen der Bezeichner berücksichtigt, ist daher

für Ergebnisse mit einer akzeptablen Vollständigkeit zwingend notwendig. Durch das Tokenisieren der Softwaremodulbezeichner und die Selektion der längsten zusammenhängenden Übereinstimmungen (siehe z. B. [RRK+08, TT04]) stieg die Anzahl der Treffer auf durchschnittlich 57,4 Treffer pro Forumeintrag.

Eine zufriedenstellende Vollständigkeit konnte jedoch bei subjektiver Beurteilung der Resultate auch auf diese Art und Weise nicht erreicht werden. Die gezielte Ausnutzung aller in SAPTerm vorhandenen Informationen wie den technischen Konzepten, die den Softwaremodulen zugeordnet sind, und den Beziehungen zwischen Softwaremodulen ist daher geboten.

Genauigkeit

Beides, der fehlertolerante Abgleich und das Tokenisieren der Softwaremodulbezeichner führten dazu, dass von den 57,4 Treffern pro Foreneintrag im Durchschnitt 56,4 nicht eindeutig einer Entität zugewiesen werden konnten. Ohne ein Verfahren zur Dismabiguierung von Treffern im Dokumentkontext ist mit diesem Ansatz keine akzeptable Genauigkeit zu erreichen.

Mehrdeutigkeiten sind bei der Anwendung jedes größeren Refenzdatensatzes zu erwarten. Im Beispiel von SAPTerm sind lediglich 30% aller Bezeichner eindeutig. Wenn einzelne Token der Bezeichner berücksichtigt werden, verschärft sich dieses Problem enorm. Die gesamte Terminolgie basiert auf lediglich 23.000 Wörtern (im Vergleich zu 1.000.000 Wörtern in der Englischen Sprache [MCM03]). Das am häufigsten auftretende Wort „*Management*“ wird in mehr als 3.600 Bezeichnern verwendet und tritt im betrachteten Dokumentenkorpus in ca. 10.000 Dokumenten auf.

Als weiterführendes Beispiel für Mehrdeutigkeiten sei an dieser Stelle das Sofwaremodul „*Plant Maintenance*“ genannt. Dessen Kurzform „*PM*“ trat 5.000 mal im Beispielkorpus auf. Ohne auf alternative Bedeutungen von „*PM*“ in SAPTerm einzugehen, zeigte eine Stichprobe, dass lediglich 10% der Treffer ein Konzept in SAPTerm referenzierten und es sich bei 90% der Treffer um den Teil eines Zeitstempels handelte.

Ein Weg, Mehrdeutigkeiten aufzulösen ist es, zusätzliche Informationen, die im Referenzdatensatz zu den Entitäten in Beziehung stehen, wie technische Konzepte und Abkürzungen im Fall von SAPTerm zur Disambiguierung auszunutzen. Das Problem der Mehrdeutigkeiten wird durch ein solches Vorgehen zunächst jedoch verschärft, da z. B. das Wort „*PM*“ in 655 technischen Konzepten in SAPTerm auftritt. Wenn man alle Token aller Bezeichner in SAPTerm betrachtet, steigt die Anzahl der Übereinstimmungen pro Forumeintrag auf über 700 - also im Durchschnitt auf ca. 200 Übereinstimmungen pro Satz.

Zusätzlich konnte in vielen Fällen beobachtet werden, dass die oben eingeführte Filterung bezüglich der längsten zusammenhängenden Übereinstimmungen oft dazu führte, dass Treffer, die in Beziehung zu den wirklich relevanten Entitäten stehen, nicht berücksichtigt wurden. Als Beispiel führen wir an dieser Stelle den Textausschnitt „*PM: maintenance costs by plant and equipment*“, an. Die Phrase „*plant and equipment*“ ist ein technisches Konzept im Kontext des Softwaremoduls „*Financial Accounting*“, wogegen „*Plant*“ Bestandteil des Langnamens des Moduls „*PM*“ und „*equipment*“ ein Konzept im Kontext dieses Moduls ist. Die relevanten Treffer „*plant*“ und „*equipment*“ im Kontext von „*PM*“ würden also durch die Filterung entfernt.

Natürlich ist jedoch die Heuristik, der längsten Übereinstimmung eine größere Bedeutung beizumessen, in vielen Fällen durchaus sinnvoll. Durch das Aufgeben der Filterung entsprechend der längsten zusammenhängenden Übereinstimmungen wird die Anzahl Übereinstimmungen pro Foreneintrag vervielfacht.

Die bisher genannten Beispiele zeigen die manigfaltigen Problemstellungen, die sich bei Zuordnung von Softwaremodulen zu Foreneinträgen ergeben. Die Hoffnung ist nun, dass eine geeignete Bewertung von Einzeltreffern und eine Aggregation von in Beziehung stehenden Treffern zu einer hohen Genauigkeit bei der Identifikation von Softwaremodulen führt.

Eine einfache Heuristik, die die Bewertungen von Treffern in der direkten Nachbarschaft von Entitäten aggregiert, ist jedoch ebenso wenig Erfolg versprechend wie die Aggregation von Übereinstimmungen hinsichtlich der „Wurzelknoten" der Modulhierarchie. Die erste Variante vernachlässigt die Entitäten-Hierarchie. Im Falle, dass ein Text ein Integrationsproblem zwischen zwei Softwaremodulen beschreibt, sollte z. B. ein Elternelement in der Modulhierarchie als relvantestete Entität für den Text identifiziert werden. Die zweite Variante verallgemeinert zu sehr und beschreibt in den meisten Fällen nicht die eigentliche Intention des Textes.

Übertragbarkeit

Die in diesem Kapitel erläuterten Probleme treten in ähnlicher Form häufig im Unternehmenskontext auf. Die Hierarchie von Softwaremodulen ist eine spezielle Art einer Produkt-Teil-Hierarchie, die in vielen Bereichen vorkommen. Das Problem der Selektion von korrekten Hierarchieebenen für einen gegebenen Text oder Textausschnitt ist ebenso ubiquitär. Man bedenke zum Beispiel die Verarbeitung von Rechnungen, in denen Rechnungspositionen in Einzelpositionen aufgeschlüsselt werden (z. B. „*Ölfilter wechseln*" als Teil von „*Ölwechsel*").

Bei der Verwendung von Taxonomie-artigen Terminologien, in denen nicht jedem Element in der Hierarchie Instanzdaten zugeordnet sind, stellen sich ähnliche Probleme. Die Europäische Union stellt zum Beispiel eine Terminologie[2] für ihre „Kunden" zur Verfügung, die in Fachgebiete eingeteilt ist (z. B. „*Wirtschaftsleben*" > „*Verwaltungsrecht*" > „*öffentliche Ausschreibungen*"). Wenn ein Text in ein Fachgebiet (entspricht hier einer Entität) eingeordnet werden soll, ist die Bestimmung der korrekten Hierarchiebene entscheidend, da natürlich auch Fachgebiet-übergreifende Texte von Bedeutung sind.

Die Verwendung von Fachvokabular zur Umschreibung von Entitäten in Datenbanken und Terminologien wurde bereits hinreichend motiviert. Zur Produktbeschreibung werden z. B. insbesondere im Dienstleistungsbereich in strukturierten wie unstrukturierten Daten eher Umschreibungen von Leistungen verwendet, als eindeutige Bezeichnungen. Hinsichtlich der Verwendung von Fachvokabular in Terminologien sei an dieser Stelle auf die IT-Terminologien von IBM[3] oder Microsoft[4] hingewiesen, die in Struktur und Charakteristiken der SAP Terminologie ähneln.

Auch in *Amazons Internet Movie Database (IMDB)*[5] finden sich neben eindeutigen Bezeichnern, auch Darstellernamen, Filmtitel oder Rollennamen, die eher umschreibenden Charakter haben. So ist z. B. das „*Orchester des Nordwest Deutschen Rundfunks*" ein Darstellername, „*Kunst und Literatur in der DDR [..]*" ein Filmtitel oder „*Secret Service Special Agent Darren Marinkovitch*" eine Rollenbeschreibung, die in Texten in leicht veränderter Art und Weise referenziert werden können.

Klar ist, dass in jedem Referenzdatensatz wie der IMDB mit einer Größe von mehreren zehn Millionen Einträgen eine Vielzahl von Mehrdeutigkeiten besteht. So sind z. B. ca. 36.000 Personen mit dem Vornamen „*John*" und über 1.000 Filmtitel, die die Wortgruppe „*the wedding*" enthalten, in der IMDB zu finden.

5.1.2 Textvorverarbeitung und Abgleich von Text und Referenzdaten

Bevor in Kap. 5.1.3 das *Ranking* von Entitäten für einem gegebenen Text diskutiert wird, betrachten wir die Vorsegmentierung von Texten, wobei wir hier insbesondere auf die Verarbeitung von Fließtexten eingehen. Im Weiteren beschreiben wir die Indexierung der strukturierten Referenzdaten und den Abgleich von Textsegmenten mit Attributinhalten. Abschließend defi-

[2] `http://iate.europa.eu`, Stand: 01.08.2009
[3] `http://www-01.ibm.com/software/globalization/terminology`, Stand: 01.08.2010
[4] `http://www.microsoft.com/resources/glossary/default.mspx`, Stand: 01.08.2010
[5] `htttp://imdb.com`, Stand: 01.08.2010

nieren wir ein Konfidenzgewicht zur Bewertung von Attributübereinstimmungen im Kontext eines Textsegments.

Textvorverarbeitung

Es gibt zwei verbreitete Ansätze zum Abgleich von Texten mit Wörterbüchern. Am gebräuchlichsten ist die Variante, ein gleitendes Textfenster mit der Größe des längsten Wörterbucheintrags über den Text zu führen und jeweils den Fensterinhalt zeichen- oder tokenweise mit Wörterbucheinträgen abzugleichen. Es existieren optimierte Varianten zur effizienten Bestimmung der relevantesten Wörterbucheinträge für ein Tokenfenster (siehe [CNS06]).

Der Ansatz von [CNS06] schlägt eine Tokenisierung der Wörterbucheinträge und eine Aufteilung in „starke“ und „schwache“ Token auf Basis von *TF-IDF*-Gewichten [RW99] vor. Ein Fenster mit der Größe des längsten Wörterbücheintrags wird tokenweise über den Ursprungstext geführt und mit den starken Token des Wörterbuchs abgeglichen. Anschließend werden die (rekonstruierten) Wörterbucheinträge im Kontext des Textfensters bewertetet. Wenn Wörterbucheinträge äußerst lang sind und auch im Text verteilte Teilübereinstimmungen bewertet werden sollen, ist eine Adaption dieses Verfahrens nicht ohne weiteres möglich.

Die zweite Variante des Wörterbuchabgleichs basiert auf einer Vorsegmentierung des Textes und dem Abgleich einzelner Segmente mit dem Wörterbuch (siehe z. B. [CGRM06]). Dadurch wird im Allgemeinen der Suchraum für Übereinstimmungen stark eingegrenzt. Zusätzlich wirkt sich in der Regel eine anwendungsspezifische Erzeugung von Textsegmenten positiv auf die Extraktionsqualität aus, da z. B. irrelevante Daten in Texten ausgeblendet werden können.

Unter Annahme von Token-basierten *TF-IDF*-Gewichten wäre z. B. für den Text „*Configuration failed because the Adapter Framework ...*“ im Kontext von SAPTerm das Wort „*because*“ mit Abstand die am höchsten bewertete Übereinstimmung, da es in SAPTerm lediglich zweimal auftritt. Bei einer Vorsegmentierung und Filterung anhand von Substantivgruppen werden die Vergleiche auf die relevantesten Inhalte beschränkt und derartige Probleme vermieden.

Für verschiedene Anwendungsfälle sind unterschiedliche Vorsegmentierungsverfahren denkbar. In *RapidUIM* sind standardisierte Module zur Vorsegmentierung von HTML- und Fließtexten vorgesehen. Das HTML-Modul bricht einen Text in HTML-Elemente und Elementinhalte und ist standardmäßig so konfiguriert, lediglich die Elementinhalte als relevante Segmente weiter zu propagieren.

Leichtgewichtige morphologische Parser (so genannte *Shallow Parser*) [SP03] sind sehr gut für eine Vorsegmentierung von Fließtexten geeignet, da sie Substantiv- oder Verbgruppen ohne aufwendige Erzeugung von kompletten Abhängigkeitsbäumen mit einer Genauigkeit von über 90% erkennen [Sar08]. In *RapidUIM* stehen zwei Module, basierend auf dem *Business Objects ThingFinder* (basierend auf [Hul00]) und dem *Stanford Part-of-Speech Tagger* [TKMS03], zur Vorsegmentierung von Fließtexten zur Verfügung. Anzumerken ist, dass wir im SCN-Anwendungsfall lediglich Substantivgruppen berücksichtigen, da die relevanten SAPTerm-Inhalte in diese Kategorie fallen.

Indexierung von Referenzdaten

Um den Abgleich von Text und Referenzdaten zu beschleunigen, werden alle Attribute aller Entitäten des Referenzdatensatzes gemäß einer Konfiguration indexiert. Dabei werden nicht nur die gesamten Attributinhalte oder einzelne Wörter indexiert, sondern auch alle Token-N-Gramme, die sich aus den Attributinhalten ergeben (für alle N, $1 \leq N \leq$ Länge des Inhalts).

Eine Rekonstruktion von exakten Wörterbuchübereinstimmungen für Wortgruppen zur Laufzeit ist im Gegensatz zu einer vollständigen Tokenisierung von Referenzdaten nicht notwendig. Auch ist die Bestimmung von Relevanzwerten für zusammenhängende Wortgruppen auf Basis von Statistiken des Referenzdatensatzes ohne weiteres möglich. Ein Beispiel soll den Vor-

teil eines solchen Vorgehens illustrieren. Gegeben sei der Text „*Configuration Adapter Framework*“ und die SAPTerm-Konzepte „*JMX Configuration Adapter*“ und „*XI Adapter Framework*“. Für dieses Beispiel würde, obwohl der Textausschnitt „*Adapter Framework*“ genau so im zweiten Konzept auftritt, die Übereinstimmung mit dem ersteren Konzept höher bewertet, da „*Configuration*“ ein wesentlich höheres *TF-IDF*-Gewicht als „*Framework*“ zugeordnet wird. Wenn wir Token-N-Gramme und entsprechende Relevanzwerte verwenden, besteht dieses Problem nicht.

Im Folgenden notiere a ein Attribut, t ein Textsegment und $\tilde{a}$ oder $\tilde{t}$ ein entsprechendes N-Gramm. Gegeben sei z. B. das Attribut $a \equiv$"*JMX Adapter Framework*" $\equiv$ abc (abc ist eine schematische Darstellung), so ergeben sich sechs N-Gramme, die im Index gespeichert werden: $\{\tilde{a}_1, \ldots, \tilde{a}_6\} \equiv$ {a,b,c,ab,bc,abc}. Da alle N-Gramme z. B. auch mit $N = 1$ gebildet werden, ist es möglich, im Text verteilt auftretende Attributübereinstimmungen (auch mit veränderter Wortreihenfolge) zu rekonstruieren. Wir diskutieren diesen Zusammenhang im Kontext der Bewertung von Entitäten.

Der Index wird in Form eines Präfixbaums kodiert. Unter Nutzung einer zusätzlichen Präfix-Kompression [BHF09] steigt der Speicherbedarf im Vergleich zu einem einfachen Wortindex in vertretbarem Maße. Zusätzlich können N-Gramme, die extrem mehrdeutig sind, entfernt werden. Derartige N-Gramme erzeugen hohe Kosten bei Abgleich und Bewertung, aber beeinflussen die Resultate bei dem im Folgenden vorgestellten Bewertungsverfahren nur geringfügig.

Alle N-Gramme (mit $N <$ Länge des ursprünglichen Attributinhalts), deren Anzahl im Attributindex größer als ein Schwellwert k ist, werden gelöscht. Der Schwellwert k sollte dabei so hoch (z. B. $k = 200$ für SAPTerm) gewählt werden, dass die Bewertung von Entitäten nicht signifikant beeinflusst wird. Für das Beispiel von SAPTerm ergibt sich mit $k = 200$ eine Indexgröße, die im Vergleich zu einem Tokenindex lediglich um den Faktor 2 größer ist. Im Falle der IMDB ist der Speicherbedarf lediglich um den Faktor $1,6$ größer.

Abgleich und Bewertung von Textsegmenten

Zur Laufzeit wird für jedes Textsegment t eine disjunktive Indexanfrage nach allen Segment-N-Grammen: $\tilde{t}_1 \vee \tilde{t}_2 \vee \ldots \vee \tilde{t}_m$ erzeugt. Resultat sind die k am wenigsten mehrdeutigen Attribut-N-Gramme, die mit der Anfrage übereinstimmen.

Der Abgleich berücksichtigt auch approximative Übereinstimmungen, die ähnlicher als ein konfigurierbarer Schwellwert sind und ein Präfix bestimmter Zeichenlänge mit dem entsprechenden Segment-N-Gramm der Anfrage teilen. Auf die Ähnlichkeitsmetrik gehen wir im Kontext der Bewertung von Attributübereinstimmungen ein. Die Einschränkung auf ein gemeinsames Präfix (z. B. der Zeichenlänge 1) beschleunigt die Anfrageverarbeitung erheblich [CGK06].

Weiterhin werden sich überlagernde, kürzere Übereinstimmungen für ein und das selbe Attribut entfernt, wie z. B. für $t = a \equiv$ ab die Attribut-N-Gramme $\tilde{a}_1 \equiv$ a und $\tilde{a}_2 \equiv$ b, so dass für die Übereinstimmung von a und t lediglich das längste Attribut-N-Gramm $\tilde{a}_3 \equiv$ ab berücksichtigt wird.

In Abb. 5.1 ist ein Beispiel für den Abgleich des Textes „*The configuration adaptor framework of XI*“ mit den Referenzdaten dargestellt. Die Übereinstimmung mit dem Attribut-N-Gramm „*adaptor*“ ist eine exakte Übereinstimmung, jedoch in dem gegebenen Kontext irrelevant, da es sich um einen Fehler in der Schreibweise handelt und der Nutzer offensichtlich „*adapter*“ meinte.

Die Übereinstimmung mit „*Configuration Adapter*“ ist der längste zusammenhängende Treffer bezüglich der Zeichenkettenlänge und überschneidet sich mit dem Attributtreffer „*Adapter Framework*“. Die Übereinstimmung mit dem Attribut „*XI Adapter Framework*“ ist am relevantesten. Die Bestandteile dieses Attributs treten jedoch im Text in anderer Reihenfolge als im Attribut

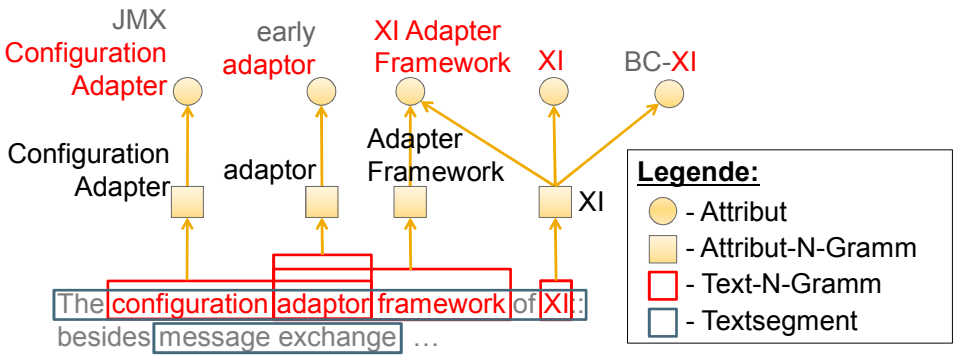


Abb. 5.1: Beispiel für den Abgleich von Text und Attributen.

auf. „*XI*“ kommt im Referenzdatensatz weiterhin sehr oft als Abkürzung vor, was durch zwei zusätzliche Treffer in Abb. 5.1 illustriert wird.

Das Beispiel in Abb. 5.1 zeigt die wesentlichen Problemstellungen hinsichtlich der Bewertung von Übereinstimmungen in einem Segment. Im Folgenden wird auf die Kombination von verschiedenen Bewertungsmetriken zur Bestimmung eines Konfidenzgewichts für Attributübereinstimmungen eingegangen, bevor wir in Kap. 5.1.3 die Bewertung von Entitäten im Kontext des Gesamtdokuments betrachten.

Ähnlichkeit von Übereinstimungen auf Zeichenebene. Approximative Abgleichverfahren auf Zeichenebene sind hilfreich, um die Vollständigkeit von Identifikationsergebnissen zu erhöhen, insbesondere wenn fehlerbehaftete unstrukturierte Daten analysiert werden, aber auch wenn leichte syntaktische Variationen zugelassen werden sollen. Daher werden alle Attribut-N-Gramme als Übereinstimmung berücksichtigt, deren Ähnlichkeit (notiert als $w_{sim} \in [0,1]$) mit einem Segment-N-Gramm mindestens einem konfigurierbaren Schwellwert $s \in [0,1]$ entsprechen.

In der Literatur werden eine Vielzahl von Bewertungsmetriken für approximative Übereinstimmungen beschrieben, die für spezielle Anwendungsszenarien geeignet sind (siehe [EIV07] für einen Überblick). Zum Beispiel ist die Jaro-Winkler-Distanz, welche Fehler am Beginn der Zeichenkette stärker gewichtet, besonders für Eigennamen (z. B. von Personen) geeignet, da Nutzer eher am Ende von Namen zu Fehlern neigen. *Soundex* ist hingegen ein phonetisches Bewertungsmaß und besonders bei textuell erfasster Sprachkommunikation zu präferieren.

In der Praxis werden häufig einfachere, aber weniger spezialisierte Verfahren wie die Levenshtein- oder Zeichen-N-Gramm-Ähnlichkeit verwendet [Nav01]. Die Levenshtein-Distanz betrachtet die Anzahl der Editieroperationen, die notwendig sind, um eine Zeichenkette einer zweiten anzupassen. Die Zeichen-N-Gramm-Ähnlichkeit basiert auf der Anzahl der übereinstimmenden Teilzeichenketten fester Länge, wobei auch die N-Gramm-Positionen betrachtet werden müssen [GIJ+01].

Eine Bewertung mit Zeichen-N-Gramm-Ähnlichkeiten erweist sich jedoch insbesondere bei dem Vergleich von kurzen Wörtern wie z. B. Akronymen als ungeeignet [WXLZ09]. Daher ist die Levensthein-Distanz für den Einsatz im *Enterprise Search*-Umfeld besser geeignet und wird im Folgenden für den Abgleich zwischen Textsegmenten und Attributinhalten herangezogen.

Sei $sim(\tilde{t}, \tilde{a}) \in N$ die Levensthein-Distanz zwischen einem Text-N-Gramm $\tilde{t}$ und einem Attribut-N-Gramm $\tilde{a}$. Zur Normierung der Levensthein-Distanz verwenden wir die Anzahl der Buchstaben des Text-N-Gramms $|\tilde{t}|$, so dass der entstehende Wert die relative Anzahl von Änderungen zum Orginaltext widerspiegelt. In der Praxis hat sich außerdem eine weitere Skalierung des Ähnlichkeitswertes mit $1/sim(\tilde{t}, \tilde{a})$ als günstig erwiesen, da so Fehler stärker „ge-

ahndet“ und Nutzern die Konfiguration von $s \in [0,1]$ erleichtert wird. Ein Wert von $s = 0,5$ würde ohne Skalierung bereits zu Treffern führen, die sich in 50% der Buchstaben vom Text unterscheiden.

Die Metrik zur Gewichtung der Ähnlichkeit von Text- und Attribut-N-Grammen auf Zeichenebene – notiert als $w_{sim}(\tilde{t}, \tilde{a}) \in [0,1]$ – wird zusammenfassend mit folgender Formel berechnet:

$$w_{\text{sim}}(\tilde{t}, \tilde{a}) = \frac{1}{sim(\tilde{t}, \tilde{a}) + 1} \left(1 - \frac{\min(sim(\tilde{t}, \tilde{a}), |\tilde{t}|)}{|\tilde{t}|}\right), \tag{5.1}$$

wobei der linke Teil der Gleichung die Skalierung gemäß $1/sim(\tilde{c}, \tilde{t})$, der rechte Teil die normierte Levenshtein-Distanz repräsentiert und $\min(sim(\tilde{t}, \tilde{a}), |\tilde{t}|)$ ein Ergebnis von $w_{\text{sim}}(\tilde{t}, \tilde{a}) \in [0,1]$ sicherstellt. In Abb. 5.2 ist die Bewertung der Ähnlichkeit anhand des laufenden Beispiels illustriert.

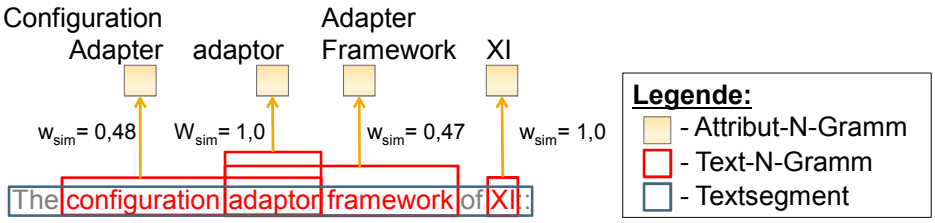


Abb. 5.2: Ähnlichkeit auf Zeichenebene für Beispiel aus Abb. 5.1

Relevanz von Übereinstimmungen im Kontext überlappender Treffersequenzen. Die Relevanzbewertung von Text- und Attribut-N-Grammen ist neben der Ähnlichkeitsbestimmung ein wichtiger Faktor für die Bewertung von Treffern. Im Kontext von Referenzdaten wie SAPTerm kann die Ein- bzw. Mehrdeutigkeit eines Attribut-N-Gramms als Relevanzfaktor herangezogen werden. Die Ein- bzw. Mehrdeutigkeit beschreibt, in welchem Maße eine Text-Attribut-N-Gramm-Übereinstimmung ein bestimmtes Attribut identifiziert.

Wörter oder Wortgruppen der Referenzdaten treten in den hier betrachteten Anwendungsfällen auch im allgemeinen Sprachgebrauch der Domäne auf. Daher soll eine generelle Tendenz – $p_{\text{out}} \in [0,1]$ – berücksichtigt werden, mit der eine Übereinstimmung nicht als Treffer klassifiziert wird. Wir folgen damit der generellen Idee von *unbedingten Wahrscheinlichkeiten* (engl.: *Prior Probabilities* [ZL04]). Für die hier betrachteten Anwendungsfälle hat sich ein $p_{\text{out}} = 0,2$ als geeignet erwiesen.

Zwei weitere Aspekte, die sich aus der konkreten Konstellation von Treffern im Text ergeben, sind zu betrachten: Zum einem sollen auch Treffer, die kürzer als die längste Übereinstimmung von Text- und Referenzdatensatz sind, berücksichtigt werden und müssen dementsprechend bewertet werden (siehe Kap. 5.1.1). Zum anderen überlagern sich Übereinstimmungen mit verschiedenen Attribut-N-Grammen häufig. Im laufenden Beispiel *„The configuration adaptor framework of XI“* überschneiden sich z. B. die Übereinstimmungen mit $\tilde{a}_1 =$ *„Configuration Adapter“* und $\tilde{a}_2 =$ *„Adapter Framework“*. Die Bedeutung der Phrase *„configuration adaptor framework“* ist in diesem Kontext unklar, so dass beide Treffer dementsprechend niedriger gewichtet werden sollten.

Zur Bestimmung von sich überschneidenden Treffern werden bereits zum Zeitpunkt der Erzeugung von Text-N-Grammen mögliche Überschneidungen erfasst. Attribut-N-Gramme werden im Resultat der Anfrage entsprechend der vorher gespeicherten, möglichen Überschneidungen mit anderen Treffern gruppiert. Sei $s'(\tilde{a})$ die Textsequenz, in der überlappende Treffer identifiziert wurden wie z. B. $s'(\tilde{a}) =$ *“configuration adaptor framework“* für die Übereinstimmung

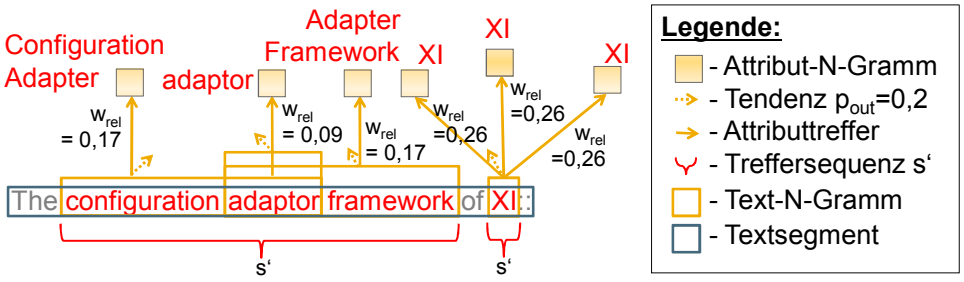


Abb. 5.3: Relevanzgewichte für Beispiel aus Abb. 5.1

mit $\tilde{a}$ ="*Configuration Adapter*" (siehe Abb. 5.3), so wird das Gewicht von $\tilde{a}$ entsprechend des Verhältnisses von Tokenanzahl in Attribut-N-Gramm und Textsequenz skaliert: $|\tilde{a}|/|s'(\tilde{a})|$. Sei weiterhin $N(s'(\tilde{a}))$ die absolute Anzahl von Attribut-N-Grammen, die mit der Sequenz $s'(\tilde{a})$ übereinstimmen, so kann das Relevanzgewicht – notiert als $w_{\text{rel}}(\tilde{t}, \tilde{a}) \in [0,1]$ – für eine Attributübereinstimmung wie folgt beschrieben werden:

$$w_{\text{rel}}(\tilde{t}, \tilde{a}) = \frac{|\tilde{a}|}{|s'(\tilde{a})|} \cdot \frac{(1 - p_{\text{out}})}{N(s'(\tilde{a}))}, \quad \tilde{t} \equiv \tilde{a} \tag{5.2}$$

wobei der linke Teil von Gleichung 5.2 die Relevanz im Kontext von sich überschneidenden und einander enthaltenden Treffern im untersuchten Textsegment bewertet und der rechte Teil die Ein- bzw. Mehrdeutigkeit von Treffern im Kontext der Referenzdaten sowie die Tendenz $p_{\text{out}} \in [0,1]$ berücksichtigt, dass eine Übereinstimmung nicht als Treffer klassifiziert wird.

Wir merken an, dass in Gleichung 5.2 keine logarithmische Glättung von $N(s'(\tilde{a}))$ vorgenommen wird, wodurch mehrdeutige Treffer stark diskriminiert werden. Da Phrasen, die in den Referenzdaten äußerst mehrdeutig sind, auch in Dokumenten häufig auftreten (da sie im normalen Sprachgebrauch der Domäne verwendet werden), kann dieser diskriminierende Charakter durch die Trefferfrequenz im Dokument gegebenenfalls ausgeglichen werden (siehe Kap. 5.1.3).

In Abb. 5.3 sind die Werte von w_{rel} für das laufende Beispiel aus Abb. 5.1 dargestellt. Ein Wert von $p_{\text{out}} > 0$ stellt unter anderem sicher, dass mehrere Treffer, die dem selben Attribut im Referenzdatensatz zugeordnet werden können, aber in anderer Reihenfolge auftreten, ein geringeres aggregiertes Gewicht erhalten als zusammenhängende Treffer.

Abdeckung von Attribut-N-Grammen. Nachdem die Ähnlichkeit und Relevanz von Text- und Attribut-N-Grammen bestimmt wurde, bewerten wir zum Abschluss die Abdeckung eines Attribut-N-Gramms bezüglich des ursprünglichen Attributinhalts. Dieses Bewertungskriterium ist grundsätzlich unabhängig vom Text und wir daher als $w_{\text{cov}}(a, \tilde{a}) \in [0,1]$ notiert. Zur Berechnung verwenden wir das Verhältnis der Tokenanzahl in Attribut-N-Gramm und Attribut:

$$w_{\text{cov}}(a, \tilde{a}) = \frac{|\tilde{a}|}{|a|} \cdot p_{\text{T}}, \tag{5.3}$$

wobei $p_{\text{T}} \in [0,1]$ einen Faktor zur Gewichtung einer bestimmten Attributklasse spezifiziert. Zum Beispiel können Attribute, die weniger zur Identifikation, jedoch zur Disambiguierung geeignet sind (z. B. Quantitäten von Bestellpositionen), eine geringe Gewichtung erhalten.

Wir nehmen im Folgenden der Einfachheit halber eine Konfiguration mit $p_T = 1$ an. Die Abdeckung $w_{cov}(a, \tilde{a})$ wird zum Zeitpunkt der Erzeugung von Attribut-N-Grammen berechnet, im Index gespeichert und mit den Anfrageresultaten dem Bewertungsverfahren zur Verfügung gestellt.

Konfidenzgewicht für Treffer innerhalb eines Textsegments

Gewichte für einzelne, eventuell im Textsegment verteilte Attributübereinstimmungen werden zu einem aggregierten Konfidenzgewicht – notiert als $w_{conf}(a, t)$ – für das Auftreten eines Attributs a in einem Textsegment t zusammengeführt, welches als Grundlage für die spätere Bewertung von Entitäten dient (siehe Kap. 5.1.3). Jede einzelne Übereinstimmung eines Text- und Attribut-N-Gramms $\tilde{a} \equiv \tilde{t}, \tilde{a} \in a, \tilde{t} \in t$ wird zur Bewertung des Attributs a berücksichtigt. Die Summe der Gewichte normieren wir anhand der Anzahl der überschneidenden Treffersequenzen in einem Textsegment (notiert als $|S'|, S' \leftarrow \bigcup_{\tilde{t} \in t, \tilde{a} \in a, \tilde{a} \equiv \tilde{t}} s'(\tilde{a})$), die bei der Berechnung der Werte für $w_{rel}(\tilde{a}, \tilde{t})$ erfasst wurde, so dass $w_{conf}(a, t) \in [0, 1]$ gilt. Dies führt zu folgender Gleichung:

$$w_{conf}(a, t) = \sum_{\tilde{a} \equiv \tilde{t}, \tilde{a} \in a, \tilde{t} \in t} \frac{w_{sim}(\tilde{a}, \tilde{t}) \cdot w_{rel}(\tilde{a}, \tilde{t}) \cdot w_{cov}(a, \tilde{a})}{|S'|}. \tag{5.4}$$

Das so berechnete Konfidenzgewicht dient als Basis für die Bewertung von Entitäten im Kontext des Gesamtdokumentes.

5.1.3 Bewertung von Entitäten im Kontext des Gesamtdokuments

Im Folgenden wird das Graph-basierte Modell zur Bewertung im Kontext des Dokuments beschrieben. Wir gehen zuerst auf die Abbildung von Text- und Referenzdaten, auf das Datenmodell von *RapidUIM* ein, und diskutieren darauf aufbauend das Bewertungsschema.

Graph-basiertes Datenmodell

Die Bewertung von Entitäten basiert auf dem Meta-Datenmodell von *RapidUIM*, welches in Kap. 3.1.3 beschrieben wurde. Im Folgenden wird die Anwendung von *Dokument-Konzept-Graphen* für die Bewertung von Entitäten erläutert. Wir wiederholen die wichtigsten Konzepte und erläutern die Lösung des Bewertungsproblems mit DKGs.

Ein Dokument wird als Dokument-Konzept-Graph $DKG = (N, E)$ repräsentiert, wobei N eine Menge von Knoten und $E \subseteq N \times N$ eine Menge von Kanten notiert. Knoten eines DKGs werden in zwei disjunkte Klassen unterschieden, $N = N_D \cup N_S$. Die Knotenmenge N_D bildet syntaktische und N_S semantische Strukturen ab. Kanten $E = E_D \cup E_{D\_S} \cup E_S$ beschreiben die Beziehungen zwischen syntaktischen Strukturen (E_D), die Abbildung von syntaktischen Strukturen auf semantische Konzepte (E_{D_S}), sowie die Beziehungen zwischen semantischen Konzepten (E_S).

Syntaktische Strukturen sind z. B. Textsegmente, Sätze, Absätze oder Dokumentseiten. Höherwertige Strukturen wie Sätze, Absätze oder Dokumentseiten können zur Begrenzung des Bewertungskontextes herangezogen werden. In *RapidUIM* lässt sich eine derartige Eingrenzung des Bewertungskontextes durch Konfiguration von indirekten Datenabhängigkeiten modellieren (z. B. eine Bewertung für einzelne Dokumentseiten).

Auch eine mehrstufige Bewertung und Filterung von Zwischenergebnissen ist ohne weiteres zur Adaption von Konzepten der *dynamischen Programmierung* (siehe z. B. [CGRM06]) möglich. Ein solche Konfiguration ist insbesondere zur Verbesserung der Verarbeitungszeiten bei sehr

großen Dokumenten von Vorteil (z. B. zuerst seitenweise Bewertung und darauf folgende Aggregation für das Gesamtdokument) und profitiert zusätzlich von der Parallelisierung, die in Kap. 3.2 vorgestellt wurde.

Im Folgenden beschränken wir uns auf die Bewertung von Entitäten innerhalb eines „Bewertungskontextes", den wir der Einfachheit halber als „Dokument" bezeichnen. Wir betrachten dementsprechend lediglich die Menge der Knoten N_D eines „Dokuments", die die Textsegmente gemäß der Vorsegmentierung beschreiben, und die Knotenmenge N_S.

Die Knotenmenge N_S entspricht hier der Menge an Attributen und Entitäten, die im Kontext eines Dokuments von Bedeutung sind, und durch Übereinstimmungen zwischen Text und Referenzdaten sowie weitere in Beziehung stehende Entitäten gegeben ist. Die Beziehungen zwischen den semantischen Strukturen, die durch den Referenzdatensatz definiert sind, werden durch gerichtete Kanten $E_S \subseteq N_S \times N_S$ repräsentiert.

Wie im Beispiel des DKGs aus Abb. 3.3 in Kap. 3.1.3 eingeführt, wird für jedes Attribut, für das eine Bewertung $w_{\text{conf}}(a, s) > 0$ gemäß Gleichung 5.4 vorliegt, genau ein Knoten $a \in N_S$ im DKG erzeugt. Die Kanten E_{D_S} setzen Textsegmente und Attribute gemäß der Übereinstimmungen der Token-N-Gramme in Beziehung. Weiterhin werden für die zu Attributen gehörigen Entitäten Knoten erzeugt. Wir notieren eine Entität im Folgenden als $\varepsilon \in N_S$. Eine Kante $e \in E_S$ bildet die Zugehörigkeit eines Attributs $a \in N_S$ zu einer Entität $\varepsilon \in N_S$ ab.

Im weiteren Aufbau des DKG werden die Beziehungen zwischen Entitäten berücksichtigt. Dabei werden entsprechend der Konfiguration indirekt betroffene Entitäten zu den bereits im DKG vorhandenen Knoten ergänzt, aber auch die Beziehungen zwischen den bereits vorhanden DKG-Knoten abgebildet. Im Falle von SAPTerm werden für Knoten, die eine Abkürzung repräsentieren, entsprechende Knoten für technische Konzepte, für technische Konzepte die entsprechenden Softwaremodule und weiterhin die relevanten Entitäten entsprechend der Softwaremodulhierarchie rekursiv erzeugt.

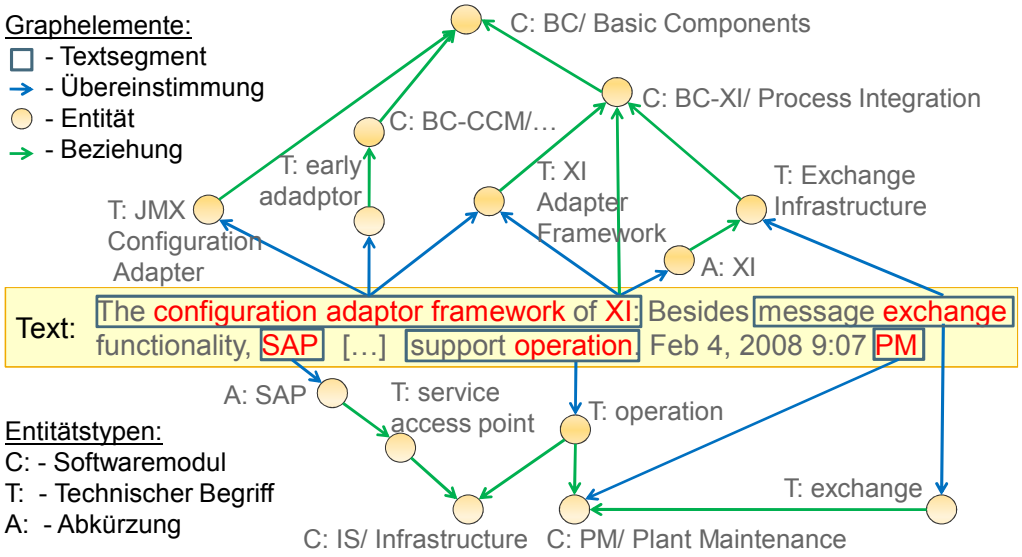


Abb. 5.4: Beispiel für Graph-basiertes Datenmodell zum Ranking von Entitäten

Ein Beispiel-DKG, der die Beispiele aus Kap. 5.1.2 erweitert, ist in Abb. 5.4 dargestellt. Der Übersichtlichkeit halber verzichten wir in Abb. 5.4 auf die Darstellung der Attributknoten und verbinden Knoten für Textsegmente und Entitäten direkt. Ein DKG bildet das Dokument in einen Teilgraphen des Referenzdatensatzes ab. Das Problem besteht nun darin die relevantesten Bestandteile des DKG zu filtern.

Mehrere Treffer im Kontext einer Entität ε führen zu einem erhöhten Grad an Gewissheit, dass ε im Text referenziert wurde, wie z. B. anhand der Softwaremodule $\varepsilon_1 \equiv$*„BC/ Basic Components“* und $\varepsilon_2 \equiv$*„PM/ Plant Maintenance“* dargestellt ist. Beide Entitäten werden jedoch in diesem Beispiel durch Mehrdeutigkeiten bzw. weniger relevante Übereinstimmungen in den DKG abgebildet. Der Text in Abb. 5.4 referenziert ganz klar das Softwaremodul $\varepsilon_3 \equiv$*„BC-XI/ Process Integration“* und dessen Konfiguration im Kontext eines Nachrichtenaustauschs zwischen verschiedenen SAP-Systemen. Das Modul *„PM“* vereinigt weniger Treffer als *„BC-XI“* und ist daher geringer zu gewichten. *„BC“* ist das Elternelement von *„BC-XI“*. Die Distanz von Textsegmenten zur Entität *„BC“* im DKG ist größer als im Falle von *„BC-XI“*, *„BC“* daher geringer zu gewichten. Wir betrachten die genannten Problemstellungen zur Bewertung von Entitäten für das Dokument im Folgenden.

Bewertung von Entitäten mit Dokument-Konzept-Graphen

Ein aggregiertes Gewicht für direkt durch Attributübereinstimmungen referenzierte Entitäten könnte einfach durch Aufsummieren der zugehörigen Konfidenzwerte berechnet werden. Zur Berücksichtigung von indirekten Referenzen verwenden wir jedoch eine Art *Spreading-Activation*-Algorithmus (siehe z. B. [TVA07]) zur Propagation von Konfidenzgwichten im DKG. Dazu betrachteten wir Pfade zwischen Textsegmenten und Entititäten.

Sei $p(\varepsilon, t)$ ein Pfad im DKG zwischen dem Startknoten $t \in N_D$ (Textsegment) und dem Endknoten $\varepsilon \in N_S$ (Entität). Sei weiterhin $e_1, \ldots, e_n$ die Sequenz von Kanten des DKGs, die in dem Pfad $p(\varepsilon, t)$ enthalten sind, wobei t der Startknoten der Kante e_1 und ε der Endknoten von e_n ist. Dann kann der Beitrag von $p(\varepsilon, t)$ zur Relevanz der Entität ε im Kontext des untersuchten Dokuments als Produkt der Kantengewichte des Pfades ausgedrückt werden. Kantengewichte im DKG bewerten wir wie folgt:

$$w(e_k) = \begin{cases} w_{\text{conf}}(a, t), k = 1 \text{ (siehe Gleichung 5.4)}, \\ 1 - p_{\text{self}}, k \neq 1, \end{cases} \tag{5.5}$$

wobei $p_{\text{self}} \in [0, 1]$ eine konfigurierbare Tendenz beschreibt, die die Propagation von Gewichten im DKG abschwächt. Der Parameter p_{self} kann demnach als Gewicht verstanden werden, das definiert, inwieweit eine eingehende Kante eines Knotens nur für ihn und nicht für weitere Knoten entlang der Kantensequenz des Pfades $p(\varepsilon, t)$ von Bedeutung ist.

Sei $P(\varepsilon, t)$ die Menge aller Pfade zwischen einer Entität ε und einem Textsegment t in einem DKG und $|N_D|$ die Anzahl von Textsegmenten in einem Dokument, so kann die Bewertung einer Entität auf Basis der Summe der Gewichte aller Pfade aller Textsegmente $P(\varepsilon, t_1), \ldots, P(\varepsilon, t_n)$ ausgedrückt werden. Die Bewertung einer Entität $\varepsilon \in N_S$ für ein Dokument (repräsentiert durch die Textsegmente N_D) entspricht demnach:

$$s(\varepsilon, N_D) = \sum_{\mathrm{t} \in \mathrm{N_D}} \sum_{\mathrm{p} \in \mathrm{P}(\varepsilon, t)} \frac{1}{|N_D|} \prod_{e_k \in \mathrm{p}} w(e_k), \tag{5.6}$$

wobei $1/|N_D|$ der Normierung dient. Anzumerken ist, dass nicht jeder einzelne Pfad gesondert berechnet wird, sondern alle Gewichte in einer Traversion des DKG bestimmt werden. Ergebnis der Bewertung ist eine nach Gewichten geordnete Menge von semantisch bedeutsamen DKG-Knoten. Die Ordnung der Knoten beschreibt das aggregierte Gewicht der DKG-Teilgraphen, die durch eine Entität subsumiert werden, einschließlich relevanter Textsegemente.

Dementsprechend sind im Falle von Mehrdeutigkeiten die korrekten Zuordnungen von Textsegmenten zu Attributen in den Teilgraphen der am höchsten bewerteten Entitäten zu erwar-

ten. Da zum Teil mehrere Pfade zwischen einem Textsegment und einer Entität existieren (siehe z. B. Teilgraph von $\varepsilon \equiv$„*BC-XI*“ in Abb. 5.4), ist jeweils das Attribut mit der am höchsten bewerteten ausgehenden Kante des Textsegments im Teilgraphen der betrachteten Entität als wahrscheinlichste Bedeutung heranzuziehen.

Für das Beispiel aus Abb. 5.4 wird das Sofwaremodul „*BC-XI*“ wesentlich höher gewichtet als das übergeordnete „*BC*“. Das Modul „*PM*“, dessen „übereinstimmender“ Kurzname im Beispiel eigentlich einem Zeitstempel entspricht, erhält aufgrund weniger Treffer im Kontext der Referenzdaten eine äußerst geringe Gewichtung. Da „*BC-XI*“ am höchsten gewichtet wird, kann das Segment „*configuration adaptor framework of XI*“ am wahrscheinlichsten dem technischen Begriff „*XI Adapter Framework*“ zugeordnet werden. Für das Segment „*message exchange*“ ist eine Bedeutung im Kontext des technischen Konzepts „*Exchange Infrastructure*“ naheliegender als z. B. die Bedeutung „*exchange*“ im Kontext des Moduls „*PM*“, obwohl „*Infrastructure*“ im Text nicht erwähnt wird.

Verwendung von Identifikationsergebnissen

Die oben beschriebene Methode kann im Kontext vieler Anwendungsklassen eingesetzt werden. Im Folgenden gehen wir auf die Abbildung von Einzeldokumenten auf Referenzdaten, wie sie z. B. im Falle der (semi-)automatischen Rechnungsverifikation (*Szenario I* in Kap. 1.2) notwendig ist, und die Anwendung zur semantischen Dokumentsuche ein (siehe *Szenario II* in Kap. 1.2).

Abbildung von Einzeldokumenten in die Referenzdaten. Zur Realisierung der in Kap. 1.2 diskutierten Verarbeitung von Rechnungen mit Hilfe von strukturierten Bestelldaten werden dem Nutzer jeweils Dokumentteile entsprechend der syntaktischen Knoten im DKG und deren Zuordnungen zu semantischen Konzepten zur Verifikation vorgeschlagen.

Bei der Konstruktion des DKGs werden den Entitäten jeweils syntaktische Knoten auf Basis der aggregierten Positionsindizes der subsumierten Textsegmente zugewiesen (siehe auch einführendes Beispiel in Abb. 3.3/ Kap. 3.1.3). Dem Nutzer werden, beginnend von der Zuordnung zwischen Bestellung und Dokument, die jeweils am höchsten bewerteten Entitäten für eine syntaktische Einheit präsentiert.

Wenn eine Zuordnung durch den Nutzer verifiziert wurde, wird die Prozedur rekursiv für den Teilgraphen der ausgewählten Entitäten fortgesetzt. Ergebnis ist im Falle der Rechnungsverarbeitung eine neu angelegte elektronische Rechnung, welche auf der Identifikation der subsumierten Entitäten der Bestellung basiert.

Semantische Dokumentsuche. Zur Dokumentsuche können die Identifikationsergebnisse wie folgt verwendet werden: Die am höchsten gewichteten Entitäten im DKG entsprechen einer Art Klassifikation für ein Dokument. Die durch sie subsumierten DKG-Teilgraphen beschreiben andere relevante Entitäten und disambiguierte Übereinstimmungen. Die Verwendung der DKG-Teilgraphen zur Repräsentation von Dokumenten in der Dokumentsuche verspricht daher eine höhere Genauigkeit und Vollständigkeit, da Dokumente in die Unternehmenssemantik eingeordnet und mehrdeutige Wortgruppen disambiguiert werden.

Da in den hier disktuierten Anwendungsfällen sehr viele Übereinstimmungen zwischen Text und Referenzdaten entstehen, ist eine Selektion der relevantesten DKG-Teilgraphen sinnvoll. Für ein Dokument werden die am höchsten bewerteten Entitäten sowie die Knoten in den durch sie subsumierten DKG-Teilgraphen mit ihren Gewichten gespeichert. Nutzeranfragen werden durch einen Extraktionsdienst in gleicher Weise wie Dokumente analysiert.

In der Dokumentsuche muss die Ähnlichkeit von Anfrage- und Dokumenttexten (bzw. der zugehörigen DKG-Teilgraphen) berechnet werden. Die oben beschriebenen Gewichte der Entitäten kodieren implizit die Struktur der relevanten DKG-Teilgraphen. Demnach können An-

frage und Dokumente als Vektoren von Entitätsgewichten dargestellt werden. Sei V_d der Vektor von Entitätsgewichten eines Dokuments und V_q ein entsprechender Anfragevektor, dann kann die Ähnlichkeit z. B. auf Basis des Kosinus-Ähnlichkeitsmaß wie folgt bestimmt werden:

$$sim(V_q, V_d) = cos(\theta) = \frac{\langle V_q, V_d \rangle}{||V_q||||V_d||,} \tag{5.7}$$

wobei $\langle V_q, V_d \rangle$ das euklidische Skalarprodukt und $||V_q||$ und $||V_d||$ die entsprechenden Vektornormen beschreiben. Auch eine Adaption von Ähnlichkeitsmaßen, die wie *TF-IDF* Korpusstatistiken berücksichtigen, ist möglich. Statt der Termfrequenz kann als lokales Bewertungsmaß das Entitätsgewicht angenommen werden. Die IDF-Komponente könnte beispielsweise statt der Anzahl von Dokumenten, die einen Term enthalten, die Summe der Gewichte für eine Entität über den Korpus berücksichtigen.

5.1.4 Zusammenfassung

In Kap. 5 wurde ein Ansatz zur Identifikation von Entitäten, die in Graph-struktuirerten Referenzdaten vordefiniert sind, in Texten vorgestellt. Der Ansatz ist insbesondere für Entitäten konzipiert, deren Attribute eher beschreibenden Charakter haben. Korrespondenzen werden auf Basis einer Vorsegmentierung des Textes und Token-N-Gramm-Bildung von Segmenten und Attributinhalten herbeigeführt. Zur Beschleunigung des Abgleichs werden die Attribut-N-Gramme präfixkodiert indexiert. Das Vernachlässigen von extrem mehrdeutigen N-Grammen führt zu einer geringen Steigerung der Indexgröße im Vergleich zu einem einfachen Tokenindex, bei allen Vorteilen, die eine Betrachtung von N-Grammen statt Token (wie z. B. in der Relevanzbewertung für Wortgruppen) mit sich bringt.

Übereinstimmungen von Text- und Attribut-N-Grammen werden bezüglich ihrer Ähnlichkeit auf Zeichenebene und einem Relevanzmaß, das die Ein- bzw. Mehrdeutigkeit von Attribut-N-Grammen sowie die konkrete Treffersituation im Text berücksichtigt, bewertet. Zusätzlich wird die Abdeckung von Attribut-N-Grammen in Bezug auf die ursprünglichen Attributinhalte betrachtet. Ein Konfidenzgewicht, das sich aus den einzelnen Gewichten zusammensetzt, wird zur Bewertung von Attributtreffern im Kontext eines Textsegements berechnet.

Die Konfidenzgewichte werden in einem Dokument-Konzept-Graphen, der Textsegmente, übereinstimmende Attribute und relevante Entitäten aus dem Graphen der Referenzdaten zusammenfasst, auf Basis eines *Spreading-Activation*-Algorithmus propagiert und zu Gewichten für Entitäten aggregiert. Die Gewichte berücksichtigen mehrere Attribute einer Entität sowie Mehrfachvorkommen einzelner Attribute im Text. Sie bewerten nicht nur die Relevanz einer Entität für den Text, sondern vielmehr den durch sie subsumierten Teilgraphen des Dokument-Konzept-Graphen.

Die entstehende Ordnung von Teilgraphen des Referenzdatensatzes ermöglicht viele neuartige Anwendungszenarien, wie z. B. eine semantische Dokumentsuche, die mit bisherigen Methoden nicht ohne weiteres in der beschriebenen Art und Weise realisiert werden könnten.

5.2 Experimentelle Evaluation

Zum Nachweis der praktischen Anwendbarkeit des in Kap. 5.1 vorgestellten Verfahrens untersuchen wir im Folgenden die Qualität des *Rankings* von Entitäten für Dokumente. Der Überblick zu verwandten Arbeiten in Kap. 2.3 zeigte bereits, dass vorherige Ansätze lediglich Teilprobleme der in Kap. 1.2 diskutierten Anwendungsfälle adressieren. Verwandte Ansätze verwenden zumeist *TF-IDF*-Wortgewichte zur Bewertung von Übereinstimmungen. Um eine vergleichende Evaluation vorzunehmen, wurden daher die *TF-IDF*-basierten Ansätze von [MK07] und [CGRM06] in die hier dargestellten Problemstellungen adaptiert.

In [MK07] wird unter anderem die Aggregation von *TF-IDF*-Gewichten für Übereinstimmungen von Textfragmenten und verschiedenen Attributen von Tabellen-artig organisierten Entitäten vorgeschlagen. Wir adaptieren dieses Verfahren als „naiven" Ansatz für unsere Experimente und verwenden lediglich die Attribute der evaluierten Enititätstypen (z. B. von Softwaremodulen in SAPTerm).

Im Ansatz von [CGRM06] werden alle *TF-IDF*-Gewichte für Übereinstimmungen aller Entitätstypen im Referenzdatensatz bezüglich ihrer zugeordneten „Wurzelentitäten" in der Graphstruktur des Referenzdatensatzes aggregiert. Zusätzlich wird für zwei Anwendungsfälle, in denen eine ausgeprägte Produkt-Teil-Hierarchie vorlag, eine Erweiterung von [CGRM06] betrachtet, die einen Abschwächungsfaktor bei der Aggregation von *TF-IDF*-Gewichten im DKG berücksichtigt.

Zur Evaluation werden drei Dokumentenkorpora herangezogen. Für einen dieser Korpora stehen eine große Menge an Trainingsdaten zur Verfügung. Für einen anderen Korpus sind die Referenzdaten in Form von HTML-Seiten im WWW vorhanden. Für den Korpus mit Trainingsdaten vergleichen wir unser Verfahren zusätzlich zu einem maschinellen Lernverfahren [BAR06, FZ07]. Für den Referenzdatensatz, der im WWW verfügbar ist, adaptieren wir eine Websuchmaschine[6] zur Bewertung von Entitäten.

Die Experimente zur Websuchmaschine werden durch die Ergebnisse von [LYMC06] motiviert, die zeigten, dass Websuchmaschinen zum Teil ein besseres *Ranking* von Entitäten für Stichwortanfragen als z. B. [ACD02, HP02] liefern, insofern alle zu einer Entität in Beziehung stehenden Informationen auf einer Webseite zusammengefasst sind. Die Webseite repräsentiert in diesem Fall das „Profil" der Entität.

Das in Kap. 5.1 beschriebene *Rankingverfahren* wurde als IE-Modul in die *RapidUIM*-Plattform integriert. Das IE-Modul nutzt Standardeinstellungen für Parameter, kann aber anderweitig konfiguriert werden. Es stehen mehrere Adapter für verschiedene Typen von Referenzdaten zur Verfügung. Für die Experimente wurde ein Datenbankadapter verwendet, der durch eine Beschreibung der Graphstruktur (Tabellen sowie Primär und Fremdschlüsselbeziehungen) und die zu indizierenden Attributnamen konfiguriert wird.

Zur Implementation von alternativen *TF-IDF*-basierten Ansätzen wurde Lucene[7] und zur Implementation des maschinellen Lernansatzes die LingPipe NLP API[8] verwendet. Für die Vorsegmentierung des Textes verwenden wir den *Stanford Part-of-Speech Tagger* [TKMS03]. Die Experimente wurden auf einem Standard-Desktoprechner mit 2GHz CPU und 3GB Hauptspeicher durchgeführt. Für jedes Dokument wurde der komplette Dokument-Konzept-Graph in einer MySQL 5.1 Datenbank[9] gespeichert und das *Ranking* von Entitäten mit einem Goldstandard verglichen.

In Kap. 5.2.1 werden die in der Evaluation verwendeten Datensätze, Goldstandards und Evaluationsmetriken eingeführt. Die Experimente und Resultate der vergleichenden Evaluation werden in Kap. 5.2.2 diskutiert. In Kap. 5.2.3 betrachten wir den Einfluss des Ähnlichkeitsschwellwertes $s \in [0,1]$ und des Parameters $p_{\text{self}} \in [0,1]$, der die Kantengewichte im DKG hinsichtlich des *Spreading Activation*-Verfahrens bestimmt, auf die Bewertung von Entitäten.

5.2.1 Datensätze und Evaluationsmetriken

Kapitel 5.2.1 gliedert sich wie folgt: Zu Beginn werden die verwendeten Dokumentenkorpora, Referenzdatensätze sowie deren Charakteristiken vorgestellt. Abschließend führen wir die Evaluationsmetriken ein.

[6] Google: `http://google.com`, Stand: 1.06.2009
[7] `http://lucene.apache.org/`, Stand: 1.06.2009
[8] `http://alias-i.com/lingpipe`, Stand: 1.06.2009
[9] `http://dev.mysql.com/`, Stand: 1.06.2009

Dokumentenkorpora

Dokumentenkorpus „Filmbewertungen". Als Anwendungsszenario, das verwandt dem *Szenario I* aus Kap. 1.2 ist, wird die Zuordnung von Filmtiteln zu Filmbewertungen betrachtet. Der Dokumentenkorpus für diesen Anwendungsfall wurde auf Basis von 214 Bewertungen[10] der populärsten Filme der Jahre 1920 bis 1990 erstellt. Die Filmtitel am Beginn der Bewertungen wurden entfernt und als Goldstandard zur Evaluation verwendet. Alle anderen Filmtitel in den Texten, inklusive vorheriger Filme von Darstellern, wurden im Korpus belassen. Die Filmbewertungen wurden weiterhin in Teile zu je 8 Sätzen zerlegt, so dass die beinhalteten Substantivgruppen einfach als Stichwortanfragen für eine Websuchmaschine aufbereitet werden können. Als Resultat standen ca. 1000 Dokumente[11] mit zugehörigem Filmtitel für die Evaluation zur Verfügung. Ziel ist es, einem Dokument den ursprünglichen Filmtitel auf Basis von Erscheinungsjahr, Darstellern, Produktionsfirmen etc. zuzuweisen.

Dokumentenkorpus „Foreneinträge". Der Anwendungsfall im Kontext des SAP Community Network (SCN) zielt auf eine semantische Anreicherung von Dokumenten ab (*Szenario II* in Kap. 1.2), um das *Ranking* von Dokumenten zu verbessern und eine Navigation in Ergebnislisten mit Hilfe von Facetten für Softwaremodule zu unterstützen. Eine möglichst exakte Zuordnung von Foreneinträgen zu Softwaremodulen ist zur Realisierung dieses Anwendungsfalls essentiell. Der Dokumentenkorpus[11] hat eine Größe von 100 zufällig ausgewählten Foreneinträgen (Forumfragen und -antworten). Experten für SAP-Technologien erstellten für jeden Foreneintrag eine geordnete Liste von relevanten Softwaremodulen der Länge n, die im Folgenden durch U_n notiert wird. Die Länge von U_n lag bei $n_{\min} = 2$, $n_{\max} = 10$ und $\bar{n} \approx 3.6$.

Dokumentenkorpus „Kundenanfragen". Der Anwendungsfall, der als *Szenario III* in Kap. 2.3 diskutiert wurde, erfordert eine exakte Zuordnung von Kundenanfragen zu Softwaremodulen. Der Dokumentenkorpus zur Evaluation dieses Anwendungsfalls hat eine Größe von 700 abgeschlossenen Kundenanfragen. Jedes Dokument wurde durch die SAP Kundenbetreuung genau einem Softwaremodul zugeordnet. Anzumerken ist, dass auch mehrere Softwaremodule im Kontext einer Anfrage betroffen sein können. Es wird jedoch lediglich die relevanteste im System erfasst. Zusätzlich steht eine Menge von 7.000 Dokumenten ($\approx$ 5.000.000 Wörter) zum Training eines maschinellen Lernverfahrens zur Verfügung. Der Trainingskorpus enthält lediglich Dokumente für 1.500 Softwaremodule (also 10% aller Module). Aus dem Testkorpus wurden dementsprechend zuvor alle Dokumente entfernt, die Softwaremodule referenzieren, für die weniger als 50 Trainingsdokumente zur Verfügung standen.

Referenzdatensätze

Referenzdatensatz Internet Movie Database. Die Internet Movie Database (IMDB) wird als Referenzdatensatz für den Dokumentenkorpus „Filmbewertungen" verwendet. Die im WWW zur Verfügung stehenden Textexporte der IMDB[12] wurden zu diesem Zweck mit IMDbPY[13] geparst und in einem RDBMS gespeichert. Die in der Evaluation verwendeten Tabellen der Datenbank enthalten 1.224.983 Filmnamen (Name und Erscheinungsjahr), 1.224.983 alternative Filmnamen (im Sinne von *„auch bekannt als"*), 2.296.365 Personennamen (Vor- und Nachname in einer Spalte), 513.112 alternative Personnennamen (Spitznamen von Schauspielern), 2.075.439 Rollennamen und 194.057 Unternehmen. Neben den Primär-Fremdschlüsselbeziehungen zwischen den Tabellen für alternative und korrekte Namen existieren 785.818 Beziehungen zwischen Filmnamen (z. B. *„remake of"*,*"spin off from"* etc.) und 16.421.791 n-zu-m

[10] `http://www.filmsite.org/allfilms2.html`, Stand: 1.06.2009
[11] Verfügbar unter: `http://ev.brauer-info.de/gs.zip`, Stand: 01.10.2010
[12] `http://www.imdb.com/interfaces#plain`, Stand: 01.06.2009
[13] `http://imdbpy.sourceforge.net`, Stand 01.05.2009

Beziehungen zwischen Unternehmen, Personen und Filmen. Das IE-Modul wurde so konfiguriert, dass es die Beziehungstabelle als jeweils einem Beziehungstyp von Unternehmen zu Filmen sowie von Personen zu Filmen interpretiert.

Referenzdatensatz SAP Terminologie. Die englische Version der SAP Terminologie (SAP-Term) wird als Referenzdatensatz für die Dokumentenkorpora „Foreneinträge“ und „Kundenanfragen“ verwendet, um Softwaremodule für einen gegebenen Text zu identifizieren. Sie enthielt zum Zeitpunkt der Experimente: 16.117 Softwaremodule mit jeweils einem Attribut für Kurz- und Langname, 112.381 technische Konzepte, die Softwaremodulen zugeordnet sind und 20.579 orthographische Varianten und Abkürzungen von technischen Konzepten. Neben den Beziehungen zwischen den verschiedenen Entitätstypen existieren 15.789 Beziehungen zwischen Softwaremodulen, die deren Hierarchie beschreiben. Die Synonymbeziehungen zwischen technischen Begriffen werden ignoriert.

Evaluationsmetriken

Wir evaluieren hier die korrekte Zuordnung von Dokumenten zu Entitäten eines kontrollierten Vokabulars [Mud98]. Prinzipiell handelt es sich dabei um die inverse Problemstellung im Vergleich zum *Information Retrieval*. Statt von einer Nutzeranfrage auf eine geordnete Liste von Dokumenten zu schließen, ist hier eine geordnete Liste von Entitäten für ein Dokument gesucht.

Zur Evaluation von *Information Retrieval*-Systemen mit geordneten Resultatlisten wurden im Rahmen der *Text REtrieval Conference (TREC)* die Maße *Genauigkeit* (P_k) und *Vollständigkeit an Position k* (R_k) vorgeschlagen (siehe [MRS08]). Im Folgenden bezeichnet k, wenn nicht anders erläutert, die Rankposition eines Resultats. Wir betrachten im Folgenden alle k, $1 \leq k \leq 10$. Bei der Bestimmung von P_k und R_k werden die Liste der vom Nutzer als relevant deklarierten Objekte – notiert als $\mathcal{U}_\mathrm{n}$ – und eine entsprechende vom System erstellte Liste $\mathcal{L}$ von Objekten verglichen.

Zur Evaluation wird die Systemliste an Position k abgeschnitten, welche im Folgenden als $\mathcal{L}_\mathrm{k}$ notiert wird. Für jede Position k werden R_k und P_k bestimmt. Im Falle, dass mehrere Testreihen mit verschiedenen Anfragen (hier Dokumente) evaluiert werden, wird das arithmetische Mittel der einzelnen Werte für die Position k ermittelt. Zur Berücksichtigung der Ordnung in der Nutzerliste werden hier R_k und P_k mit einer leichten Modifikation verwendet. Dazu wird auch die Nutzerliste $\mathcal{U}_\mathrm{n}$ an Position k abgeschnitten und die Schnittmenge von $\mathcal{L}_\mathrm{k}$ und $\mathcal{U}_{\min(\mathrm{n},\mathrm{k})}$, im Folgenden vereinfacht als $\mathcal{U}_\mathrm{k}$ notiert, zur Bewertung herangezogen.

Die Genauigkeit P_k und Vollständigkeit R_k sind wie folgt definiert:

$$P_\mathrm{k} = \frac{|\mathcal{U}_\mathrm{k} \cap \mathcal{S}_\mathrm{k}|}{k}, \quad R_\mathrm{k} = \frac{|\mathcal{U}_\mathrm{k} \cap \mathcal{S}_\mathrm{k}|}{n}. \tag{5.8}$$

Anzumerken ist, dass selbst ein perfektes System nie $R_\mathrm{k} = 1$ für $k < n$ und nie $P_\mathrm{k} = 1$ für $k > n$ erreichen kann. Die Berücksichtigung des *Nutzerrankings* hat den Effekt, dass ein „falsches“ *Ranking* durch einen Anstieg der Genauigkeit P_k für $k < n$ beobachtet werden kann. Ein Beispiel soll diesen Sachverhalt illustrieren. Angenommen ein System ordnet das dem Nutzer nach relevanteste Objekt an Position $k = 2$ und das zweite Objekt der Nutzerliste an Position $k = 1$, so ergibt sich für dieses Beispiel ein $P_1 = 0$ aber $P_2 = 1$.

Die Güte eines Systems kann basierend auf den Werten für R_k und P_k sehr gut mit Hilfe eines Vollständigkeits-Genauigkeits-Graphen (VGG) visualisiert werden (siehe z. B. Abb. 5.7 in der Auswertung). Hierzu werden die Wertepaare $(R_\mathrm{k}, P_\mathrm{k})$ für alle $k, 1 \leq k \leq 10$ in ein gemeinsames Koordinatensystem abgebildet. Der entstehende Graph kann von links nach rechts gelesen werden. Beim ersten Messwertepaar links handelt es sich um die Messung für $k = 1$. Die Er-

gebnisse für $k > 1$ befinden sich rechts von diesem. Es gilt: je größer die Fläche unter einem Graphen des VGG für ein Verfahren, desto besser die Resultate.

Zur Evaluation werden auch Dokumentenkorpora betrachtet, in denen lediglich eine Entität ε pro Dokument im Goldstandard vorgegeben ist. Als Gütemaß wird in diesem Fall die Metrik *Erfolg an Position k* (notiert als S_k) verwendet. Sie beschreibt, an welcher Position k das erwartete Resultat ε in der Systemliste $\mathcal{L}$ auftritt. Für ein Dokument ist S_k wie folgt definiert:

$$S_k = |\{\varepsilon\} \cap \mathcal{L}_k|. \tag{5.9}$$

In der Evaluation wird für S_k das arithmetische Mittel bezüglich aller Dokumente im Korpus dargestellt. Zum Beispiel erreicht ein System, wenn es das relevante Objekt immer an Position $k = 5$ identifiziert, ein $S_1 = 0$ und $S_5 = 1$.

5.2.2 Resultate der vergleichenden Evaluation

Im Folgenden diskutieren wir die verwendete Konfiguration, die Resultate der einzelnen Experimente und den Einfluss der Parameter für den Abschwächungsfaktor p_{self} und den Ähnlichkeitsschwellwert s.

Wenn nicht anders dargestellt, wurden folgende Standardparameter verwendet:

- Tendenz, dass eine Wortgruppe kein Treffer ist: $p_{\text{out}} = 0,2$.
- Gewichtung für Attributtypen: $p_T = 1$.
- Ähnlichkeitsschwellwert für berücksichtigte Attribut-N-Gramme: $s = 0,7$.
- Berücksichtigte Top-k Übereinstimmungen für ein Textsegment: $k = 200$.
- Als Abschwächungsfaktor für die Gewichtspropagation wird das Reziproke der durchschnittlichen Pfadlänge zwischen Blatt- und Wurzelelementen des Referenzdatensatzes gewählt. Für SAPTerm ergibt sich demnach ein $p_{\text{self}} \approx 0,2$ und für die IMDB ein $p_{\text{self}} \approx 0,5$.

Für die *TF-IDF*-basierten Vergleichsverfahren folgen wir dem in EROCS [CGRM06] vorgeschlagen Bewertungsmetriken. Statt alle Top-k Resultate des Wörterbuchabgleichs zu verwenden, werden alle Übereinstimmungen, die ein *TF-IDF*-Gewicht kleiner als ein Wert λ tragen, gemäß [CGRM06] nicht berücksichtigt. Wir verwenden, wie von [CGRM06] vorgeschlagen, ein $\lambda = 4$ für die IMDB. Für die Experimente mit SAPTerm lieferte eine Konfiguration mit $\lambda = 1$ bessere Ergebnisse und wurde dementsprechend so verwendet.

Der Ansatz von [MK07] schlägt eine Aggregation von *TF-IDF*-Gewichten für verschiedene Attributübereinstimmungen einer Entität, ohne Berücksichtigung von weiteren Beziehungen vor. Das Verfahren wird hier, um eine Filterung für die relevantesten Übereinstimmung gemäß [CGRM06] erweitert, als „naives“ Vergleichsverfahren herangezogen.

Experimente für den Dokumentenkorpus „Filmbewertungen“

Im Fall der IMDB existiert für jeden Film eine Homepage im WWW, die alle in Beziehung stehenden Entitäten listet, so dass eine Websuchmachine zur Evaluation genutzt werden kann. Als Websuchmaschine wurde entsprechend den Experimenten von [LYMC06] Google ausgewählt. Es wird angenommen, dass *PageRank* eines der wichtigsten Relevanzmaße von Google ist, welches auch von [BHP04, BHN+02, KPC+05] zum Ranken von Entitäten vorgeschlagen wurde. Die Popularität eines Films wird dabei nicht nur von Referenzdatensatz-internen Links (z. B.

von IMDB-*Homepages* der Schauspieler) bestimmt, sondern auch durch eingehende Links von externen Webseiten (z. B. von Wikipedia).

Dokumente des Goldstandards wurden mit Hilfe eines *Part-of-Speech Taggers* in Stichwortanfragen umgewandelt und an *Googles Webservice API*[14] mit einer Einschränkung auf IMDB-Film-Homepages[15] gesendet. Die Liste der nach Relevanz geordneten Filme (Titel der Homepages) wurde manuell mit dem Goldstandard verglichen. Die Ergebnisse des Experiments sind durch „*G4IMDB*" in Abb. 5.5 gekennzeichnet. Die Ergebnisse des in Kap. 5.1 vorgestellten *Rankings* sind als „*RapidUIM (partielle Prop)*" in Abb. 5.5 dargestellt.

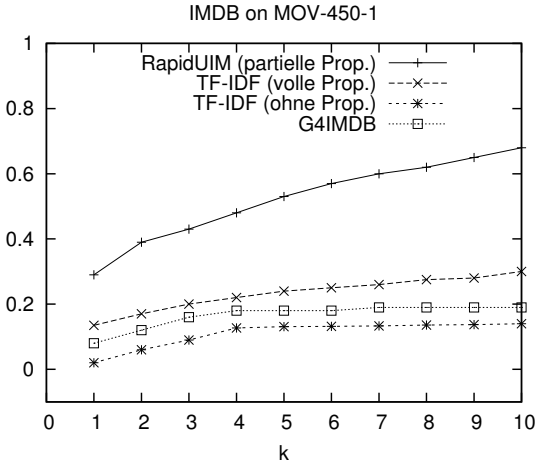


Abb. 5.5: S_k für den Dokumentenkorpus „Filmbewertungen"

Der „naive" Ansatz gemäß [MK07] ist als „*TF-IDF (ohne Prop.)*" illustriert. Es werden lediglich *TF-IDF*-Gewichte für Übereinstimmungen mit der Filmtabelle (Filmname, Erscheinungsjahr) aggregiert. Es erreicht dadurch lediglich ein $S_{10} \approx 0,14$. Die Resultate von „*G4IMDB*" sind im Durchschnitt, bei Betrachtung der ersten drei Rankpositionen, um den Faktor 2 besser als die von „*TF-IDF (ohne Prop.)*".

Der *TF-IDF*-basierte Ansatz gemäß [CGRM06], der als „*TF-IDF (volle Prop.)*" dargestellt ist, liefert um $\Delta S_1 \approx 0,06$ und $\Delta S_{10} \approx 0,11$ bessere Ergebnisse als „*G4IMDB*". Als Schlussfolgerung kann gezogen werden, dass popularitätsbasierte *Rankingverfahren* wie *PageRank* für die Bewertung von Entitäten hinsichtlich eines Dokumentes weniger geeignet sind.

Die Ausnutzung von Wortgruppengewichten statt *TF-IDF*-Wortgewichten und die Berücksichtigung von approximativen Übereinstimmungen in „*RapidUIM (partielle Prop)*" führt zu wesentlich besseren Ergebnissen im Vergleich zu „*TF-IDF (volle Prop.)*". Der Unterschied beträgt $\Delta S_1 \approx 0,16$ und $\Delta S_{10} \approx 0,38$.

Experimente für den Dokumentenkorpus „Kundenbetreuung"

Für den Dokumentenkorpus „Kundenbetreuung" stehen 7.000 Dokumente zum Training eines maschinellen Lernansatzes, der gemäß [BAR06, FZ07] implementiert wurde, zur Verfügung. Der maschinelle Lernansatz basiert auf einem generativen N-Gramm-Sprachmodell, wobei für die Experimente die Länge der Token-N-Gramme zwischen 1 und 10 variiert wurde. Das Ziel des Experiments ist es, zu bewerten, wie sich die in Kap. 5.1 vorgestellte Methode im Vergleich zu einer speziell für den Dokumentenkorpus trainierte Lösung verhält.

[14] `http://code.google.com/intl/en/apis/soapsearch/reference.html`, Stand 01.06.2009
[15] `http://www.imdb.com/title`, Stand 10.06.2009

In Abb. 5.6 sind die erzielten Ergebnisse für ein Unigramm-Sprachmodell (*Unigramm-SM*), Trigramm-Sprachmodell (*Trigramm-SM*) und 5-Gramm-Sprachmodell (*5-Gramm-SM*) dargestellt. *Unigramm-SM* erzielt die schlechtesten Ergebnisse. *Trigramm-SM* und *5-Gramm-SM* erzielen Ergebnisse vergleichbarer Güte. Eine weitere Erhöhung der Tokenanzahl führte zu keiner Verbesserung. Anzumerken ist, dass das 5-Gramm-Sprachmodell, welches im Vergleich zum Unigramm-Sprachmodell statistisch relevante Wortfolgen betrachtet, zu einer durchschnittlichen Verbesserung von $\Delta S_k \approx 0.07$ führt.

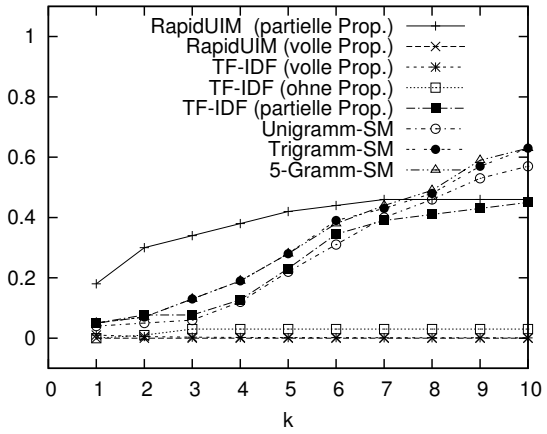


Abb. 5.6: S_k für den Anwendungsfall „Kundenbetreuung"

Der in Kap. 5.1 vorgestellte Ansatz – siehe *„RapidUIM* (partielle Prop.)" in Abb. 5.6 – wird hier mit $s = 0,1$ verwendet, um eine größere Bandbreite an syntaktischen Variationen zuzulassen, und $p_{\text{self}} = 0,1$ um längere Distanzen zwischen Treffern und Entitäten zu „überbrücken" (siehe Kap. 5.2.3 für Details).

Der approximative Abgleich, zusammen mit der Berücksichtigung von semantisch relevanten Wortfolgen und vordefinierten Beziehungen, die im Referenzdatensatz spezifiziert sind, führt im Vergleich zu den Experimenten mit dem Sprachmodell zu wesentlich besseren Ergebnissen für $k = 1, \ldots, 7$. Die Verbesserung beträgt $\Delta S_1 \approx 0,14$ und $\Delta S_2 \approx 0,23$. Dieser Unterschied nimmt für $k > 2$ ab und schlägt für $k > 8$ ins Gegenteil um. Es ist anzunehmen, dass der Grund für diesen Effekt der Umstand ist, dass das Sprachmodell nur für etwa 10% der Softwaremodule trainiert wurde, wogegen *RapidUIM* alle Softwaremodule berücksichtigt.

Neben dem maschinellen Lernansatz wurden die *TF-IDF*-basierten Verfahren evaluiert. Der gemäß [CGRM06] implementierte Ansatz ist als *„TF-IDF (volle Prop.)"* in Abb. 5.6 dargestellt. Es zeigte sich in diesem Experiment, dass der Parameter p_{self} bei der Verwendung von SAPTerm im Gegensatz zur IMDB ein enorm wichtiger Einflussfaktor ist. Zum Vergleich zeigen wir daher *„RapidUIM (volle Prop.)"*, die Konfiguration des hier diskutierten Verfahrens mit $p_{\text{self}} = 0$. In den Resultaten beider Experimente wird keine einzige Entität korrekt identifiziert, da nie Wurzelentitäten der Hierarchie die korrekte Lösung darstellten. Die Resultate zeigen die Notwendigkeit für einen Abschwächungsfaktor bei der Propagation von Gewichten im Graphen.

Als Variante von [CGRM06] wurde eine Aggregation von *TF-IDF*-Gewichten mit Abschwächungsfaktor implementiert. Mögliche Konfigurationen von $p_{\text{self}} \in [0, 1]$ für diesen Ansatz wurden mit einem Intervall von 0,1 getestet. Ein Wert von $p_{\text{self}} = 0,3$ führte zu den besten Ergebnissen, welche in Abb. 5.6 als *„TF-IDF (partielle Prop.)"* dargestellt sind. Die Ergebnisse sind vergleichbar mit denen des Sprachmodells.

Die Resultate des von [MK07] inspirierten *TF-IDF*-Ansatzes sind als *„TF-IDF (ohne Prop.)"* in Abb. 5.6 dargestellt. Die Ergebnisse dieses Experiments zeigen, dass die Ausnutzung von in Beziehung stehenden Entitäten in diesem Anwendungsfall essentiell ist, um akzeptable Ergebnisse zu erreichen.

Experimente für den Dokumentenkorpus „Forensuche“

Für die Experimente zum Anwendungsfall „Forensuche“ standen keine Trainingsdaten zur Verfügung, so dass das Sprachmodell nicht verwendet werden konnte. Wiederum wurden Experimente für die *TF-IDF*-basierten Ansätze durchgeführt (siehe *Vollständigkeit-Genauigkeit-Graph* in Abb. 5.7). In diesem Fall stellte sich $p_{\text{self}} = 0,7$ als beste Konfiguration für *„TF-IDF (partielle Prop.)“* heraus.

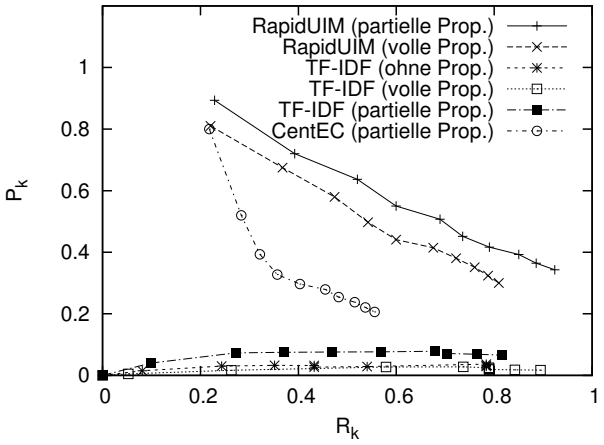


Abb. 5.7: Vollständigkeits-Genauigkeits-Graph für den Anwendungsfall „Forensuche“

Die Resultate in Abb. 5.7 zeigen, dass *„TF-IDF (ohne Prop.)“* zu einer schlechten Genauigkeit mit einem Maximum von $P_5 = 0,032$, aber einer akzeptablen Vollständigkeit mit $R_{10} \approx 0,8$ führt. *„TF-IDF (volle Prop.)“* erreicht die gleiche Vollständigkeit bei $k = 6$ und erzielt $R_{10} \approx 0,9$, da häufig „Wurzelentitäten“ in diesem Anwendungsfall relevant sind. *„TF-IDF (partielle Prop.)“* erzielt eine leicht verbesserte Genauigkeit.

Der Grund für die geringe Genauigkeit der *TF-IDF*-basierten Ansätze liegt in der Tatsache, dass zusammenhängende Wortgruppen oft ein „zu geringes“ aggregiertes Gewicht erhalten, wenn sie aus häufig im Referenzdatensatz auftretenden Wörtern bestehen. Dies führt dazu, dass relevante Entitäten zwar erkannt, aber im Vergleich zu Entitäten, die Wörter enthalten, die selten im Referenzdatensatz auftreten, benachteiligt werden. Häufig entsteht aus diesem Grund eine Rangfolge in der Resultatliste, die sich stark von der des Goldstandards unterscheidet.

Dokumente sind in diesem Anwendungsfall im Vergleich zu Dokumenten der „Kundenanfragen“ eher kurz (im Durchschnitt kürzer als 4 Sätze) und damit vergleichbar zu extrem langen Stichwortanfragen. Weiterhin sind die Pfade zwischen Wurzel- und Blattknoten in SAPTerm im Vergleich zur IMDB eher lang. Ein Test, ob in diesem Anwendungsfall kompaktere Antwortgraphen zu bevorzugen sind, erscheint daher sinnvoll. Dementsprechend wurde *CentEC* als alternative Aggregationsstrategie für diesen Anwendungsfall implementiert. Wir folgten dabei dem Gewichtsaggregationsverfahren, das in BANKS [BHN+02] vorgeschlagen wurde.

CentEC interpretiert jede Entität im Graphen als potenziellen Mittelpunkt eines Resultats und summiert die Konfidenzgewichte von Attributübereinstimmungen der direkt benachbarten Entitäten auf. Statt durch *PageRank* (bzw. *ObjectRank*) bestimmte Gewichte [BHN+02] werden in *CentEC* die in Kap. 5.1 diskutierten Konfidenzgewichte aggregiert. Als Schwächungsfaktor stellte sich $p_{\text{self}} = 0,3$ für dieses Verfahren als am besten geeignet heraus. Die Resultate dieses Vergleichsverfahrens sind als *„CentEC (partielle Prop.)“* in Abb. 5.7 dargestellt.

„CentEC (partielle Prop.)“ zeigt sehr gute Ergebnisse für $k = 1$, verliert aber mit wachsenden k an Vollständigkeit und Genauigkeit gegenüber dem in der vorliegenden Arbeit diskutierten Verfahren *„RapidUIM (partielle Prop.)“*. Im Detail betragen die Unterschiede in der Genauigkeit $\Delta P_1 \approx 0,10$ und $\Delta P_5 \approx 0,20$ und in der Vollständigkeit $\Delta R_5 \approx 0,29$ und $\Delta R_{10} \approx 0,37$. Beim Vergleich von *„RapidUIM (partielle Prop.)“* und *„RapidUIM (volle Prop.)“*, das wie bereits

erwähnt mit $p_{\text{self}} = 0$ konfiguriert wurde, zeigt sich in diesem Anwendungsfall, dass die Wahl eines geeigneten p_{self} die Vollständigkeit und Genauigkeit um mindestens 15 Prozentpunkte für alle Rankpositionen verbessert.

5.2.3 Einfluss ausgewählter Parameter

Das Problem der Laufzeitverbesserung durch eine Einschränkung auf die relevantesten Zwischenergebnisse wurde in der Literatur ausführlich diskutiert (siehe z. B. [CNS06]). Im Allgemeinen gilt für den in dieser Arbeit beschriebenen Ansatz, dass der Parameter k so groß gewählt werden sollte, dass er die eigentliche Bewertung nur unwesentlich beeinflusst. Weiterhin stellte sich eine Parametrisierung mit $p_{\text{out}} = 0,2$ in allen Anwendungsfällen als geeignet heraus.

Die Experimente in Kap. 5.2.2 legten nahe, dass der Parameter p_{self} einen großen Einfluss auf die Qualität der Resultate hat. Weiterhin konnte besonders im Anwendungsfall „Kundenbetreuung“ ein zusätzlicher Einfluss des Parameters s, der den Ähnlichkeitsschwellwert für berücksichtigte Übereinstimmungen von Text- und Attribut-N-Grammen festlegt, beobachtet werden. Im Folgenden werden der Einfluss von p_{self} im Anwendungsfall „Forensuche“ und „Kundenbetreuung“ dargestellt und verschiedene Konfigurationen des Parameters s im Anwendungsfall „Kundenbetreuung“ analysiert.

Einfluss des Abschwächungsfaktors

In den Experimenten zeigte sich, dass der Wert für den Abschwächungsfaktor durch $p_{\text{self}} = 1/(\text{durchschnittliche Pfadlänge im Referenzdatensatz})$ für das in Kap. 5.1 vorgestellte Verfahren relativ gut eingestellt werden kann (z. B. für SAPTerm $p_{\text{self}} \approx 0,2$). Die verschiedenen Werte für p_{self} im Falle der Experimente zu den *TF-IDF*-basierten Ansätzen lassen sich am ehesten damit erklären, dass „rohe“ *TF-IDF*-Gewichte verwendet werden, die z. B. nicht anhand des Maximalwertes normiert werden.

Es ist aber klar, dass die optimale Einstellung von p_{self}, auch für den hier entwickelten Ansatz, immer vom konkreten Anwendungsfall abhängt, da die Distanz über die Konfidenzgewichte im Graph propagiert werden müssen, um ein sehr gutes *Ranking* für Entitäten zu erhalten, von Anwendungsfall zu Anwendungsfall variieren kann. So konnte z. B. festgestellt werden, dass sich im Anwendungsfall „Forensuche“ die relevanten Entitäten in geringerer Distanz zu den Treffern befinden als im Anwendungsfall „Kundenbetreuung“.

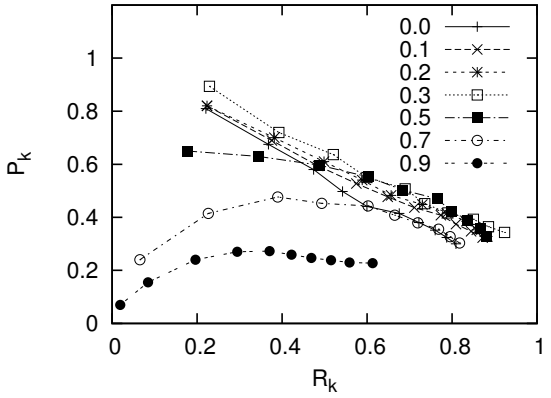


Abb. 5.8: Vollständigkeits-Genauigkeits-Graph für variierende p_{self} im Anwendungsfall „Forensuche“

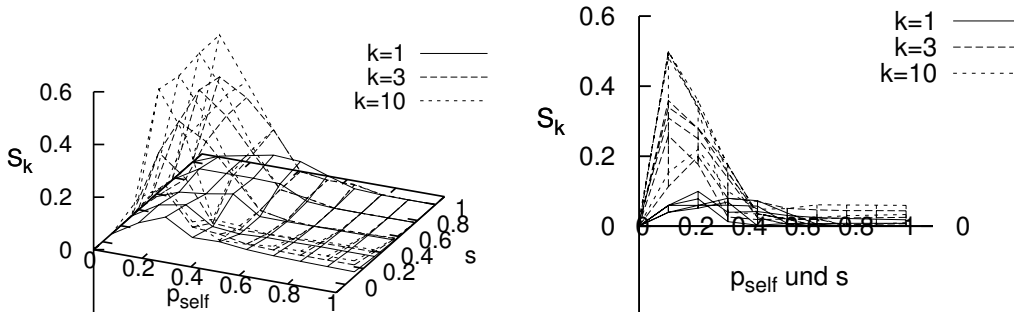
Ein Abschwächungsfaktor von $p_{\text{self}} = 0,3$ lieferte die besten Ergebnisse für den Korpus „Forensuche“, wogegen im Anwendungsfall „Kundenbetreuung“ die Konfiguration $p_{\text{self}} = 0,1$ bes-

sere Ergebnisse lieferte. In Abb. 5.8 ist die Verteilung von Vollständigkeits- und Genauigkeitswerten für den Anwendungsfall „Forensuche“ dargestellt. Die Konfiguration mit $p_{\text{self}} = 0,3$ und dem Ergebnis $P_1 = 0,9$ liefert hier minimal bessere Ergebnisse als $p_{\text{self}} = 0,2$ mit dem Ergebnis $P_1 = 0,83$.

In Abb. 5.9 ist der Einfluss von p_{self} und s auf S_k für verschiedene Rankpositionen ($k = \{1,3,10\}$) im Anwendungsfall „Kundenbetreuung“ dargestellt. Tatsächlich kann für $k = 1$ mit $p_{\text{self}} = 0,2$ ein besseres *Ranking* (in Kombination mit $s = 0,1$) beobachtet werden als mit anderen Konfigurationen. Wenn aber $k = 3$ oder $k = 10$ betrachtet werden, stellt sich $p_{\text{self}} = 0,1$ als bessere Konfiguration heraus.

Einfluss des Ähnlichkeitsschwellwertes

Der Einfluss des Ähnlichkeitsschwellwerts s hat im Anwendungsfall „Kundenbetreuung“ einen starken Einfluss (siehe Abb. 5.9). In den anderen Anwendungsfällen war dessen Einfluss geringer, zeigte aber eine ähnliche Charakteristik: Ein niedriger Wert für s verbessert die Resultate für niedrigere Rankpositionen (z. B. $k = 1$), wogegen er in der Regel einen äußerst geringen Einfluss auf die Ergebnisse für höhere Positionen wie $k = 10$ hat. Es entsteht daher ein Zielkonflikt zwischen guten Ergebnissen am Beginn der Ergebnisliste und einer geringen Verarbeitungszeit durch Reduktion der zu berücksichtigenden approximativen Treffer während der Aggregation von Konfidenzgewichten.



(a) Einfluss von p_{self} und s auf S_k (b) Profil von Abb. 5.9(a)

Abb. 5.9: Einfluss von p_{self} und s für den Anwendungsfall „Kundenbetreuung“

Im konkreten Anwendungsfall führt ein Wert von $s = 0,1$ zu $S_1 \approx 0,18$ und $s = 0,7$ zu $S_1 \approx 0,06$. Die Resultate für S_{10} unterscheiden sich nur unwesentlich. Der Wert $s = 0,7$ führt jedoch im Durchschnitt zu einer Reduktion von 5.000 zu 1.000 Knoten in den Dokument-Konzept-Graphen, also zu einer enormen Beschleunigung der Verarbeitungszeit. Im Anwendungsfall „Kundenbetreuung“ ist ein gutes *Ranking* an Position $k = 1$ wichtiger als die Verarbeitungszeit, so dass eine Konfiguration mit $s = 0,1$ sinnvoll ist.

5.2.4 Zusammenfassung

Die Experimente mit dem Referenzdatensatz IMDB und dem Korpus „Filmbewertungen“ zeigten, dass ein „naiver“ *TF-IDF*-basierter Ansatz, der in Beziehung stehende Entitäten bei der Identifikation ignoriert, in diesem Szenario wenig geeignet ist. Auch eine Lösung, die auf ein popularitätsbasiertes *Rankingverfahren* einer Websuchmaschine zurückgreift und alle Attribute im Kontext einer Entität ausnutzt, ist in diesem Anwendungsfall wenig effektiv. Das hier vorgestellte Verfahren liefert weiterhin wesentlich bessere Ergebnisse als ein *TF-IDF*-basierter Ansatz, der Attributübereinstimmungen aller im Referenzdatensatz definierten Entitäten berücksichtigt.

Die Ergebnisse der Experimente mit dem Dokumentenkorpus „Kundenbetreuung“ und dem Referenzdatensatz SAPTerm zeigten weiterhin, dass das hier vorgestellte Verfahren auch einem maschinellen Lernansatz, der mit 7.000 Dokumenten für 10% der relevanten Entitäten trainiert wurde, überlegen ist. Auch in diesem Anwendungsfall war unser Verfahren den *TF-IDF*-basierten Ansätzen überlegen. In SAPTerm liegt eine ausgeprägte Art einer Produkt-Teil-Hierarchie vor. Daher wurden zusätzlich Experimente mit einem *TF-IDF*-basierten Ansatz durchgeführt, der einen Abschwächungsfaktor bei der Aggregation von Gewichten berücksichtigt. Die Resultate dieses Ansatzes waren vergleichbar zu denen des maschinellen Lernansatzes.

Im Anwendungsfall „Forensuche“ wurden die Ergebnisse bezüglich der oben genannten *TF-IDF*-basierten Ansätze gesichert. Unser Ansatz lieferte auch hier signifikant bessere Ergebnisse. Auch eine alternative Gewichtsaggregation, die in diesem Anwendungsfall erfolgversprechend war, und die Konfidenzgewichte von Attributübereinstimmungen der direkt benachbarten Entitäten berücksichtigte, wurde getestet. Es zeigte sich, dass ein solcher Ansatz gute Ergebnisse für die erste Rankposition liefert. Die Genauigkeit und Vollständigkeit für höhere Rankpositionen ist jedoch wesentlich schlechter, als mit der Aggregationsstrategie, die in Kap. 5.1 vorgeschlagen wurde.

Eine Untersuchung zu dem Einfluss des Ähnlichkeitsschwellwertes zeigte, dass ein niedriger Schwellwert bessere Resultate am Beginn der Ergebnisliste liefert, die Resultate an höheren Rankpositionen jedoch nur unwesentlich beeinflusst. Weiterhin konnte gezeigt werden, dass eine Konfiguration des Abschwächungsfaktors für die Gewichtspropagation relativ gut auf Basis des Reziproken der durchschnittlichen Pfadlängen zwischen Blatt- und Wurzelknoten im Referenzdatensatz bestimmt werden kann, wenn kein Testkorpus vorliegt.

5.3 Verwandte Arbeiten in der Identifikation von Entitäten

In Kap. 2.3.2 wurden bereits verwandte Arbeiten überblicksweise abgegrenzt. Im Folgenden diskutieren wir ausgewählte Arbeiten detailliert und gehen noch einmal auf die in Kap. 3.3 verwendeten Vergleichsverfahren ein.

Wir merken zusätzlich an, dass die in Kap. 5.1 vorgestellte Methode Ideen aus dem Bereich der Graph-basierten Duplikaterkennung in strukturierten Daten [EIV07,LCW07,CKM07] adaptiert. Einige Arbeiten in diesem Gebiet betrachten semi-strukturierte Daten wie XML [PWN06]. Verfahren, in denen eine Datenquelle unstrukturierter Natur und die andere Graph-strukturiert ist, wurden im Bereich der Informationsintegration jedoch bisher nicht untersucht.

In Kap. 5.3.1 grenzen wir Ansätze aus dem Bereich der Sinndisambiguierung mit Referenzdaten ab. Wir unterscheiden dabei in Verfahren zur Wortsinndisambiguierung und Methoden zur Disambiguierung von Entitäten (Personen, Ortsbezeichnungen etc.), wobei letztere Ansätze dem hier vorgestellten Verfahren näher verwandt sind.

Die zu unserem Ansatz am engsten verwandten Arbeiten sind im Gebiet der Identifikation von Entitäten mit Graph-strukturierten Referenzdatensätzen angesiedelt und werden in Kap. 5.3.2 betrachtet. Zu Beginn gehen wir auf Ansätze ein, die Ontologien als eine Art Wörterbuch verwenden und vordefinierte Beziehungen der Entitäten zum Zeitpunkt der Dokumentsuche auswerten. Abschließend diskutieren wir Ansätze, die implizit in Texten referenzierte Entitäten identifizieren.

5.3.1 Referenzdaten-basierte Sinndisambiguierung

Im Fokus der Wortsinndisambiguierung steht zumeist die Selektion des korrekten „*Synonymrings*“ (engl.: „*Synset*“), die in einem Referenzdatensatz vorgegeben sind, für ein Wort eines Textes entsprechend seiner Semantik. Wir betrachten Arbeiten aus diesem Bereich zu Beginn von Kap. 5.3.1. Ein zu der hier vorgestellten Methode näher verwandter Forschungsbereich

ist die Disambiguierung von Eigennamen, in dem nicht die Semantik von einzelnen Wörtern, sondern die korrekte Zuordnung von Textsegmenten und Entitäten im Sinne dieser Arbeit untersucht wird. Wir diskutieren die in diesem Bereich entwickelten Verfahren im Anschluss.

Wortsinndisambiguierung mit Graph-strukturierten Referenzdatensätzen

Zur Wortsinndisambiguierung werden in der Literatur drei Klassen von Ansätzen beschrieben (siehe [IV98, Nav09] für einem Überblick und [AAD+09] für einen Vergleich): die Referenzdaten-basierte Disambiguierung, die Disambiguierung mit Hilfe von statistischen Modellen und *Clusterverfahren*, die alleine auf Korpusstatistiken basieren. Wir beschränken uns im Folgenden auf Ansätze, die wie unser Verfahren Referenzdaten verwenden.

Der populärste zur Wortsinndisambiguierung eingesetzte Graph-strukturierte Referenzdatensatz ist *WordNet* [Mil95]. *WordNet* umfasst ca. 128.000 Begriffe[16] und besitzt damit eine relativ geringe Mächtigkeit im Vergleich zu Referenzdaten im Unternehmensumfeld, wie z. B. SAP-Term (ca. 155.000 Konzepte), beschreibt aber eine wesentlich größere Domäne (die Englische Sprache). Der Grund dafür ist, dass Eigennamen wie auch die Kombination von Wörtern zu neuen Bedeutungen für abgegrenzte Domänen in *WordNet* nicht berücksichtigt werden.

[Voo93] gehört zu einer der frühesten Arbeiten, die den Referenzdatensatz *WordNet* ausnutzten, und schlägt zur Disambiguierung die Einteilung von *WordNet* in nicht mehrdeutigen Regionen – in denen ein Wort und dessen Synonyme eindeutig sind – vor. Grundlage der Disambiguierung sind Korpusstatistiken. In einer Vorverarbeitungsphase wird die durchschnittliche Anzahl von Übereinstimmungen von Text und Wörtern für jede Region in einem kompletten Scan des Korpus erfasst. In der Disambiguierungsphase wird der Wortsinn selektiert, der in der Region mit der höchsten relativen Trefferanzahl im Vergleich zur durchschnittlichen Trefferanzahl auftritt.

In der Literatur wurde eine Vielzahl von Graph-basierten Ansätzen, die auf Korpusstatistiken verzichten, diskutiert [SP95, RS95, AR96, MTT+98, SOT03, HCK+04, ALS09, NL10]. Sie basieren zumeist auf der Berechnung von minimalen Spannbäumen des aktuellen untersuchten Wortes und der Kontextwörter. Es wird der Wortsinn selektiert, dessen Spannbaum die minimale aggregierte Distanz zwischen den beinhalteten Knoten aufweist. Einfache Verfahren verwenden als Distanzmetrik die Anzahl von Kanten. Neuere Verfahren adaptieren z. B. *PageRank*, *HITS* oder *InDegree* zur Gewichtung von Kanten (siehe z. B. [NL10] für einen Vergleich).

Hinsichtlich der Einschränkung der Disambiguierung auf den Wortkontext werden so genannte *Lexical Chaining*-Verfahren [GM03] diskutiert, die potenziell verwandte Wortsinne im Kontext des Gesamtdokuments einander zuordnen, und eine gemeinsame Disambiguierung ermöglichen. Selbst, wenn als Kontext der gesamte Dokumentext herangezogen wird, sind die im Bereich der Wortsinndisambiguierung vorgeschlagenen Verfahren nicht ohne weiteres geeignet, um implizite Referenzen hinsichtlich relevanter Entitäten zu bewerten.

Weiterhin sind Ansätze, die approximative Abgleichverfahren im Umfeld der Wortsinndisambiguierung untersuchen, nicht bekannt. Die Methode, die in Kap. 5.1 vorgeschlagen wurde, unterstützt auch die Zuweisung von einem „Wortsinn“ für approximative Übereinstimmungen. Dies ist insbesondere interessant, wenn ein Wort mit dem Referenzdatensatz exakt übereinstimmt, jedoch der Dokumentkontext eine Bedeutung hinsichtlich eines lediglich approximativ übereinstimmenden Konzepts nahelegt, wie z. B. anhand der Attributübereinstimmungen zu „*exchange*“ und „*Exchange Infrastructure*“ in Abb. 5.1 auf S. 116 diskutiert wurde.

Disambiguierung von Eigennamen

Ansätze, die sich mit der Disambiguierung von Eigennamen befassen, sind mit den oben diskutierten Arbeiten nahe verwandt. Das Ziel der Disambiguierung von Eigennamen ist es, die

[16] `http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html`, Stand 04.04.2010

Bedeutung von nicht eindeutig identifizierten Entitäten, wie Personen oder Organisationen, mit Hilfe des textuellen Kontexts zu bestimmen. Wiederum lassen sich die Ansätze in maschinelle Lernverfahren und Verfahren, die Referenzdaten verwenden, unterscheiden. Ansätze, die so genannte „Profile" für Entitäten erlernen, also die Merkmalsverteilung im Kontext von Entitäten statistisch beschreiben, sind z. B. [DEG+03, BP06, Cuc07, KNTM08, SLH09].

Im Folgenden stehen Ansätze, die Graph-strukturierte Referenzdaten zur Disambiguierung von Entitäten verwenden, im Mittelpunkt. In diesem Bereich sind neben Verfahren, die Graph-strukturierte Referenzdaten wie Wikipedia[17], DBLP[18] oder Verbindungen in Sozialen Netzwerken ausnutzen, auch Ansätze aus dem Bereich des geographischen *Information Retrieval* einzuordnen, die z. B. geographische Taxonomien zur Disambiguierung von Ortsnamen verwenden.

[BM05] schlägt die Verwendung von Linkstrukturen zwischen Personen in Sozialen Netzwerken vor, um *Cluster* von miteinander verbundenen Personen abzuleiten, welche wiederum zur Disambiguierung genutzt werden können. In [MCN06] wird die Disambiguierung von Personen mit Hilfe von Graphen beschrieben, die aus dem Emailverkehr konstruiert werden. Es werden anwendungsspezifische Relevanzmaße bezüglich dem Inhalt von Emails und der zeitlichen Entwicklung des persönlichen Netzwerks betrachtet.

[HAMA06] diskutiert einen ebenfalls anwendungsspezifischen Ansatz zur Disambiguierung von Personennamen in Konferenzankündigungen auf Basis des DBLP-Referenzdatensatzes. Es wird im Speziellen betrachtet, in welchem Kontext die in Beziehung stehenden Entitäten zur aktuell zu disambiguierenden Entität zu erwarten sind (z. B. Organisationen im rechten Kontext des Personennamens). Die Bewertung erfolgt auf Basis der Anzahl von im Kontext einer Entität gefundenen, in Beziehung stehenden Entitäten. Kontexteinschränkungen können mit *RapidUIM* einfach modelliert werden und damit eine generische Lösung für derartige Anwendungsklassen geschaffen werden.

[Cuc07] und [HZ09] stellen Ansätze vor, die es erlauben, die Struktur von Wikipedia zur Disambiguierung von Entitäten heranzuziehen, wobei jedoch lediglich unmittelbar in Beziehung stehenden Entitäten zur Disambiguierung verwendet werden. Es handelt sich eher um einfache „Profile" von Entitäten und nicht um einen Ansatz, der die Graph-struktur des Referenzdatensatzes im Sinne dieser Arbeit ausnutzt.

Im Forschungsgebiet des geographischen *Information Retrieval* [Sue09] wird zumeist die geographische Nähe zwischen Orten, Regionen etc. genutzt, um eine Disambiguierung vorzunehmen. [AHSS04] verwendet eine geographische Taxonomie, um so genannte „Fokusregionen" eines Textes zu bewerten. Als Konfidenzgewichte werden vorkonfigurierte Gewichte für bestimmte Situationen verwendet (z. B. ein Gewicht von 0,5 für mehrdeutige Treffer). Die Konfidenzgewichte werden im Taxonomiegraph propagiert. Ortsnamen sind jedoch meist kurz (einzelne Wörter) und können ohne weiteres zusammenhängend im Text erkannt werden. Die Problematik von linguistischen Heterogenitäten und Mehrdeutigkeiten ist wesentlich geringer als in den hier betrachteten Anwendungsfällen.

5.3.2 Identifikation von Entitäten mit Graph-strukturierten Referenzdaten

Im Folgenden diskutieren wir Arbeiten aus dem Bereich der Ontology-basierten Dokumentsuche und gehen abschließend auf Ansätze zur Identifikation von implizit in Dokumenten referenzierten Entitäten ein.

[17] `http://wikipedia.org`, Stand: 01.10.2010
[18] `http://dblp.uni-trier.de`, Stand: 01.10.2010

Ontologie-basierte Dokumentsuche

Für die Dokumentsuche in digitalen Bibliotheken, dem WWW, aber auch für die *Enterprise Search* wurden eine Reihe von Ansätzen entwickelt, die Dokumente mit semantischen Informationen anreichern. Das Ziel dieser Anreicherung ist es zumeist, Dokumente mit zusätzliche Metadaten zu annotieren, um z. B. ein Filtern von Suchergebnissen mit Hilfe von semantischen Facetten bzw. eine Navigation in Suchergebnissen zu ermöglichen oder die Vollständigkeit von Suchergebnissen zu verbessern. Wir gehen im Folgenden auf ausgewählte Ansätze ein, die Entitäten, die in Ontologien vordefiniert sind, für die Dokumentsuche ausnutzen.

Für das u38-System [MSM06] wurde das *Verlinken* von aus Unternehmensontologien bekannten Entitäten mit Dokumenten vorgeschlagen. Die Ontologie dient als einfaches Wörterbuch. Eine Bewertung von Übereinstimmungen oder die Identifikation von implizit referenzierten Entitäten wird nicht diskutiert. Zur Bewertung von Dokumenten für Stichwortanfragen wird eine Adaption von *PageRank* auf Basis der Ontologie und der entstehenden Verweise zwischen Ontologie und Dokumentenkorpus vorgeschlagen. [VFC05] schlägt für eine ähnliche Problemstellung *TF-IDF* zur Bewertung von semantischen Annotationen auf Basis von Statistiken des Gesamtkorpus vor.

[NV03] diskutiert eine *Query Expansion* unter Ausnutzung von *WordNet*. Nach einer Disambiguierungsphase für Anfragewörter werden die zu den disambiguierten Wortsinnen in Beziehung stehenden Konzepte in *WordNet* betrachtet. Sich überschneidende Konzepte im Netzwerk der verschiedenen Wortsinne werden zur Erweiterung der Anfrage ausgewählt. Es handelt sich bei diesem Ansatz um ein einfaches Verfahren zur Identifikation von implizit referenzierten Entitäten. Es werden weder im Dokument verteilte noch approximative Übereinstimmungen berücksichtigt.

Im Bereich der biomedizinischen Dokumentsuche wurden viele Ansätze entwickelt, die es erlauben, auf Dokumenten und deren Inhalten mit Hilfe von semantischen Annotationen zu navigieren. *GOPubMed* [DAA+08], *TextPresso* [MKS04] und *GOAnnotator* [CSL+06] speichern ähnlich wie [MSM06] Verweise von Dokumenten zu Konzepten einer Ontologie, um eine Klassifikation von Dokumenten und Navigation in Suchresultaten zu ermöglichen. *AliBaba* [PNLH09] nutzt verschiedene Ontologien zur Identifikation von Entitäten und extrahiert unbekannte Beziehungen zwischen Entitäten. Es ermöglicht eine Navigation in aus Dokumenten extrahierten Fakten. Auf relevante Ausschnitte der Dokumente kann von den extrahierten Fakten aus zugegriffen werden.

Die meisten Systeme im Bereich der biomedizinischen Dokumentsuche erlauben Anfragen nach einem gemeinsamen Auftreten von Entitäten in Textstrukturen (z. B. Sätzen oder Einleitungen von wissenschaftlichen Artikeln) und eine Navigation im Dokumentenkorpus anhand der Struktur einer gegebenen Ontologie und/ oder extrahierter Beziehungen. Das Problem von uneindeutigen, im Text verteilten Übereinstimmungen, stellt sich z. B. bei der Identifikation von Entitäten des Typs „biologischer Prozess“. Wir diskutieren derartige Ansätze im Folgenden.

Indirekte Identifikation von Entitäten in Dokumenten

Einige wenige Ansätze diskutieren das Problem der Identifikation von Entitäten, die in Texten nur implizit referenziert werden. Im Kontext dieser Problemstellung werden Statistische Maschinelle Lernverfahren und Referenzdatensätze verwendet.

Statistische Maschinelle Lernverfahren trainieren, ähnlich den Lernverfahren zur Disambiguierung, „Profile“ für Entitäten, die Wörter, die im Kontext einer Entität von Bedeutung sind, als statistisches Modell abbilden. Am bekanntesten in diesem Gebiet sind Ansätze, die im Kontext des *Enterprise Search Track* der *Text REtrieval Conference (TREC)* entwickelt wurden (siehe z. B. [CVS05, SVC06, BVCS07]). Statistische Modelle werden mit Dokumenten trainiert, die je-

weils mit einer Entität annotiert sind. Wir verglichen unser Verfahren zu solch einem Ansatz, der an [BAR06, FZ07] angelehnt ist, in Kap. 5.2.

In [GYLRS08] wird die Identifikation von Entitäten z. B. vom Typ „*biologischer Prozess*“ mit Bezeichnungen der Form „*activation of JNKK activity*“ in wissenschaftlichen Abhandlungen auf Basis des Graph-strukturierten Referenzdatensatzes *Gene Ontology (GO)* [Con04] beschrieben. Basis der in [GYLRS08] vorgeschlagenen Identifikationsmethode sind *TF-IDF*-Wortgewichte und ein Gewicht, welches die Nähe der identifizierten Bestandteile eines Bezeichners im Kontext von Textsegmenten (Substantivgruppe, Satz bzw. Absatz) bewertet. Textsegmente definieren in [GYLRS08] die Grenzen in denen einzelne Entitäten bewertet werden. Verschiedene Attribute oder Beziehungen zwischen Entitäten werden nicht berücksichtigt.

Ansätze, die verschiedene Attribute einer Entität zur Identifikation berücksichtigen, sind [CSC05] und [MK07]. [CSC05] unterstützt die Verwendung von alternativen Namen für Entitäten, die in GO definiert sind und aggregiert inverse Wortfrequenzen zur Bewertung von Entitäten. [MK07] untersucht die Identifikation von Entitäten mit Attributen verschiedenen Typs (Hersteller, Produktname etc.). Texte (hier Titel von Foreneinträgen) werden als Wortvektor dargestellt und eine Übereinstimmung zwischen Text- und Referenzdatentoken berücksichtigt, wenn deren Jaro-Winkler-Distanz größer als ein Schwellwert ist. Die Jaro-Winkler-Distanz wird bei der Bewertung von Entitäten vernachlässigt. Aggregierte Gewichte werden stattdessen auf Basis von *TF-IDF* oder dem Jaccard-Koeffizient bestimmt, wobei festgestellt wird, dass die *TF-IDF*-Metrik bei Vernachlässigung von approximativen Übereinstimmungen die besten Ergebnisse liefert. Der Ansatz von [MK07] wurde dementsprechend zur Evaluation in Kap. 5.2 verwendet.

Der Ansatz, der unserem Verfahren am meisten ähnelt, ist EROCS [CGRM06] (Vergleichsverfahren in Kap. 5.2). Die *TF-IDF*-Wortgewichte für Treffer innerhalb des Teilbaums einer „Wurzelentität“ werden aufsummiert ohne weitere Strukturmerkmale, wie z. B. den Abstand zwischen Treffern und Entitäten in der Baumstruktur zu berücksichtigen. Diese Art der Aggregation von Gewichten für „Wurzelentitäten“ ist für Anwendungsfälle, in denen z. B. Produkt-Teil-Hierarchien vorliegen, nicht geeignet. Die Mehrdeutigkeit von Attributinhalten bzw. ihrer Bestandteile oder approximative Abgleichverfahren werden weiterhin in [CGRM06] nicht diskutiert.

Der Ansatz von [CGRM06] fokussiert vielmehr die effiziente Abbildung von Textfragmenten auf „Wurzelentitäten“ mit Hilfe von einem neuartigen *Caching*-Verfahren für Anwendungsfälle, in denen die Referenzdaten nicht im Hauptspeicher gehalten werden können. Einzelne Textsegmente (hier Substantivgruppen) werden im ursprünglichen Anwendungsfall von [CGRM06] zu grob-granularen, zusammenhängenden Textfragmenten (z. B. Teile von Absätzen) zusammengefasst, die derselben „Wurzelentität“ zugeordnet werden können. Das Ziel von [CGRM06] ist daher eher eine äußerst effiziente Segmentation eines Textes in Bezug auf „Wurzelentitäten“ einer Baumstruktur und nicht die effektive Relevanzbewertung von Entitäten.

5.4 Zusammenfassung

In Kap. 5 wurde ein Ansatz zur Identifikation von Entitäten in Texten auf Basis von Graph-strukturierten Referenzdatensätzen beschrieben, der als Extraktionsmodul in das in Kap. 3.1 beschriebene *RapidUIM*-System eingebettet ist. Der Ansatz zeichnet sich dadurch aus, dass er insbesondere für Referenzdatensätze geeignet ist, in denen Bezeichner von Entitäten eher Umschreibungen gleichen. Er berücksichtigt weiterhin verschiedene Attribute zur Identifikation von einzelnen Entitäten sowie deren vordefinierte Beziehungen.

Zur Beschleunigung des Abgleichs zwischen Text und Referenzdaten werden Attribute indexiert. Es handelt sich dabei um einen Token-N-Gramm-Index (mit variabler Tokenanzahl) statt eines einfachen Tokenindex, wodurch zum einen extrem mehrdeutige Token-N-Gramme entfernt und zum anderen eine Relevanzbewertung von Tokensequenzen unterstützt wird. Die

Bewertung von Attributübereinstimmungen innerhalb eines Textsegments, das so genannte Konfidenzgewicht, wird auf Basis von drei Metriken berechnet: der Ähnlichkeit der Text- und Attribut-N-Gramme, der Relevanz im Textsegment (z. B. bei Überlappungen) und dem Referenzdatensatz (Ein- bzw. Mehrdeutigkeit) sowie der Abdeckung der Attribut-N-Gramme bezüglich der ursprünglichen Attributinhalte.

Textsegmente, übereinstimmende Attribute und zugehörige Entitäten werden in einen *Dokument-Konzept-Graphen (DKG)* abgebildet. Zusätzlich berücksichtigen wir weitere in Beziehung stehende Entitäten des Referenzdatensatzes im DKG, so dass auch implizit referenzierte Entitäten identifiziert werden können. Der DKG beschreibt somit alle Konzepte des Textes und des Referenzdatensatzes, die potenziell relevant für ein Dokument sind. Ein *Spreading Activation*-Alogrithmus propagiert Konfidenzgewichte im DKG, wobei ein Abschwächungsfaktor die Propagation limitiert. Die am höchsten bewerteten Entitäten subsumieren die Teilgraphen des Referenzdatensatzes, die als für das Dokument relevant selektiert werden und anschließend durch Anwendungen konsumiert werden können.

Die Untersuchung von verwandten Arbeiten ergab, dass bisherige Ansätze lediglich Teilprobleme der hier betrachteten Anwendungsfälle adressieren. Ein Ansatz, der alle oben genannten Merkmale vereinigt, wurde in der Literatur bisher nicht beschrieben. Die Evaluation zeigte die praktische Anwendbarkeit des entwickelten Verfahrens. Insbesondere stellte sich heraus, dass die vorgestellten Bewertungsmethoden ein wesentlich besseres *Ranking* von Entitäten im Vergleich zu weit verbreiteten auf *TF-IDF*-Wortgewichten aufbauenden Ansätzen erzeugt. Auch konnte gezeigt werden, dass die vorgestellte Methode bessere Ergebnisse als ein maschinelles Lernverfahren und eine Websuchmaschine liefert, die für diese Problemstellungen adaptiert wurden.

Eine Reihe von Parametern beeinflussen die Bewertung, wobei in der Evaluation insbesondere ein starker Einfluss des Abschwächungsfaktors und des Ähnlichkeitsschwellwertes nachgewiesen wurde. Hinsichtlich des Ähnlichkeitsschwellwertes konnte gezeigt werden, dass ein niedrigerer Schwellwert bessere Resultate am Beginn der Ergebnisliste liefert, aber die Qualität der Resultate an höherer *Rankposition* nicht beeinflusst. Es handelt sich daher um ein klassisches Ausgleichsproblem zwischen guten Ergebnissen zu Beginn der Resultatliste und geringen Verarbeitungszeiten.

In den hier betrachteten Anwendungsfällen konnte weiterhin dargestellt werden, dass für den Abschwächungsfaktor eine geeignete Konfiguration auf Basis des Reziproken der durchschnittlichen Pfadlängen zwischen Blatt- und Wurzelknoten im Referenzdatensatz bestimmt werden kann. Inwiefern diese Heuristik auf andere Anwendungsfälle übertragbar ist und ob z. B. eine optimale Konfiguration des Abschwächungsfaktors für individuelle Beziehungen auf Basis eines nicht annotierten Dokumentenkorpus erlernt werden kann, bleiben als zukünftige Forschungsprobleme bestehen.

Als zukünftige Problemstellung ist weiterhin eine zur parallelen Datenverarbeitung orthogonale Laufzeitoptimierung denkbar. Insbesondere ist eine Kompression des Referenzdatensatzgraphen entlang möglicher Pfade vorstellbar, in dem zum Teil vorberechnete Pfadgewichte eingebettet werden können. Da nicht alle Entitäten, die in einem Dokument-Konzept-Graph abgebildet werden, für die Weiterverwendung durch eine Anwendung relevant sind, ist grundsätzlich eine Bewertung auf Basis von Text-Referenzdaten-Übereinstimmungen möglich, ohne dass eine De-kompression vorgenommen werden muss. Relevante Teilgraphen müssen erst zum Zeitpunkt der Weiterverwendung durch eine Anwendung wiederhergestellt werden, so dass von einer wesentlichen Laufzeitverbesserung durch einen derartigen Ansatz auszugehen ist.

Es ist des weiteren möglich, Gewichte z. B. für die Ähnlichkeit von Text- und Attribut-N-Grammen, nachdem sie einmal berechnet wurden, zu speichern und wieder zu verwenden, so dass das Problem des approximativen Abgleichs für bekannte Übereinstimmungen zu einem exakten Abgleich reduziert wird.

6 Zusammenfassung und Ausblick

Die vorliegende Arbeit untersuchte Methoden und Verfahren zur Informationsextraktion (IE) für Anwendungen im Kontext der *Enterprise Search*. Zu Beginn dieser Arbeit wurden Technologien aus dem Bereich der Informationsextraktion klassifiziert und deren Relevanz für Anwendungen im Bereich der *Enterprise Search* bewertet. Weiterhin wurde als Grundlage der vorliegenden Arbeit eine Extraktionsplan-basierte Plattform für die effiziente Implementation von IE-Systemen vorgestellt.

Die wesentlichen Resultate der vorliegenden Arbeit sind:

1. ein Verfahren zur Intra-Dokument-Parallelisierung von Extraktionsplan-basierten IE-Systemen, welches die Verarbeitung von einzelnen Dokumenten signifikant beschleunigt,
2. Algorithmen und Bewertungsverfahren zur Inferenz von Regulären Ausdrücken anhand von Beispielentitäten und
3. eine Methode zur Identifikation von Entitäten in Texten, die durch Graph-strukturierte Referenzdatensätze vordefiniert sind.

Im Folgenden fassen wir die Ergebnisse zusammen und geben einen Ausblick auf zukünftige Forschungsprobleme.

6.1 Zusammenfassung der Ergebnisse

Die vorliegende Arbeit wurde dadurch motiviert, dass sich die Extraktion und Identifikation von Entitäten im Kontext der *Enterprise Search* in wesentlichen Punkten z. B. von der Extraktion aus dem WWW unterscheidet. Zum einen müssen verschiedenartigste Typen vom domänenspezifischen Entitäten unterstützt werden, wogegen im WWW häufig Entitäten von allgemeinem Interesse wie Personen oder Organisationen von Bedeutung sind. Zum anderen erfordert die unterschiedliche Struktur von Dokumenten in Unternehmen, dass unterschiedliche Extraktionsverfahren, die häufig auch anwendungsspezifisch sind, entwickelt werden müssen und mit generischen Verfahren zu kombinieren sind.

Eine effiziente Wiederverwendung und Komposition von generischen und anwendungsspezifischen IE-Modulen ist daher gerade im Kontext von Textanwendungen im Unternehmens-

bereich vorteilhaft. Ein weiteres Unterscheidungsmerkmal ist, dass im Unternehmenskontext eine Vielzahl von strukturierten Datenquellen zur Verfügung stehen, die durch bisherige Extraktionsverfahren nicht im vollen Maße ausgenutzt werden können.

Geschäftsrelevante Entitäten lassen sich in zwei Klassen einteilen: Zum einen können viele Entitäten, die einer speziellen Syntax folgen, sehr gut mit einfachen Regulären Ausdrücken beschrieben werden, da sie anhand von (zum Teil unbekannten) Bildungsvorschriften erzeugt werden. Die zweite Klasse von hier betrachteten Entitäten trägt Bezeichner, die eher einen umschreibenden Charakter haben. Sie werden daher in Texten auf andere Art und Weise als in den Referenzdaten definiert erwähnt bzw. nur implizit durch in Beziehung stehende Entitäten referenziert.

Grundlage der in dieser Arbeit vorgestellten Konzepte ist die *RapidUIM*-Plattform, die im Kontext dieser Arbeit entwickelt wurde. Sie ermöglicht die Entwicklung von generischen und anwendungsspezifischen IE-Modulen auf Basis einer gemeinsamen API und stellt ein generisches Graph-basiertes Datenmodell zur Abbildung von Dokumenten, Entitäten und deren Beziehungen zur Verfügung. Insbesondere zeichnet sich *RapidUIM* dadurch aus, dass IE-Module grundsätzlich beliebig miteinander kombiniert werden können, so dass eine Wiederverwendung von einzelnen Modulen im Gegensatz zu vorherigen Ansätzen ohne weiteres möglich ist.

Die Kombination von IE-Modulen zu IE-Systemen geschieht durch deren graphische Komposition zu Extraktionsplänen, wie anhand des zusammenfassenden Beispiels eines Extraktionsplans für das Anwendungsszenario zur Annotation von Foreneinträgen des SAP Community Network (SCN) in Abb. 6.1 dargestellt ist (siehe Kap. 1.2.2). Die Logik dieses Extraktionsplans segmentiert einen Foreneintrag entsprechend seiner Struktur, extrahiert Java-Fehlernachrichten mit Regulären Ausdrücken („REGEX“ in Abb. 6.1.(2)) und identifiziert implizit referenzierte Entitäten (IE-Module in Abb. 6.1.(3)). Anwendungsspezifische Module, wie etwa zur Annotation von Dokumentstrukturen (z. B. „BODY“ oder „TITLE“), werden dazu mit generischen IE-Modulen (z. B. „PARAGRAPH“, „SENTENCE“, „UNION“ oder „POS-FILTER“) kombiniert.

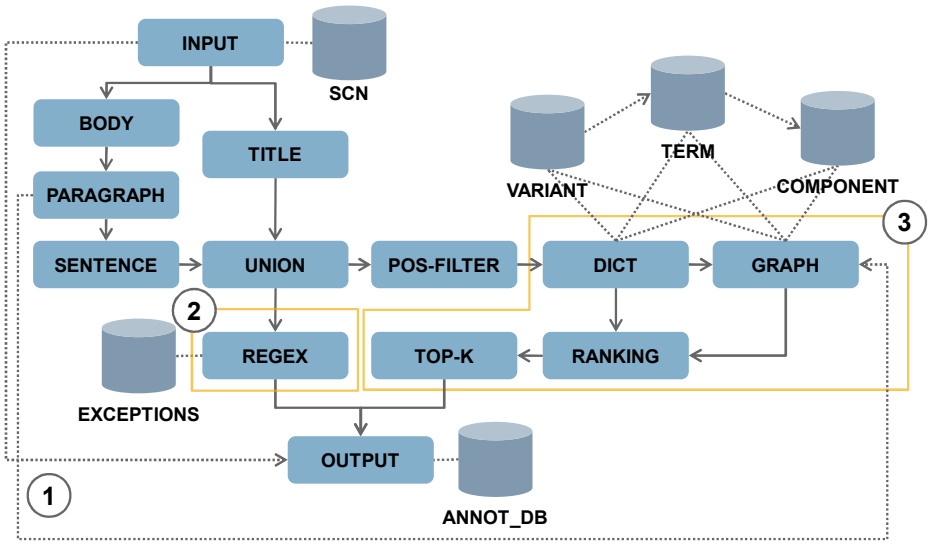


Abb. 6.1: Zusammenfassendes Beispiel für die Annotation von Foreneinträgen

In Abb. 6.1 sind die Beiträge dieser Arbeit im Kontext des SCN-Anwendungsszenarios dargestellt. Wir fassen diese im Folgenden zusammen.

Parallelisierte Datenverarbeitung für Extraktionsplan-basierte IE-Systeme. Für Systeme wie *RapidUIM*, die Abhängigkeiten von Extraktionsaufgaben explizit abbilden, wurde eine parallelisierte, Datenstrom-artige Dokumentverarbeitung in Kap. 3 vorgestellt. Sie entkoppelt einzelne IE-Module, die in Mehrprozessorarchitekturen somit voneinander autonom Daten verarbeiten können.

Die asynchrone, parallele Verarbeitung bei Produzent-Konsument-Abhängigkeiten (direkte Datenabhängigkeiten) wird durch die Verwendung von Warteschlangen realisiert. Eine Datenparallelität besteht jedoch nicht nur hinsichtlich der parallelen Verarbeitung von Daten innerhalb einer Modulsequenz, sondern kann durch die Verwendung von mehreren Instanzen eines IE-Moduls erweitert werden. Mehrere Modulinstanzen verarbeiten in diesem Fall Eingaben derselben Warteschlange.

Indirekte Datenabhängigkeiten, die z. B. bei der Extraktion von Beziehungen zwischen Entitäten innerhalb von Satzgrenzen auftreten, können modelliert und bei der Parallelisierung berücksichtigt werden. In Abb. 6.1 wird z. B. eine indirekte Abhängigkeit für das Modul „GRAPH“ definiert, so dass für dieses Beispiel der Kontext zur Bewertung von Entitäten auf Absätze („PARAGRAPH“) eingegrenzt wird. Ein effizientes Verfahren zur Überprüfung von derartigen Abhängigkeiten ermöglicht hier eine nebenläufige Bewertung von Entitäten, sobald alle notwendigen Eingaben im Kontext eines Absatzes zur Verfügung stehen. Weiterhin wurden in Kap. 3 theoretisch erreichbare Laufzeitverbesserungen im Vergleich zu einer sequentiellen Verarbeitung in herkömmlichen Plattformen analysiert.

Die detaillierte Untersuchung von verwandten Arbeiten ergab, dass ein vergleichbares Verfahren bisher nicht beschrieben wurde. Es bestehen Parallelen zu *MapReduce*-basierten Ansätzen dahingehend, dass in zwei Typen von Operationen, hier der Extraktion aus und Aggregation von Zwischenergebnissen, unterschieden wird. Eine Datenstrom-artige Verarbeitung zur Informationsextraktion wird von *MapReduce*-basierten Ansätzen jedoch bisher nicht unterstützt. Weiterhin betrachteten wir die Beziehung des vorgestellten Verfahrens zu komplementären Optimierungsverfahren wie der Zusammenführung von Operationen für gleiche Merkmalstypen und der Kosten-basierten Optimierung von Extraktionsplänen.

Eine experimentelle Evaluation zeigte die praktische Anwendbarkeit der entwickelten Mechanismen und untermauerte die theoretische Abschätzung der Laufzeitverbesserung. Auf einem 4-Prozessorensystem konnten maximale Laufzeitverbesserungen um den Faktor $3,8$ erreicht werden. Es wurde gezeigt, dass der *Overhead*, der durch die Parallelisierung entsteht, gering ausfällt.

Weiterhin untersuchten wir die Laufzeitverbesserungen in Abhängigkeit vom Grad der möglichen Parallelität, der unter anderem durch die Komplexität des Extraktionsplans und die Anzahl der notwendigen Einzeloperationen (also in den meisten Fällen durch die Dokumentgröße) gegeben ist. Es konnte beobachtet werden, dass die relative Laufzeitverbesserung bis zu einem gewissen Grad der möglichen Parallelität ansteigt und bei einem extrem hohen Grad sinkt (z. B. bei extrem großen Dokumenten). Die absolut erzielbaren Laufzeitverbesserungen steigen kontinuierlich.

Inferenz von Regulären Ausdrücken anhand von Beispielentiäten. Die Inferenz von Regulären Ausdrücken ist durch das Modul „REGEX“ in Abb. 6.1.(2) dargestellt. Hier ist das Ziel, anhand einer Menge von Beispiel-Fehlermeldungen eine Menge von Regulären Ausdrücken zu erzeugen, die deren Bildungsvorschrift abbildet, und zur Extraktion von unbekannten Fehlermeldungen aus Texten verwendet werden kann. Die Konzepte zur Regelinferenz wurden in Kap. 4 erläutert.

Das Inferenzverfahren berücksichtigt Zeichen- und Tokenklassen und identifiziert fixe Zeichenketten oder Enumerationen von diesen, die an einer bestimmten Position der Entitäten zu erwarten sind. Es unterstützt damit einen Ausgleich zwischen Genauigkeit und Vollständigkeit der erzeugten Regeln bei der Extraktion. Initial werden alle Beispielentitäten mit Hilfe der definierten Zeichen- und Tokenklassen auf verschiedenen Abstraktionsebenen syntaktisch analy-

siert. Klassifikationssequenzen werden jeweils in einen Präfix- und Suffixautomaten überführt, in dem parallel zu jeder Tokenklassifikation ein Teilautomat auf Zeichenebene exisitiert. Im Weiteren werden so genannte feingranulare Muster (fixe Zeichenketten) auf Basis der Verteilungen von Instanzteilsequenzen im Automaten bestimmt.

Mit Hilfe eines Verfahrens, das auf dem Prinzip der *Minimum Description Length (MDL)* basiert, werden die verschiedenen Abstraktionsebenen für Teilsequenzen des Automaten bewertet und die am besten geeigneten Musterbeschreibungen selektiert. Das *MDL*-Prinzip schafft dabei einen Ausgleich zwischen Komplexität und „Passgenauigkeit" der final erzeugten Regeln. Auch die Ergebnisse aus Präfix- und Suffixautomat werden mit der berechneten „*Description Length*" verglichen und als Ergebnis ein Satz von Regulären Ausdrücken erzeugt.

Eine detaillierte Analyse von verwandten Arbeiten zeigte, dass in der Literatur bisher kein Verfahren beschrieben wurde, welches für Menschen einfach modifizierbare Reguläre Ausdrücke anhand einer gegebenen Menge von Beispielentitäten erlernt. Vielmehr noch zeigte sich, dass bisherige Regellernverfahren im Bereich der Informationsextraktion voneinander abhängige Merkmale verschiedener Granularität nicht unterstützen. Derartige Merkmalskombinationen wurden bisher lediglich durch *Statistische Maschinelle Lernverfahren* wie z. B. *Conditional Random Fields (CRF)* unterstützt, die jedoch bei der Modellbildung auf (annotierte) Trainingsdokumente angewiesen sind und deren manuelle Feinjustierung aufwändig ist.

In der Evaluation konnte für acht typische Typen von Entitäten gezeigt werden, dass das hier vorgestellte Verfahren ähnliche und zum Teil bessere Ergebnisse für die Extraktion von Entitäten liefert als ein *CRF*-basierter Ansatz mit vergleichbarer Merkmalskombination, obwohl dieser anhand von annotierten Beispieldokumenten, die Negativbeispiele zur Verfügung stellen, trainiert wurde und Kontextinformationen auswertet.

Identifikation von Entitäten auf Basis Graph-strukturierter Referenzdaten. Die Identifikation von Entitäten, die in Graph-strukturierten Daten vordefiniert sind, wird in Abb. 6.1.(3) durch vier Module illustriert. Die entsprechend entwickelten Konzepte wurden in Kap. 5 vorgestellt.

Das hier entwickelte Verfahren adressiert im Speziellen Entitäten, die durch umschreibende Bezeichner gekennzeichnet sind und miteinander in Beziehung stehen. Zur Herstellung von initialen Übereinstimmungen wird eine Vorsegmentierung von Texten vorgeschlagen (*POS-Filter* in Abb. 6.1). Textsegmente und Attribute der Entitäten werden in Token-N-Gramme zerlegt und miteinander abgeglichen (*DICT* in Abb. 6.1.(3)). Zur Beschleunigung des Abgleichs indexieren wir Attribut-N-Gramme und filtern zur Reduktion des Speicherbedarfs und Verbesserung der Laufzeit extrem mehrdeutige N-Gramme.

Ausgangspunkt der Bewertung von Entitäten für einen Text ist ein Konfidenzgewicht, welches drei Metriken kombiniert: die Ähnlichkeit der Text- und Attribut-N-Gramme, die Relevanz der Treffer in Textsegment und Referenzdatensatz sowie die Abdeckung der Attribut-N-Gramme hinsichtlich der ursprünglichen Attributinhalte. Zur Berücksichtigung von Beziehungen zwischen Entitäten und Unterstützung von impliziten Referenzen im Text bildet das *GRAPH*-Modul in Abb. 6.1.(3) relevante Konzepte des Referenzdatensatzes in das Dokumentmodell ab, so dass ein so genannter Dokument-Konzept-Graph (DKG) entsteht.

Ein *Spreading Activation*-Verfahren propagiert Konfidenzgewichte im DKG, wobei ein Schwächungsfaktor zur partiellen Einschränkung der Propagation berücksichtigt wird. Resultat der Bewertung ist eine Ordnung von potenziell relevanten Entitäten, die wiederum die für das Dokument relevantesten Teilgraphen des Referenzdatensatzes subsumieren. Die relevantesten Teilgraphen enthalten somit die wahrscheinlichsten Bedeutungen von Textsegement-Attribut-Übereinstimmungen für den betrachteten Kontext (im Beispiel im Kontext eines Absatzes).

Die Analyse von verwandten Arbeiten zeigte, dass ein vergleichbares Verfahren, welches alle oben genannten Bewertungsmechanismen in ähnlicher Art kombiniert, in der Literatur bisher

nicht beschrieben wurde. Die meisten bisherigen Lösungen verwenden *TF-IDF*-Wortgewichte und berücksichtigen damit weder die Sequenz innerhalb vordefinierter Wortfolgen noch die Konstellation von Treffern im Text (wie Überlappungen von Treffern) bei der Bewertung. Auch die Abschwächung bei der Aggregation von einzelnen Treffergewichten und damit einhergehende Berücksichtigung der Struktur des Referenzdatensatzes wurde im Bereich der Identifikation von Entitäten in Texten bisher nicht untersucht.

Die Evaluation in Kap. 5.2 zeigte die Vorteilhaftigkeit der beschriebenen Konzepte, vor allem gegenüber *TF-IDF*-basierten Verfahren, die Graph-strukturierte Referenzdaten zur Identifikation von Entitäten verwenden. Auch konnte gezeigt werden, dass die vorgestellte Methode bessere Ergebnisse als ein maschinelles Lernverfahren und eine Websuchmaschine, die für diese Problemstellungen adaptiert wurden, liefert.

6.2 Zukünftige Erweiterungsmöglichkeiten

Mögliche Erweiterungen der hier entwickelten Verfahren wurden bereits in den entsprechenden Kapiteln erläutert (siehe Kap. 3.5, Kap. 4.4 und Kap. 5.4). Im Kontext von *RapidUIM* sind verschiedenste Erweiterungen vorstellbar. Eine verteilte Ausführung von IE-Systemen, die mit *RapidUIM* modelliert sind, wurde bereits auf Basis eines *High Performance Message Hubs* [Vin06], welches Dokumente verteilten Programminstanzen zuweist, realisiert.

Hinsichtlich der Skalierbarkeit stellt sich damit zukünftig die Frage nach effizienten Speicherlösungen, die für den entstehenden, extrem hohen Datendurchsatz optimiert sind. Eine verteilte Datenspeicherung eignet sich am besten zur Beseitigung dieses Engpasses. Als zukünftiges Forschungsproblem stellt sich damit die Frage, wie Extraktionsdaten optimal hinsichtlich der effizienten Speicherung und Abfrage verteilt werden können.

Im Kontext der parallelen Datenverarbeitung verspricht eine Adaption der in dieser Arbeit entwickelten Mechanismen für Grafikkarten, die zum Teil mehr als 30 Prozessoren besitzen, eine enorme Beschleunigung. Weiterhin stellt sich die Frage, wie eine Parallelverarbeitung in Verfahren zur Kosten-basierten Optimierung von Extraktionsplänen berücksichtigt werden kann.

In *RapidUIM* können Extraktionspläne mit Dokument-übergreifenden Operationen ohne weiteres modelliert, jedoch nicht wie in Kap. 3 beschrieben, ausgeführt werden, da einzelne Zwischenergebnisse, die bei Dokument-übergreifenden Operationen benötigt werden, persistent zu speichern sind. Eine Erweiterung des vorgestellten Datenverarbeitungsverfahrens zur effizienten Unterstützung von derartigen Operationen ist ebenfalls ein interessantes zukünftiges Forschungsproblem.

Im Kontext der Inferenz von Regulären Ausdrücken ist zukünftig eine Erweiterung um Merkmale höherer Granularität denkbar, so dass das Verfahren einfach mit bekannten Methoden aus dem Bereich der *Wrapper-Induktion* kombiniert werden kann. Weiterhin ist die Kombination des vorgestellten Verfahrens mit anderen Methoden, wie traditionellen Regellernverfahren oder Statistischen Maschinellen Lernverfahren anzustreben, so dass auch wertvolle Kontextmerkmale in Anwendungsfällen mit gegebenen Trainingskorpus im vollen Umfang unterstützt werden können.

Das hier beschriebene Verfahren zur Identifikation von Entitäten auf Basis von Graph-strukturierten Referenzdaten sieht eine Reihe von Parametern vor. Eine automatische Bestimmung von geeigneten Parametern anhand nicht annotierter Dokumente ist ein interessantes zukünftiges Forschungsfeld. Zum Beispiel ist es vorstellbar, dass Abschwächungsfaktoren für jede einzelne Kante im Referenzdatensatz auf Basis von gemeinsamen Vorkommen verschiedener Attribut-N-Gramme in nicht annotierten Texten bestimmt werden können. Weiterhin ist eine weitere algorithmische Laufzeitoptimierung für das in dieser Arbeit vorgestellte Verfahren anzustreben.

6.3 Zukünftige Forschungsfragen in der Enterprise Search

Die vorliegende Arbeit untersuchte die einfache und effiziente Erstellung von Extraktionsprogrammen und die Ausnutzung von unternehmensinternen strukturierten Referenzdaten zur Verbesserung von *Enterprise Search*-Anwendungen. Effektive Extraktionsprogramme für die *Enterprise Search*, in denen wie bei Verifikation von Rechnungen wiederkehrende Dokumentverarbeitungsoperationen notwendig sind, können durch Experten mit Hilfe der vorgestellten Methoden auf effiziente Art und Weise entwickelt werden. Es stellen sich jedoch eine Reihe von zukünftigen Forschungsfragen in diesem Bereich:

Ad-hoc Informationsextraktion aus externen und internen Datenquellen. Es gibt eine Vielzahl von Geschäftsprozessen, in denen ad-hoc Informationsextraktionsprogramme, die von Nutzern zur Laufzeit erstellt werden, einen Mehrwert versprechen. Weiterhin sind Informationsextraktionsprogramme bezüglich geschäftsrelevanter Entitäten in bestimmten Anwendungsfällen nicht auf das Intranet oder die Internetpräsenz eines Unternehmens begrenzt, z. B. um mögliche Kunden oder aktuelle Produktentwicklungen von Konkurrenten zu finden. Für solche ad-hoc Extraktionsprogramme sind weitergehende Ansätze wünschenswert, die es technisch wenig versierten Endnutzern ermöglichen, Extraktionsprogramme auf einfachste Weise (z. B. als eine Art Stichwortanfrage) zu erstellen. Auch sind Techniken, die die Auswahl von potenziell relevanten Dokumenten unterstützen, notwendig, die über bisherige Verfahren des *Focused Querying* hinausgehen. Ein Problem besteht zum Beispiel darin, dass Dokumente verschiedenster interner und externer Quellen auf deren Relevanz für ein ad-hoc Extraktionsprogramm beurteilt werden müssen.

Kontinuierliche Informationsextraktion und Datenanalyse. Weiterhin sind auch Extraktionssysteme vorstellbar, die auf Dokumentströmen z. B. von *Micro-Blogging*-Plattformen operieren. Durch die kontinuierliche Extraktion von Kundenmeinungen und Entitäten sowie deren Korrelation mit geschäftsrelevanten Ereignissen, wie z. B. dem Start von Marketingkampagnen oder der Präsentation von neuen Produkten durch Konkurrenzunternehmen, könnten Unternehmensstrategien quasi in Echtzeit an Kundenwünsche angepasst werden. Die hier vorgeschlagenen Konzepte können in derartigen Szenarien die Implementation von Informationsextraktionsprogrammen unterstützen. Weitergehende Forschungsthemen betreffen insbesondere die zeitlichen Aspekte, wie die Aggregation und Auswertung von kontinuierlichen Textdaten mit Ereignissen, die in der Literatur bisher nicht in ausreichendem Maße erforscht wurden.

Nutzerinteraktion und -unterstützung. Die Navigation und Visualisierung von Dokument- bzw. Korpusinhalten ist eine wichtige Komponente für einen Paradigmenwechsel vom „Suchen“ hin zum „Finden“ von Informationen. Die hier vorgestellte Anwendung zur statistischen Analyse von Dokumentenkorpora basiert auf einem *OLAP*-Werkzeug und erlaubt eine Hypothesen-getriebene Analyse von Korpusinhalten für geschulte Analysten. Mit Hilfe von Navigationsoperationen wie *Slice*, *Dice*, *Roll Up* oder *Drill Down* kann in Extraktionsergebnissen entlang verschiedenster Dimensionen navigiert werden. Um derartige detaillierte Analysen für technisch wenig versierte Endnutzer verfügbar zu machen, sind Verfahren notwendig, die einem Nutzer mögliche Hypothesen und Navigationspfade vorschlagen und Ergebnisse in geeigneter Art und Weise visualisieren.

Zukünftige Herausforderungen liegen daher vor allem in der Einbindung von technisch wenig versierten Endnutzern in die Erstellung von Informationsextraktionsprogrammen und die Analyse von Extraktionsergebnissen, da eben diese Nutzer zumeist über das größte Domänenwissen verfügen. Zukünftige Forschungsfragen in dem Bereich der Informationsextraktion für die *Enterprise Search* liegen daher in dem Gebiet der vereinfachten Erzeugung von Extraktionsprogrammen und vor allem in der Konzeption von Anwendungen, die es Endnutzern ermöglichen, Extraktionsergebnisse verschiedenster Datenquellen auf einfachste Weise zu geschäftsrelevanten Informationen zu aggregieren und diese zu explorieren.

A Anhang

Der Anhang dieser Arbeit enthält Beispieldokumente, die für die Evaluation in Kap. 3.3 und Kap. 4.2 verwendet wurden sowie – als Ergänzung zu Kap. 4.2 – eine detaillierte Auswertung der Experimente zur Evaluation der Inferenz von Regulären Ausdrücken.

A.1 Dokumentfragmente zur Evaluation der Intra-Dokument-Parallelisierung

In Bsp. A.1 ist das Textfragment dargestellt, das für die Experimente in Kap. 3.3.1 und Kap. 3.3.3 verwendet wurde. Das Textfragment wurde je nach Konfiguration dupliziert und die Duplikate zu einem Dokument für die Evaluation zusammengefügt. Ziel war es in Kap. 3.3.1 den *Raum* („*HAW GN-K35*“) und die *Zeit* („*10:30 AM - 11:30 AM, April 7, 2004*“) zu extrahieren. Der Extraktionsplan in Kap. 3.3.3 zerlegte das Dokument in Absätze, so dass die Absätze (entsprechen quasi den ursprünglichen Fragmenten) nebenläufig verarbeitet wurden. Aus diesen wurden jeweils die Raum- und Zeitangaben extrahiert und zu Annotationen vom Typ „*Treffen*“ aggregiert.

```
Speech recognition has matured to the point where it is now being widely
applied in a range of applications including desktop dictation, cell phone
name dialing, agent technology, automated operator services, telematics,
call center automation and help desks. Dave Ferrucci will give a UIMA
overview and discuss the types of component metadata that UIMA components
provide in room HAW GN–K35. The presentation is scheduled at
10:30 AM – 11:30 AM, April 7, 2004.
```

Bsp. A.1: Dokumentfragment zur Evaluation von Laufzeitverbesserungen bei Aufgabenparallelität und Datenparallelität in Situationen mit direkten Datenabhängigkeiten

In Bsp. A.2 ist das HTML-Fragment, das im ersten Szenario in Kap. 3.3.2 verwendet wurde, dargestellt. Wie oben beschrieben, wurde dieses Fragment je nach Konfiguration dupliziert und zu einem künstlichen Dokument zusammengeführt. Der Extraktionsplan in Kap. 3.3.2 zerlegt das Dokument in Tabellenzellen, die jeweils einzeln durch nachfolgende Instanzen eines Moduls zur Extraktion von Raumnummern (hier „*HAW GN-K35*“) analysiert wurden.

```
<tr>
<td>10:30 AM – 11:30 AM</td>
<td>April 7, 2004</td>
<td>HAW GN-K35</td>
<td>Dave Ferrucci (dave.ferrucci@uima.org)</td>
<td>Dave Ferrucci will give a UIMA overview and
discuss the types of component metadata that UIMA components provide</td>
</tr>
```

Bsp. A.2: Dokumentfragment zur Evaluation der Datenparallelität bei direkten Datenabhängigkeiten

Das Textfragment, das zur Evaluation des zweiten in Kap. 3.3.2 evaluierten Szenarios verwendet wurde, ist in Bsp. A.3 dargestellt. Der Extraktionsplan in Kap. 3.3.2 zerlegte das Dokument, welches auf Basis des Textfragments erstellt wurde, in Sätze (die quasi dem Fragment ensprechen), wendete eine morphologische Analyse zur Filterung von Substantivgruppen an und glich diese Gruppen über ein approximatives Abgleichverfahren mit einem Wörterbuch von Ortsbezeichnungen ab.

```
California adjoins the Pacific Ocean, Oregon, Nevada, Arizona, and the
    Mexican state of Baja California.
```

Bsp. A.3: Dokumentfragment zur Evaluation der Datenparallelität bei mehrstufigen direkten Datenabhängigkeiten

A.2 Beispieldokumente für Evaluation der Regelinferenz

In Bsp. A.4 ist ein Beispieldokument für den Rechnungskorpus, der zur Evaluation in Kap. 4.2 verwendet wurde, dargestellt. Basierend auf Beispielen für Rechnungsnummern (wie „*BK-0909-45672*“ in Bsp. A.4), Swift-Codes (z. B. „*BOFAUS4N*“) und Telefonnummern („*001-412-41245667*“) wurden Reguläre Ausdrücke zur Extraktion von unbekannten Entitäten erlernt. Weitere Details zu den Experimenten sind in Kap. A.3 zu finden.

```
              INVOICE                ORDER
09/09/2008
              BK-0909-45672                2335975

 AMS                          TO AKRON                      SHIP TO AKRON
166 South High Street    Heating Technologies Inc.   Heating Technologies Inc.
Akron, OH 44308          166 North High Street       166 North High Street
Phone 001-412-41245667   Akron, OH 44308             Akron, OH 44308

QTY DESCRIPTION      UNIT PRICE   DISCOUNT   LINE TOTAL
1        Laptop Thinkpad    1500,00 USD              3000,00US

Make all checks payable to Bank of America,
AKRON Bank Code SWIFT       Account
              120000     BOFAUS4N   110000072543

Thank you for your business!
```

Bsp. A.4: Beispiel einer Rechnung nach der Verarbeitung durch eine OCR-Software

In Abb. A.1 ist ein Beispiel des Korpus zur Evaluation der Extraktion von Notebookbezeichnern wie „*HP EliteBook Mobile*“, „*HP EliteBook 2739p*“ oder „*Compaq nc4400*“ dargestellt.

Das Beispiel für einen Thread des SCN-Forums in Abb. A.2 illustriert Dokumente, die zur Bemessung der Extraktionsqualität für Java-Fehlernachrichten (z. B. „*System.Runtime.InteropServices.COMException*“) verwendet wurden. Es verdeutlicht die Ähnlichkeit

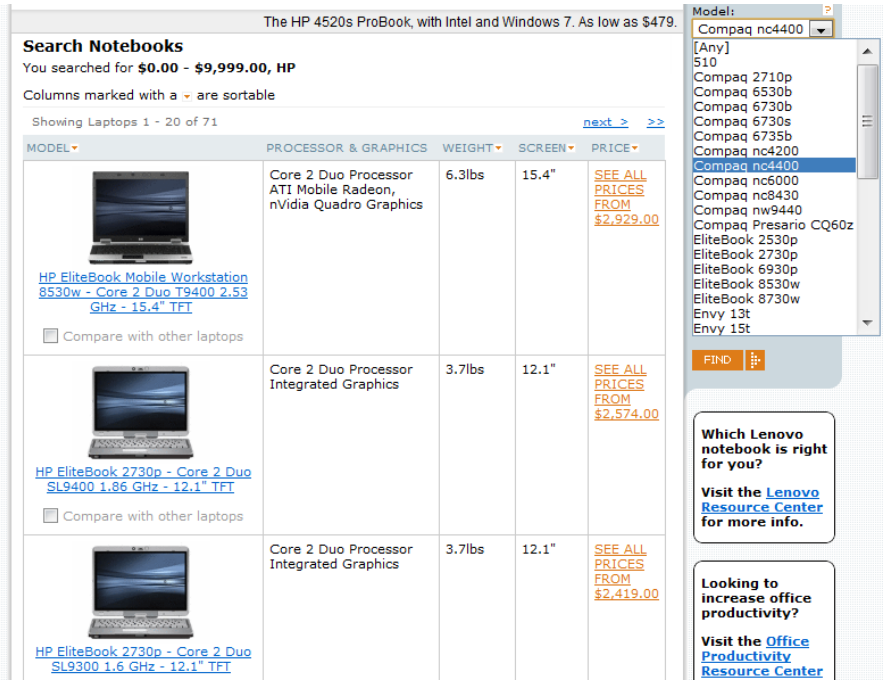


Abb. A.1: Beispiel für Bewertungsseite von Notebooks

von anderen Konzepten verschiedener Programmiersprachen mit der Syntax von Fehlernachrichten.

Unhandled Exception: Invalid field name
Posted: Oct 16, 2008 12:46 PM

Reply

Hi guys,

i developed a ASPX application, using CrystalReports 10.

If i open the ASPX page which generates a Crystal Report it works on all PCs.

BUT on PCs with a japanese XP it won't.

I allways get this error:

Invalid field name.
Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Runtime.InteropServices.COMException: Invalid field name.

Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:

[*COMException (0x800001fd):*

Invalid field name.]
CrystalDecisions.ReportAppServer.ClientDoc.ReportClientDocumentClass.Open(Object& DocumentPath, Int32 Options) +0
CrystalDecisions.ReportAppServer.ReportClientDocumentWrapper.Open(Object& DocumentPath, Int32 Options) +72
CrystalDecisions.ReportAppServer.ReportClientDocumentWrapper.EnsureDocumentIsOpened() +218

Abb. A.2: Beispiel für Thread des SCN-Forums mit Java-Fehlermeldungen

In Abb. A.2 sehen wir einen Ausschnitt des *„ReLIE-Korpus“* für den Entitätstyp *„Softwareprodukt“*. Im Korpus werden potenzielle Entitäten (z. B. *„DTD XHTML 1.0“*) und zugehöriger linker und rechter Kontext gespeichert. Für die Evaluation in Kap. 4.2 wurden linker und rechter

Kontext sowie die mögliche Entität zu Textfragementen zusammengeführt und zusätzlich nicht gekennzeichnete Entitäten manuell annotiert, so dass jedes Vorkommen einer Entität (auch im Kontext) zur Evaluation herangezogen werden konnte (siehe Kap. 4.2.1 für Details).

```
[..]
<context>
<pos>
<left><!-- </left>
<right> Mon Oct 10 22:16:44 2005 -->      <html>  <head>  <title>College of
    Literature, Science, and the Art</right>
</pos>
</context>
<anchor>Vignette V6</anchor>

<context>
<left> <!DOCTYPE html PUBLIC \-//W3C//</left>
<right> Transitional//EN\ \http://www.w3.org/TR/xhtml1/DTD/xhtml1-
    transitional.dtd\>  <html xmlns=\http://w</right>
</context>
<anchor>DTD XHTML 1.0</anchor>
[..]
```

Bsp. A.5: Beispiel für Textfragmente des *„ReLIE-Korpus“*

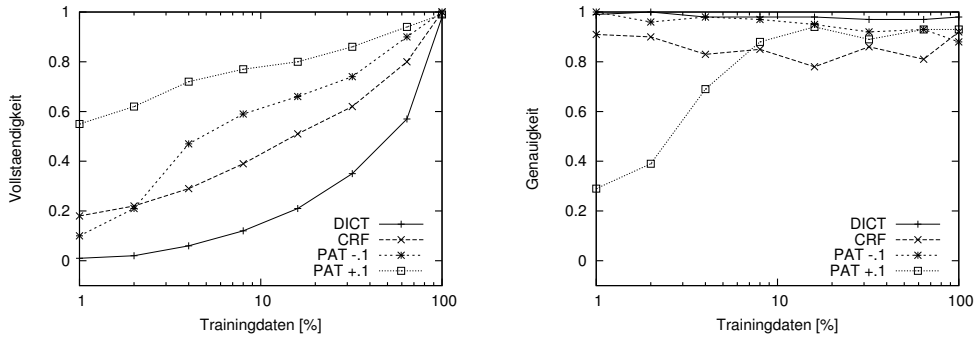
A.3 Detaillierte Ergebnisse zur Evaluation der Regelinferenz

Im Folgenden werden die detaillierten Ergebnisse der Evaluation für die Inferenz von Regulären Ausdrücken im Vergleich zu einem CRF-Ansatz dargestellt (siehe Kap. 4.2). Es werden Experimente für die zwei Standardkonfigurationen des in Kap. 4.1 vorgestellten Verfahrens gezeigt: die Konfiguration mit $\beta = -0,1$ liefert im Allgemeinen eine hohe Genauigkeit und eine Verwendung von $\beta = 0,1$ einer hohen Vollständigkeit. Erstere Konfiguration wird im Folgenden als *PAT-.1* und zweitere mit *PAT+.1* bezeichnet. Ein Wörterbuch-basiertes Verfahren, das Textfragmente und Beipielentiäten über einen exakten Zeichenkettenvergleich aufeinander abbildet, wird durch *DICT* illustriert. Das Vergleichsverfahren *Conditional Random Fields* ist als *CRF* gekennzeichnet.

Weiterhin diskutieren wir Beispiele für erlernte Reguläre Ausdrücke, die unter Verwendung von 100% der Trainingsdaten erzeugt wurden. Wir verwenden dieselbe Syntax, wie in Kap. 4. Durch Spitzklammern notierte Merkmale beschreiben Zahlen- oder Tokenklassen (siehe Kap. 4.1.1 auf Seite 82). Tokenklassen werden durch ein zusätzliches *Kleene Plus* notiert. Alternation werden durch „|“ beschrieben. Klammern fassen Instanzteilsequenzen zusammen und werden hier weiterhin zur Abgrenzung von relevanten Blöcken der Regulären Ausdrücke verwendet. Das *Escape*-Symbol ist „\“. Leerzeichen werden durch „␣“ dargestellt. Weiterhin verwenden wir „[...]“ als Kennzeichnung, wenn in Enumerationen von feingranularen Mustern einzelne Werte aus Platzgründen ausgeblendet wurden.

Die Extraktionsqualität für Telefonnummern im Rechnungskorpus ist in Abb. A.3 zu sehen. *PAT+.1* erzielt sehr früh eine hohe Vollständigkeit, wogegen eine akzeptable Genauigkeit bei 4% der Trainingsdaten erreicht wird. *PAT-.1* erzeugt immer sehr genaue Ergebnisse, jedoch eine geringere Vollständigkeit als *PAT+.1*. Beide Konfigurationen sind in der Vollständigkeit dem *CRF*-Ansatz überlegen.

In Tab. A.1 sind drei Beispiele von den 28 erlernten Regulären Ausdrücken unter Verwendung von 100% der Trainingsdaten dargestellt. Beide Konfigurationen stellen typische Syntaxmerkmale von Telefonnummern, wie z. B. ein führendes „+“ oder eingegeklammerte Vorwahlen dar. *PAT-.1* erlernte weiterhin charakteristische Vorwahlen, die im Kontext des Unternehmens, das



(a) Vollständigkeit (b) Genauigkeit

Abb. A.3: Extraktionsqualität für Telefonnummern (Rechnung)

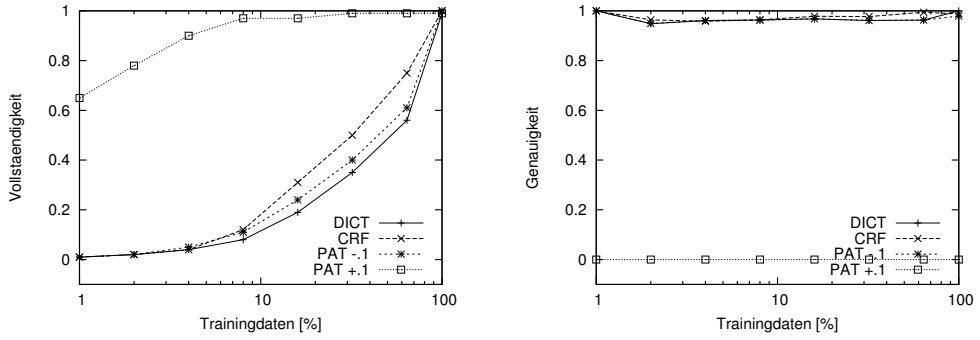
die Rechnungen erhielt, relevant sind und oft in den Beispielen auftretende Zahlen, die in der Mitte und am Ende von Telefonnummern zu finden sind. In Kombination mit diesen Zahlen werden typische Längen von Zahlenblöcken in den Regulären Ausdrücken abgebildet.

Tab. A.1: Beispiele für Reguläre Ausdrücke zur Extraktion von Telefonnummern (Rechnung)

***PAT-.1* (3/28)**
(\+|\()(0621|07251|[...]|0351|06222)(\)|-)(_)((4)⟨NB⟩{3}|⟨NB⟩{3}(5|4)⟨NB⟩)⟨SC⟩⟨NB⟩$^+$
(06227|06205|0049|[...]|06221|06803)(-|/)(_)⟨NB⟩$^+$⟨WT⟩(4)⟨NB⟩{2}
(06227|06205|0049|[...]|06221|06803)(-|/)(_)⟨NB⟩$^+$⟨SC⟩⟨NB⟩$^+$
***PAT+.1* (3/28)**
(\+|\()⟨NB⟩{1,5}(\)|-)(_)⟨NB⟩$^+$⟨SC⟩⟨NB⟩$^+$
⟨NB⟩{1,5}(-|/)(_)⟨NB⟩$^+$⟨WT⟩⟨NB$^+$⟩
(06227|06205|0049|[...]|06221|06803)(-|/)(_)⟨NB⟩$^+$⟨SC⟩⟨NB⟩$^+$

Die Regulären Ausdrücke, die durch *PAT+.1* erlernt wurden (siehe Tab. A.1), abstrahieren im Vergleich zu *PAT-.1* in höherem Maße von den Instanzen. Zum Teil werden typische Längeninformationen für Zahlenblöcke in die Regulären Ausdrücke übernommen. Im letzten Beispiel werden wie in *PAT-.1* Vorwahlen als charakteristisch bewertet.

Bei der Extraktion von SWIFT-Codes aus Rechnungen ist, wie in Abb. A.4 zu sehen ist, die Konfiguration *PAT+.1* wenig geeignet. Beide, *PAT-.1* und *CRF* liefern im Gegensatz eine hohe Genauigkeit. Die Vollständigkeit ist im Vergleich zu *DICT* bis zu einer Verwendung von 16% der Trainingsdaten jedoch lediglich als befriedigend zu bewerten.



(a) Vollständigkeit (b) Genauigkeit

Abb. A.4: Extraktionsqualität für SWIFT-Codes (Rechnung)

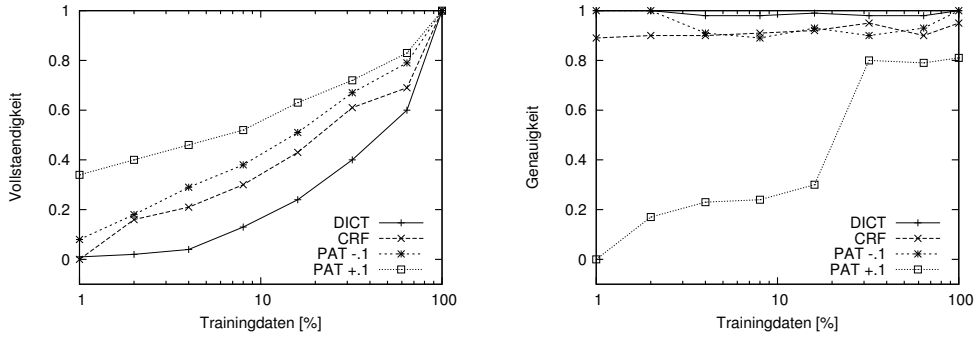
In Tab. A.2 sind die zwei erlernten Regulären Ausdrücke für die Extraktion von SWIFT-Codes für *PAT-.1* und *PAT+.1* dargestellt. In der Konfiguration *PAT-.1* werden Enumerationen von charakteristischen Zeichen erlernt, wodurch die hohe Genauigkeit, aber auch die geringe Vollständigkeit bei einer Verwendung von weniger als 16% der Trainingsdaten zu erklären ist. So denn typische in SWIFT-Codes auftretende Zeichen erlernt sind, steigt die Vollständigkeit.

PAT+.1 abstrahiert stärker von den gegebenen Instanzen auf Basis von Tokenklassen. Da in dem Korpus eine Vielzahl von syntaktisch ähnlichen Konzepten auftreten, geht die Genauigkeit dieser Ausdrücke gegen 0. Durch die verschiedenen Längen von SWIFT-Codes wird zu Tokenklassen und nicht zu Zahlenblöcken mit bestimmter Länge abstrahiert. Ein Test, ob Reguläre Ausdrücke, die typische Zahlenblocklängen von SWIFT-Codes beschreiben, bessere Ergebnisse liefern, zeigte, dass eine solche Abstraktion die Genauigkeit nicht wesentlich verbessert. Der Grund dafür ist, dass die Konzepte mit ähnlicher Syntax ebenso eine ähnliche Länge aufweisen.

Tab. A.2: Beispiele für Reguläre Ausdrücke zur Extraktion von Swift-Codes (Rechnung)

***PAT-.1* (2/2)**	***PAT+.1* (2/2)**
(G|H|E|C|D|A|B|O|P|N|L|I|W|V|T|S|R)(C|A|B|O|K|L|I)[...](E|U|T|S|K|L|I)	⟨UC⟩$^+$
(H|W|F|S|A|R|B|O|P|N)(V|S|A|B|O|P|N|I|Z)[...](2|4|3|C|R|E|S|C|H|Z)	⟨UC|NB⟩$^+$

Bei der Extraktion von Rechnungsnummern aus dem Rechnungskorpus ist, wie in Abb. A.5 zu sehen ist, die Konfiguration *PAT+.1* besser geeignet. Sie liefert eine hohe Vollständigkeit und ab dem Training mit 16% der Daten auch eine gute Genauigkeit. *PAT-.1* und *CRF* liefern miteinander vergleichbare Ergebnisse. An den Beispielen für erzeugte Reguläre Ausdrücke in Tab. A.5 ist zu sehen, dass *PAT-.1* sehr spezifische Muster, wie charakteristische Buchstaben am Beginn einer Rechnungsnummer erlernt. Hier liefert die Darstellung des Präfixes als Buchstabenfolge der Länge 2 in den Beispielen zu *PAT+.1* eine höhere Vollständigkeit bei einer guten Genauigkeit (ab Verwendung von 32% der Trainingsdaten).



(a) Vollständigkeit (b) Genauigkeit

Abb. A.5: Extraktionsqualität für Rechnungsnummern (Rechnung)

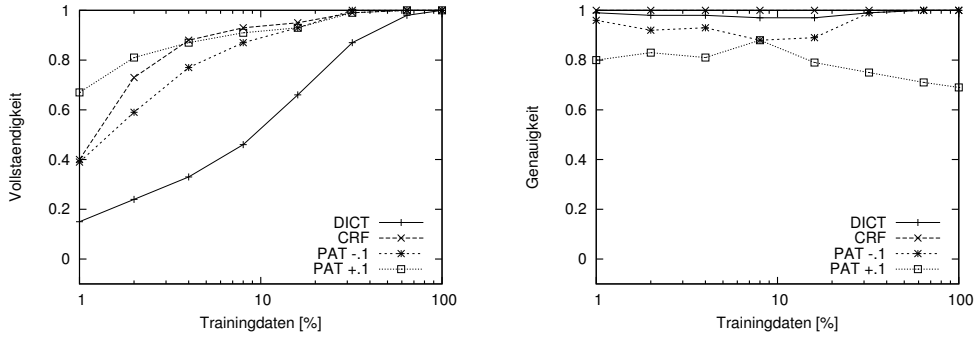
Tab. A.3: Beispiele für Reguläre Ausdrücke zur Extraktion von Rechnungsnummern (Rechnung)

***PAT-.1* (3/7)**	***PAT+.1* (3/7)**
(G|V|C|B|P|K)(H|S|D|B|O|K)(-|/)⟨NB⟩{4,9}	⟨UC⟩{2}(-|/)⟨NB⟩{4,9}
(2008|19040|[...]|0909)(-|\.)⟨NB⟩$^+$(-)(47)⟨NB⟩{4}	(2008|19040|[...]|0909)(-|\.)⟨NB⟩$^+$(-)⟨NB⟩$^+$
(#)⟨NB⟩$^+$	(#)⟨NB⟩$^+$

In der zweiten Regel in Tab. A.3 werden typische Zahlen, wie das Jahr („2008“) oder die Kombination von Monat und Jahr („0909“) als charakteristisch klassifiziert. In *PAT+.1* werden diese Muster erst ab einem Training mit mehr als 16% der Beispieldaten berücksichtigt, wodurch

der Sprung in der Genauigkeit zu erklären ist. Die Syntaxregel „#⟨NB⟩$^+$“ tritt in den Regulären Ausdrücken von *PAT-.1* und *PAT+.1* auf und ist in diesem Korpus eine sehr typische Beschreibung von Rechnungsnummern. Die zugrundeliegenden Instanzen variieren in ihrer Länge, wodurch sich eine Merkmalsrepräsentation auf Tokenebenen für die Zahlenfolge ergibt.

In Abb. A.6 ist die Extraktionsqualität für den Anwendungsfall „Notebookbezeichner“ dargestellt. *PAT-.1* und *PAT+.1* sind *CRF* insbesondere bei wenigen gegebenen Beispielen überlegen. *PAT+.1* erzeugt wesentlich früher eine hohe Vollständigkeit als *PAT-.1*. Die Genauigkeit von *PAT-.1* ist jedoch meist höher.



(a) Vollständigkeit (b) Genauigkeit

Abb. A.6: Extraktionsqualität für Notebookbezeichnern (Bewertungen)

In Tab. A.4 sind Beispiele für Reguläre Ausdrücke zur Extraktion von Notebookbezeichnern zu sehen. Hier liegt eine typische präfixartige Syntax vor, wie insbesondere in den Beispielen für *PAT-.1* zu sehen ist. Die Berücksichtigung von zusätzlichen feingranularen Mustern durch *PAT-.1* im Vergleich zu *PAT+.1* führt zur besseren Genauigkeit.

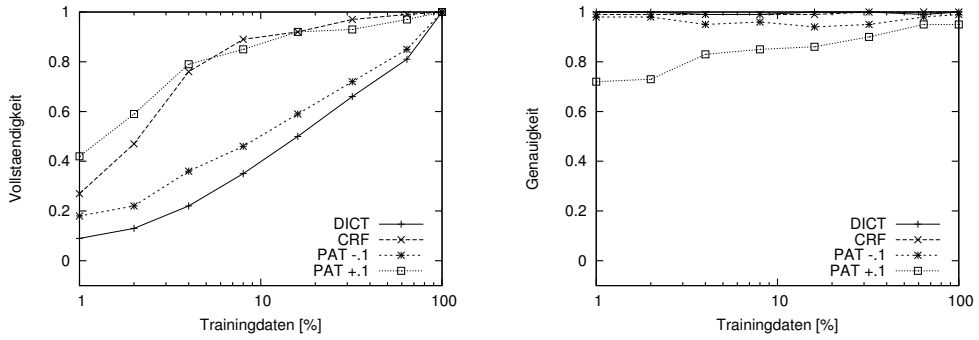
Tab. A.4: Beispiele für Reguläre Ausdrücke zur Extraktion von Notebookbezeichnern (Bewertungen)

***PAT-.1* (3/32)**	***PAT+.1* (3/32)**
(HDX)(_)(1)⟨NB⟩⟨LC⟩	(HDX)(_)⟨NB|LC⟩$^+$
(Mini)(_)(10)⟨NB⟩{2}(_)⟨UC|LC⟩$^+$	(Mini)(_)⟨NB|LC⟩$^+$(_)⟨UC|LC⟩$^+$
(Pavilion)(_)(z|d|t|x)(v|e|h|x)⟨NB⟩{2}(00)	(Pavilion)(_)⟨NB|LC⟩$^+$

Bei der Extraktion von Java-Fehlermeldungen (siehe Abb. A.7) liefern *PAT+.1* und *CRF* vergleichbare Ergebnisse. *PAT+.1* führt zu einer höheren Vollständigkeit, wogegen *CRF* eine bessere Genauigkeit bietet. *PAT-.1* ist insbesondere hinsichtlich der Vollständigkeit den anderen Verfahren unterlegen.

Die sehr guten Ergebnisse von *CRF* sind damit zu begründen, dass es Merkmale immer für einen Token und dessen Kontext bewertet. Die sequentiellen Strukturmerkmale (wiederkehrende Folge von Punkten und Token) werden daher durch das *CRF* implizit abgebildet. Zukünftig ist eine Erweiterung des hier vorgestellten Verfahrens vorstellbar, das auch die Wiederholung von Merkmalsequenzen berücksichtigt. Es ist anzunehmen, dass durch eine derartige Erweiterung in dem hier diskutierten Anwendungsfall wesentlich bessere Ergebnisse als mit einem CRF-Verfahren erreicht werden können, da *PAT+.1* bereits ähnliche Ergebnisse liefert.

In den Beispielen für Reguläre Ausdrücke zur Extraktion von Java-Fehlermeldungen in Tab. A.5 ist zu sehen, dass es sich hier um typische Suffix-artige Bezeichner handelt (enden immer mit „Exception“, „Error“ oder „Message“). Auch lässt sich der Grund für die geringe Vollständigkeit in den Ergebnissen von *PAT-.1* ablesen. Da sehr viele ähnliche Beipielentitäten gegeben sind, wie z. B. Fehler mit dem Präfix „com.adobe“ werden eine große Anzahl an feingranularen Mustern berücksichtigt. Da der syntaktische Aufbau von Fehlermeldungen (Punk-



(a) Vollständigkeit (b) Genauigkeit

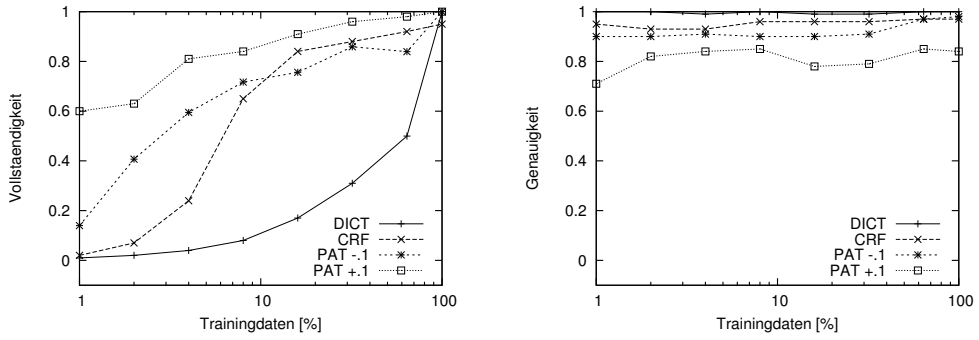
Abb. A.7: Extraktionsqualität für Java-Fehlermeldungen (SCN-Forum)

te und Token) sehr charakteristisch ist, ist eine höhere Abstraktion, wie sie in den Regulären Ausdrücken von *PAT+.1* vorliegt, besser geeignet.

Tab. A.5: Beispiele für Reguläre Ausdrücke zur Extraktion von Java-Fehlermeldungen (SCN-Forum)

***PAT-.1* (4/23)**
(java)(\.)(lang)(\.)⟨UC|LC⟩$^+$(Exception|Error|Message)
(od|i)⟨UC|LC⟩$^+$(\.)⟨UC|LC⟩$^+$(\.)⟨UC|LC⟩$^+$(Exception|Error|Message)
(com)(\.)(adobe)(\.)(ads)(\.)(operation)(\.)⟨UC|LC⟩$^+$(handle)⟨UC|LC⟩$^+$(Exception|Error|Message)
⟨UC|LC⟩$^+$(\.)⟨UC|LC⟩$^+$(\.)⟨UC|LC⟩$^+$(Exception|Error|Message)
***PAT+.1* (4/23)**
⟨LC⟩{4}⟨SC⟩⟨LC⟩{4}⟨SC⟩⟨UC|LC⟩$^+$(Exception|Error|Message)
⟨LC⟩$^+$⟨UC|LC⟩$^+$⟨SC⟩⟨UC|LC⟩$^+$⟨SC⟩⟨UC|LC⟩$^+$(Exception|Error|Message)
⟨LC⟩$^+$⟨SC⟩⟨LC⟩$^+$⟨SC⟩⟨LC⟩$^+$⟨SC⟩⟨UC|LC⟩$^+$⟨LC⟩$^+$⟨UC|LC⟩$^+$(Exception|Error|Message)
⟨UC|LC⟩$^+$⟨SC⟩⟨UC|LC⟩$^+$⟨SC⟩⟨UC|LC⟩$^+$(Exception|Error|Message)

Die Extraktionsqualität für Telefonnummern aus dem „*ReLIE-Korpus*“ (siehe Abb. A.8) zeigt ein ähnliches Verhalten wie bei der Extraktion von Telefonnummern aus Rechnungen (siehe Abb. A.3), wobei jedoch die Genauigkeit in den Resultaten von *PAT+.1* höher ist. *PAT+.1* liefert daher hier die besten Ergebnisse, da zu einem frühen Zeitpunkt eine hohe Vollständigkeit, gepaart mit einer guten Genauigkeit, erreicht wird.



(a) Vollständigkeit (b) Genauigkeit

Abb. A.8: Extraktionsqualität für Telefonnummern (ReLIE-Korpus)

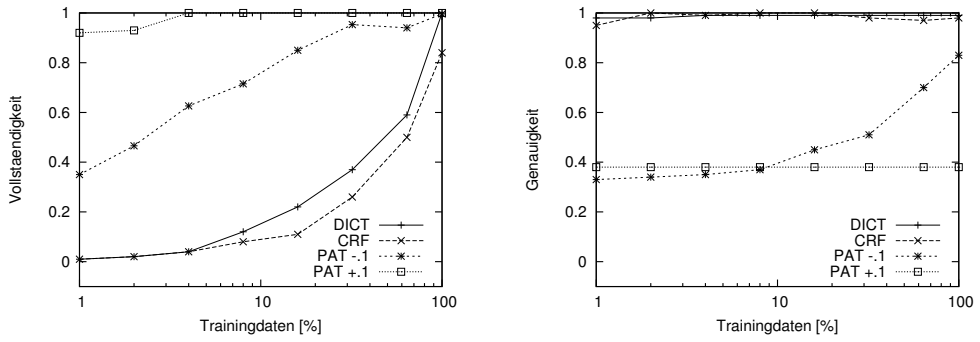
Die Beispiele für Reguläre Ausdrücke zur Extraktion von Telefonnummern aus dem „ReLIE-Korpus“ sind in Tab. A.6 zu sehen. Der erste Ausdruck berücksichtigt typische Vorwahlnum-

mern, wobei *PAT+.1* eine abstraktere Beschreibung generiert. Der zweite Ausdruck beschreibt die Klasse von Telefonnummern, die in den USA für Anrufer kostenlos sind und im Korpus häufig auftreten. Als Sonderzeichen („⟨SC⟩“) treten hier in den Beispielinstanzen Punkte, Bindestriche und Klammern auf. Die letzte Regel beschreibt eine typische Syntax für Telefonnummern mit Leerzeichen und Bindestrichen, die in den USA typischerweise verwendet wird. Die ersten beiden Zahlenblöcke besitzen eine charakteristische Länge.

Tab. A.6: Beispiele für Reguläre Ausdrücke zur Extraktion von Telefonnummern (ReLIE-Korpus)

***PAT-.1* (3/7)**	***PAT+.1* (3/7)**
(\(|\+)(1|313|...|810)(\.|\))(_)⟨NB⟩{3}(-)⟨NB⟩$^+$	(\(|\+)(1|313|...|810)(\.|\))(_)⟨NB⟩$^+$⟨SC⟩⟨NB⟩$^+$
(1)⟨SC⟩(800)⟨SC⟩(723)⟨SC⟩⟨NB⟩$^+$	(1)⟨SC⟩(800)⟨SC⟩⟨NB⟩$^+$⟨SC⟩⟨NB⟩$^+$
⟨NB⟩{1,3}(_)⟨NB⟩{3}(-)⟨NB⟩$^+$	⟨NB⟩{1,3}(_)⟨NB⟩{3}(-)⟨NB⟩$^+$

In Abb. A.9 sind die Ergebnisse für die Extraktion von Bezeichnern für Lehrveranstaltungen dargestellt. Hier ist *PAT+.1* den anderen Verfahren bei Betrachtung des F-Measures (siehe Abb. 4.6 auf S. 101) überlegen. *PAT-.1* liefert jedoch erst bei Verwendung von 16% der Trainingsdaten bessere Werte für die Genauigkeit als *PAT+.1*. Die Parameter des CRF-Verfahrens konnten in diesem Anwendungsfall aufgrund von sich überschneidenden Merkmalen im Korpus nicht optimal bestimmt werden, wodurch dessen Ergebnisse in Mitleidenschaft gezogen wurden (siehe auch Kap. 4.2.2). Das Problem besteht darin, dass typische Merkmale der Entitäten (wie z. B. deren Token) im Korpus häufig nicht als Bestandteil von Entitäten auftreten.



(a) Vollständigkeit (b) Genauigkeit

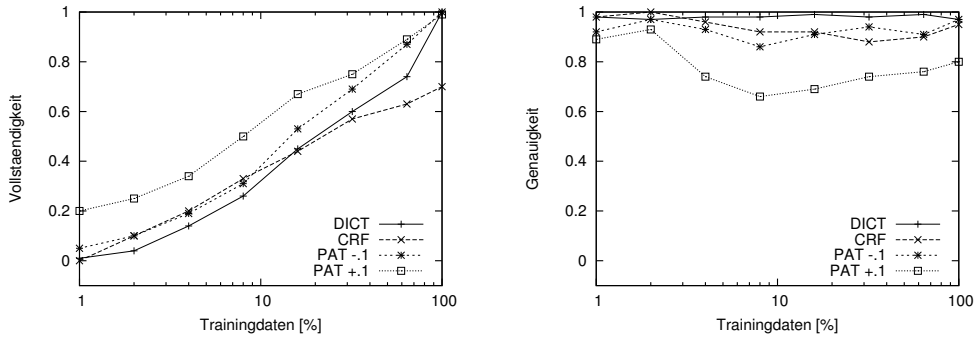
Abb. A.9: Extraktionsqualität für Lehrveranstaltungen (ReLIE-Korpus)

Der Grund für die bessere Genauigkeit von *PAT-.1* gegenüber *PAT+.1* ab einer Verwendung von mehr als 16% der Trainingsdaten, liegt hauptsächlich in der letzten Extraktionsregel, die in den Beispielen in Tab. A.7 dargestellt ist. Die Berücksichtigung von typischen Großbuchstaben, die am Beginn der Lehrveranstaltungsabkürzungen vorkommen, führt zu einer wesentlichen besseren Genauigkeit im Vergleich zu *PAT+.1*.

Tab. A.7: Beispiele für Reguläre Ausdrücke zur Extraktion von Lehrveranstaltungen (ReLIE-Korpus)

***PAT-.1* (3/3)**	***PAT+.1* (3/3)**
⟨UC|LC⟩$^+$(_)⟨NB⟩$^+$	⟨UC|LC⟩$^+$(_)⟨NB⟩$^+$
⟨UC|LC⟩$^+$⟨SC⟩(_)(271)	⟨UC|LC⟩$^+$⟨SC⟩(_)⟨NB⟩$^+$
(G|H|U|M|...|I)(E|U|C|A|...|M)⟨UC⟩{1,8}(_)⟨NB⟩$^+$	⟨UC⟩$^+$(_)⟨NB⟩$^+$

Bei der Extraktion von Softwarebezeichnungen aus dem „*ReLIE-Korpus*“ liefert *PAT+.1* eine weit höhere Vollständigkeit als die anderen Verfahren, zeigt jedoch Nachteile hinsichtlich der Genauigkeit. *PAT-.1* und *CRF* liefern vergleichbare Ergebnisse.



(a) Vollständigkeit (b) Genauigkeit

Abb. A.10: Extraktionsqualität für Softwarebezeichnungen (ReLIE-Korpus)

Die Beispiele für Reguläre Ausdrücke zur Extraktion von Softwarebezeichnungen sind in Tab. A.8 dargestellt. Sie stellen typische syntaktische Muster von Softwarebezeichnungen dar. Die erste Regel beschreibt Softwarebezeichner, die eine Zahl oder ein Jahr als Versionsnummer tragen (z. B. „*Windows 98*“, „*Fireworks 2004*“, „*Word 2007*“ oder „*Dreamweaver 4*“).

Tab. A.8: Beispiele für Reguläre Ausdrücke zur Extraktion von Softwarebezeichnungen (ReLIE-Korpus)

***PAT-.1* (3/11)**	***PAT+.1* (3/11)**
⟨UC|LC⟩$^+$(_)(⟨NB⟩{1,2}|(200)⟨NB⟩)	⟨UC|LC⟩$^+$(_)⟨NB⟩{1,4}
⟨UC|LC⟩$^+$(_)⟨UC|LC⟩$^+$(_)(6|3|4|5)(\.)⟨NB⟩	⟨UC|LC⟩$^+$(_)⟨UC|LC⟩$^+$(_)⟨NB⟩$^+$⟨SC⟩⟨NB⟩$^+$
⟨UC|LC⟩$^+$(_)(8|6|3|4)(\.)⟨NB⟩	⟨UC|LC⟩$^+$(_)⟨NB⟩$^+$⟨SC⟩⟨NB⟩$^+$

Der zweite und dritte Ausdruck stellt Softwarebezeichnungen mit Versionsnummern, die durch einen Punktsyntax in Haupt- und Unterversionen unterschieden werden, dar. Erstere der beiden Regeln wurde für Beispiele mit zwei Token und zweitere für Bezeichner mit einem Token erzeugt. *PAT+.1* liefert abstraktere Versionen dieser Regeln, die in dem hier dargestellten Anwendungsfall eine hohe Vollständigkeit mit akzeptablen Genauigkeiten liefert. Wenn die konkrete Anwendung, die die Extraktionsergebnisse konsumiert, eine hohe Genauigkeit voraussetzt, ist die Konfiguration *PAT-.1* besser geeignet.

Literaturverzeichnis

[AAB+08] AGRAWAL, Rakesh ; AILAMAKI, Anastasia ; BERNSTEIN, Philip A. ; BREWER, Eric A. ; CAREY, Michael J. ; CHAUDHURI, Surajit ; DOAN, AnHai ; FLORESCU, Daniela ; FRANKLIN, Michael J. ; GARCIA-MOLINA, Hector ; GEHRKE, Johannes ; GRUENWALD, Le ; HAAS, Laura M. ; HALEVY, Alon Y. ; HELLERSTEIN, Joseph M. ; IOANNIDIS, Yannis E. ; KORTH, Hank F. ; KOSSMANN, Donald ; MADDEN, Samuel ; MAGOULAS, Roger ; OOI, Beng C. ; O'REILLY, Tim ; RAMAKRISHNAN, Raghu ; SARAWAGI, Sunita ; STONEBRAKER, Michael ; SZALAY, Alexander S. ; WEIKUM, Gerhard: The Claremont Report on Database Research. In: *SIGMOD Record* 37 (2008), Nr. 3, S. 9–19

[AAD+09] ALEXOPOULOU, Dimitra ; ANDREOPOULOS, Bill ; DIETZE, Heiko ; DOMS, Andreas ; GANDON, Fabien L. ; HAKENBERG, Jörg ; KHELIF, Khaled ; SCHROEDER, Michael ; WÄCHTER, Thomas: Biomedical Word Sense Disambiguation with Ontologies and Metadata: Automation Meets Accuracy. In: *BMC Bioinformatics* 10 (2009)

[ABC+76] ASTRAHAN, M. M. ; BLASGEN, M. W. ; CHAMBERLIN, D. D. ; ESWARAN, K. P. ; GRAY, J. N. ; GRIFFITHS, P. P. ; KING, W. F. ; LORIE, R. A. ; MCJONES, P. R. ; MEHL, J. W. ; PUTZOLU, G. R. ; TRAIGER, I. L. ; WADE, B. W. ; WATSON, V.: System R: Relational Approach to Database Management. In: *ACM Transactions on Database Systems (TODS)* 1 (1976), Nr. 2, S. 97–137

[AC75] AHO, Alfred V. ; CORASICK, Margaret J.: Efficient String Matching: An Aid to Bibliographic Search. In: *Communications of the ACM* 18 (1975), Nr. 6, S. 333–340

[ACD02] AGRAWAL, Sanjay ; CHAUDHURI, Surajit ; DAS, Gautam: DBXplorer: A System for Keyword-Based Search over Relational Databases. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, IEEE Computer Society, 2002, S. 5–16

[AG00] AGICHTEIN, Eugene ; GRAVANO, Luis: Snowball: Extracting Relations from Large Plain-Text Collections. In: *ACM Conference on Digital Libraries (ACM DL)*, Association for Computing Machinery (ACM), 2000, S. 85–94

[Agi05] AGICHTEIN, Eugene: Scaling Information Extraction to Large Document Collections. In: *IEEE Data Engineering Bulletin* 28 (2005), Nr. 4, S. 3–10

[AHB+93] APPELT, Douglas E. ; HOBBS, Jerry R. ; BEAR, John ; ISRAEL, David J. ; TYSON, Mabry: FASTUS: A Finite-State Processor for Information Extraction from Real-world Text. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Morgan Kaufmann, 1993, S. 1172–1178

[AHS+09] AHMADI, Babak ; HADJIELEFTHERIOU, Marios ; SEIDL, Thomas ; SRIVASTAVA, Divesh ; VENKATASUBRAMANIAN, Suresh: Type-Based Categorization of Relational Attributes. In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*, Association for Computing Machinery (ACM), 2009, S. 84–95

[AHSS04] AMITAY, E. ; HAR'EL, N. ; SIVAN, R. ; SOFFER, A.: Web-a-where: Geotagging Web Content. In: *Proceeding of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2004, S. 273–280

[ALS09] AGIRRE, Eneko ; LACALLE, Oier L. ; SOROA, Aitor: Knowledge-Based WSD and Specific Domains: Performing Better than Generic Supervised WSD. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Morgan Kaufmann, 2009, S. 1501–1506

[Amd67] AMDAHL, Gene M.: Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In: *Proceedings of the Joint Computer Conference of the American Federation of Information Processing Societies (AFIPS)*, Association for Computing Machinery (ACM), 1967, S. 483–485

[AN88] AIKEN, A. ; NICOLAU, A.: Optimal Loop Parallelization. In: *ACM SIGPLAN Notices* 23 (1988), Nr. 7, S. 308–317

[Ang82] ANGLUIN, Dana: Inference of Reversible Languages. In: *Journal of the ACM* 29 (1982), Nr. 3, S. 741–765

[AR96] AGIRRE, Eneko ; RIGAU, German: Word Sense Disambiguation Using Conceptual Density. In: *Proceedings of the International Conference on Computational Linguistics (COLING)*, Morgan Kaufmann, 1996, S. 16–22

[AS92] AUSTIN, Todd M. ; SOHI, Gurindar S.: Dynamic Dependency Analysis of Ordinary Programs. In: *ACM SIGARCH Computer Architecture News* 20 (1992), Nr. 2, S. 342–351

[AV06] ADRIAANS, Pieter W. ; VITÁNYI, Paul M. B.: The Power and Perils of MDL. In: *The Computing Research Repository (CoRR)* abs/cs/0612095 (2006)

[BAR06] BALOG, Krisztian ; AZZOPARDI, Leif ; RIJKE, Maarten de: Formal Models for Expert Finding in Enterprise Corpora. In: *Proceeding of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Association for Computing Machinery (ACM), 2006, S. 43–50

[BBD+02] BABCOCK, Brian ; BABU, Shivnath ; DATAR, Mayur ; MOTWANI, Rajeev ; WIDOM, Jennifer: Models and Issues in Data Stream Systems. In: *Proceedings of the Symposium on Principles of Database Systems (PODS)*, Association for Computing Machinery (ACM), 2002, S. 1–16

[BBF+09] BRASCHLER, Martin ; BRINER, Norman ; FISCHER, Hans ; HÄNI, Markus ; MANDL, Thomas ; SCHÄUBLE, Peter ; STEINBACH, Markus ; STUKER, Jürg: Enterprise-Search-Systeme im internen Wissensmanagement: Ergebnisse einer Studie zu Perspektiven der Unternehmen. In: *Datenbank-Spektrum* 9 (2009), Nr. 30, S. 5–12

[BBH+09] BRAUER, Falk ; BARCZYNSKI, Wojciech ; HACKENBROICH, Gregor ; SCHRAMM, Marcus ; MOCAN, Adrian ; FÖRSTER, Felix: RankIE: Document Retrieval on Ranked Entity Graphs (Demo). In: *Proceedings of the VLDB Endowment* 2 (2009), Nr. 2, S. 1578–1581

[BBLM09] BARCZYNSKI, Wojciech ; BRAUER, Falk ; LÖSER, Alexander ; MOCAN, Adrian: Algebraic Information Extraction of Enterprise Data: Methodology and Operators. In: *Proceedings of the IJCAI International Workshop on Identity and Reference in Web-Based Knowledge Representation*, 2009

[BBM03] BUDINSKY, Frank ; BRODSKY, Stephen A. ; MERKS, Ed: *Eclipse Modeling Framework*. Pearson Education, 2003

[BBM09a] BARCZYNSKI, Wojciech ; BRAUER, Falk ; MOCAN, Adrian: ExplainIE - Explaining Information Extraction Systems (Poster). In: *Proceedings of the International Conference on Information Quality (ICIQ)*, HPI/MIT, 2009

[BBM+09b] BARCZYNSKI, Wojciech ; BRAUER, Falk ; MOCAN, Adrian ; SCHRAMM, Marcus ; FROEMBERG, Jan: BI Style Relation Discovery between Entities in Text. In: *Proceedings of the ICDE Workshop on New Trends in Information Integration (NTII)*, 2009

[BC04] BRODER, A. Z. ; CICCOLO, A. C.: Towards the Next Generation of Enterprise Search Technology. In: *IBM Systems Journal* 43 (2004), Nr. 3, S. 451–454

[BDS01] BORKAR, Vinayak R. ; DESHMUKH, Kaustubh ; SARAWAGI, Sunita: Automatic Segmentation of Text into Structured Records. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2001, S. 175–186

[BFBS10] BARCZYNSKI, Wojciech ; FOESTER, Felix ; BRAUER, Falk ; SCHUSTER, Daniel: AdaptIE – Using Domain Language Concept to Enable Domain Experts in Modeling of Information Extraction Plans. In: *Proceedings of the International Conference on Enterprise Information Systems (ICEIS)*, 2010

[BFG01] BAUMGARTNER, R. ; FLESCA, S. ; GOTTLOB, G.: Visual Web Information Extraction with Lixto. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2001, S. 119–128

[BFH+04] BUCK, Ian ; FOLEY, Tim ; HORN, Daniel R. ; SUGERMAN, Jeremy ; FATAHALIAN, Kayvon ; HOUSTON, Mike ; HANRAHAN, Pat: Brook for GPUs: Stream Computing on Graphics Hardware. In: *ACM Transactions on Graphics (TOG)* 23 (2004), Nr. 3, S. 777–786

[BGG09] BAUMGARTNER, Robert ; GATTERBAUER, Wolfgang ; GOTTLOB, Georg: Web Data Extraction System. In: *Encyclopedia of Database Systems*. 2009, S. 3465–3471

[BGNV08] BEX, Geert J. ; GELADE, Wouter ; NEVEN, Frank ; VANSUMMEREN, Stijn: Learning Deterministic Regular Expressions for the Inference of Schemas from XML Data. In: *Proceedings of the International World Wide Web Conference (WWW)*, Association for Computing Machinery (ACM), 2008, S. 825–834

[BH08] BERNSTEIN, Philip A. ; HAAS, Laura M.: Information Integration in the Enterprise. In: *Communications of the ACM* 51 (2008), Nr. 9, S. 72–79

[BHF09] BINNIG, Carsten ; HILDENBRAND, Stefan ; FÄRBER, Franz: Dictionary-Based Order-Preserving String Compression for Main Memory Column Stores. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2009, S. 283–296

[BHG87] BERNSTEIN, Philip A. ; HADZILACOS, Vassos ; GOODMAN, Nathan: *Concurrency Control and Recovery in Database Systems*. Reading, MA, USA : Addison-Wesley, 1987

[BHH+10] BRAUER, Falk ; HUBER, Michael ; HACKENBROICH, Gregor ; LESER, Ulf ; NAUMANN, Felix ; BARCZYNSKI, Wojciech: Graph-Based Concept Identification and Disambiguation for Enterprise Search. In: *Proceedings of the International World Wide Web Conference (WWW)*, Association for Computing Machinery (ACM), 2010, S. 171–180

[BHN+02] BHALOTIA, Gaurav ; HULGERI, Arvind ; NAKHE, Charuta ; CHAKRABARTI, Soumen ; SUDARSHAN, S.: Keyword Searching and Browsing in Databases Using BANKS. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, IEEE Computer Society, 2002, S. 431–440

[BHP04] BALMIN, Andrey ; HRISTIDIS, Vagelis ; PAPAKONSTANTINOU, Yannis: ObjectRank: Authority-Based Keyword Search in Databases. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, Morgan Kaufmann, 2004, S. 564–575

[BLD08] BRAUER, Falk ; LÖSER, Alexander ; DO, Hong-Hai: Mapping Enterprise Entities to Text Segments. In: *Proceedings of the Ph.D. Workshop in CIKM (PIKM)*, 2008, S. 85–88

[BM05] BEKKERMAN, Ron ; MCCALLUM, Andrew: Disambiguating Web Appearances of People in a Social Network. In: *Proceedings of the International World Wide Web Conference (WWW)*, Association for Computing Machinery (ACM), 2005, S. 463–470

[BN05] BLEIHOLDER, Jens ; NAUMANN, Felix: Declarative Data Fusion - Syntax, Semantics, and Implementation. In: *Advances in Databases and Information Systems (ADBIS)*, 2005, S. 58–73

[BN08] BLEIHOLDER, Jens ; NAUMANN, Felix: Data Fusion. In: *ACM Computing Surveys* 41 (2008), Nr. 1, S. 1–41

[BNST06] BEX, Geert J. ; NEVEN, Frank ; SCHWENTICK, Thomas ; TUYLS, Karl: Inference of Concise DTDs from XML Data. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, VLDB Endowment, 2006, S. 115–126

[BNSV10] BEX, Geert J. ; NEVEN, Frank ; SCHWENTICK, Thomas ; VANSUMMEREN, Stijn: Inference of Concise Regular Expressions and DTDs. In: *ACM Transactions on Database Systems (TODS)* 35 (2010), Nr. 2, S. 1–47

[BP06] BUNESCU, Razvan C. ; PASCA, Marius: Using Encyclopedic Knowledge for Named Entity Disambiguation. In: *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Association for Computational Linguistics (ACL), 2006

[Bra94] BRAZMA, Alvis: Efficient Algorithm for Learning Simple Regular Expressions from Noisy Examples. In: *Proceedings of the International Workshop on Analogical and Inductive Inference: Algorithmic Learning Theory*, Springer Verlag, 1994, S. 260–271

[BRBM09] BRAUER, Falk ; RIEGER, Robert ; BARCZYNSKI, Wojciech ; MOCAN, Adrian: Regular Language Inference for Domain-Specific Named Entity Recognition. In: *Proceedings of the IADIS International Conference WWW/Internet*, International Association for Development of the Information Society (IADIS), 2009

[Bri98] BRIN, Sergey: Extracting Patterns and Relations from the World Wide Web. In: *Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB)*, Association for Computing Machinery (ACM), 1998, S. 172–183

[Bro02] BRODER, Andrei: A Taxonomy of Web Search. In: *SIGIR Forum* 36 (2002), Nr. 2, S. 3–10

[BSB+08] BRAUER, Falk ; SCHRAMM, Marcus ; BARCZYNSKI, Wojciech ; LÖSER, Alexander ; DO, Hong-Hai: Robust Recognition of Complex Entities in Text Exploiting Enterprise Data and NLP-Techniques. In: *Proceedings of the International Conference on Digital Information Management (ICDIM)*, IEEE Computer Society, 2008, S. 551–558

[BTMC04] BONTCHEVA, Kalina ; TABLAN, Valentin ; MAYNARD, Diana ; CUNNINGHAM, Hamish: Evolving GATE to Meet new Challenges in Language Engineering. In: *Natural Language Engineering* 10 (2004), Nr. 3-4, S. 349–373

[BVCS07] BAILEY, Peter ; VRIES, Arjen P. ; CRASWELL, Nick ; SOBOROFF, Ian: Overview of the TREC 2007 Enterprise Track. In: *Proceedings of the Text REtrieval Conference (TREC)*, National Institute of Standards and Technology (NIST), 2007

[BW64] BACHMAN, C. W. ; WILLIAMS, S. B.: A General Purpose Programming System for Random Access Memories. In: *Proceedings of the Joint Computer Conference of the American Federation of Information Processing Societies (AFIPS)*, Association for Computing Machinery (ACM), 1964, S. 411–422

[Car04] CARPENTER, Bob: Phrasal Queries with LingPipe and Lucene: Ad Hoc Genomics Text Retrieval. In: *Proceedings of the Text REtrieval Conference (TREC)*, National Institute of Standards and Technology (NIST), 2004

[CBC+07] CHU, Eric ; BAID, Akanksha ; CHEN, Ting ; DOAN, AnHai ; NAUGHTON, Jeffrey F.: A Relational Approach to Incrementally Extracting and Querying Structure in Unstructured Data. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, VLDB Endowment, 2007, S. 1045–1056

[CBD99] CHAKRABARTI, Soumen ; BERG, Martin van d. ; DOM, Byron: Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. In: *Computer Networks* 31 (1999), Nr. 11-16, S. 1623–1640

[CBG99] CHAI, Joyce Y. ; BIERMANN, Alan W. ; GUINN, Curry I.: Two Dimensional Generalization in Information Extraction. In: *Proceedings of the National Conference on Artificial Intelligence*, American Association for Artificial Intelligence (AAAI), 1999, S. 431–438

[CCA+09] CONDIE, Tyson ; CONWAY, Neil ; ALVARO, Peter ; HELLERSTEIN, Joseph M. ; ELMELEEGY, Khaled ; SEARS, Russell: MapReduce Online / EECS Department, University of California, Berkeley. Version: Oct 2009. `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-136.html`. 2009 (UCB/EECS-2009-136). – Forschungsbericht

[CDYR08] CHEN, Fei ; DOAN, AnHai ; YANG, Jun ; RAMAKRISHNAN, Raghu: Efficient Information Extraction over Evolving Text Data. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, IEEE Computer Society, 2008, S. 943–952

[CE05] CAFARELLA, Michael J. ; ETZIONI, Oren: A Search Engine for Natural Language Applications. In: *Proceedings of the International World Wide Web Conference (WWW)*, Association for Computing Machinery (ACM), 2005, S. 442–452

[CGK06] CHAUDHURI, Surajit ; GANTI, Venkatesh ; KAUSHIK, Raghav: A Primitive Operator for Similarity Joins in Data Cleaning. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, IEEE Computer Society, 2006, S. 5

[CGRM06] CHAKARAVARTHY, V. T. ; GUPTA, H. ; ROY, P. ; MOHANIA, M.: Efficiently Linking Text Documents with Relevant Structured Information. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2006, S. 678

[CGX09] CHAUDHURI, Surajit ; GANTI, Venkatesh ; XIN, Dong: Mining Document Collections to Facilitate Accurate Approximate Entity Matching. In: *Proceedings of the VLDB Endowment* 2 (2009), Nr. 1, S. 395–406

[Che76] CHEN, Peter Pin-Shan: The Entity-Relationship Model—Toward a Unified View of Data. In: *ACM Transactions on Database Systems (TODS)* 1 (1976), Nr. 1, S. 9–36

[Cir01] CIRAVEGNA, Fabio: Adaptive Information Extraction from Text by Rule Induction and Generalisation. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Morgan Kaufmann, 2001, S. 1251–1256

[CK04] CHANG, Chia-Hui ; KUO, Shih-Chien: OLERA: Semisupervised Web-Data Extraction with Visual Support. In: *IEEE Intelligent Systems* 19 (2004), Nr. 6, S. 56–64

[CKGS06] CHANG, Chia-Hui ; KAYED, Mohammed ; GIRGIS, Moheb R. ; SHAALAN, Khaled F.: A Survey of Web Information Extraction Systems. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 18 (2006), Nr. 10, S. 1411–1428

[CKM07] CHEN, Z. ; KALASHNIKOV, D.V. ; MEHROTRA, S.: Adaptive Graphical Approach to Entity Resolution. In: *Proceedings of the ACM/IEEE-CS joint Conference on Digital Libraries*, 2007, S. 213

[CL01] CHANG, Chia-Hui ; LUI, Shao-Chen: IEPAD: Information Extraction Based on Pattern Discovery. In: *Proceedings of the International World Wide Web Conference (WWW)*, Association for Computing Machinery (ACM), 2001, S. 681–688

[CM99] CALIFF, Mary E. ; MOONEY, Raymond J.: Relational Learning of Pattern-Match Rules for Information Extraction. In: *Proceedings of the National Conference on Artificial Intelligence*, American Association for Artificial Intelligence (AAAI), 1999, S. 328–334

[CMBT02] CUNNINGHAM, D. H. ; MAYNARD, D. D. ; BONTCHEVA, D. K. ; TABLAN, M. V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: *Proceedings of the Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics (ACL), 2002

[CMT05] COHEN, William W. ; MINKOV, Einat ; TOMASIC, Anthony: Learning to Understand Web Site Update Requests. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, S. 1028–1033

[CNS06] CHANDEL, Amit ; NAGESH, P. C. ; SARAWAGI, Sunita: Efficient Batch Top-k Search for Dictionary-Based Entity Recognition. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, IEEE Computer Society, 2006, S. 28

[Con04] CONSORTIUM, Gene O.: The Gene Ontology (GO) Database and Informatics Resource. In: *Nucleic Acids Research* 1 (2004), Nr. 32, S. D258–D261

[CR02] CHO, Junghoo ; RAJAGOPALAN, Sridhar: A Fast Regular Expression Indexing Engine. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, IEEE Computer Society, 2002, S. 419–430

[CSC05] COUTO, Francisco M. ; SILVA, Mario J. ; COUTINHO, Pedro M.: Finding Genomic Ontology Terms in Text Using Evidence Content. In: *BMC Bioinformatics* 6 Supplement 1 (2005), S. S21

[CSL+06] COUTO, Francisco M. ; SILVA, Mário J. ; LEE, Vivian ; DIMMER, Emily ; CAMON, Evelyn ; APWEILER, Rolf ; KIRSCH, Harald ; REBHOLZ-SCHUHMANN, Dietrich: GOAnnotator: Linking Protein GO Annotations to Evidence Text. In: *Journal of Biomedical Discovery and Collaboration* 1 (2006), S. 19

[Cuc07] CUCERZAN, Silviu: Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing/ Conference on Computational Natural Language Learning (EMNLP-CoNLL)*, Association for Computational Linguistics (ACL), 2007, S. 708–716

[CVS05] CRASWELL, Nick ; VRIES, Arjen P. ; SOBOROFF, Ian: Overview of the TREC 2005 Enterprise Track. In: *Proceedings of the Text REtrieval Conference (TREC)*, National Institute of Standards and Technology (NIST), 2005

[DA09] DRURY, Brett ; ALMEIDA, J. J.: Construction of a Local Domain Ontology from News Stories. In: *Proceedings of the Portuguese Conference on Artificial Intelligence (EPIA)*, Springer Verlag, 2009, S. 400–410

[DAA+08] DIETZE, Heiko ; ALEXOPOULOU, Dimitra ; ALVERS, Michael R. ; BARRIO-ALVERS, Bill ; DOMS, Andreas ; HAKENBERG, Jörg ; MÖNNICH, Jan ; PLAKE, Conrad ; REISCHUK, Andreas ; ROYER, Loïc ; WÄCHTER, Thomas ; ZSCHUNKE, Matthias ; SCHROEDER, Michael: GoPubMed: Exploring Pubmed with Ontological Background Knowledge. In: *Ontologies and Text Mining for Life Sciences: Current Status and Future Perspectives* Bd. 08131, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2008 (Dagstuhl Seminar Proceedings)

[Dec06] DECLERCK, Thierry: SynAF: Towards a Standard for Syntactic Annotation. In: *Proceedings of the Language Resources and Evaluation Conference (LREC)*, 2006, S. 209–232

[DEG+03] DILL, Stephen ; EIRON, Nadav ; GIBSON, David ; GRUHL, Daniel ; GUHA, Ramanathan V. ; JHINGRAN, Anant ; KANUNGO, Tapas ; RAJAGOPALAN, Sridhar ; TOMKINS, Andrew ; TOMLIN, John A. ; ZIEN, Jason Y.: SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. In: *Proceedings of the International World Wide Web Conference (WWW)*, Association for Computing Machinery (ACM), 2003, S. 178–186

[DFJ+04] DING, Li ; FININ, Timothy W. ; JOSHI, Anupam ; PAN, Rong ; COST, R. S. ; PENG, Yun ; REDDIVARI, Pavan ; DOSHI, Vishal ; SACHS, Joel: Swoogle: a Search and Metadata Engine for the Semantic Web. In: *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, Association for Computing Machinery (ACM), 2004, S. 652–659

[DG92] DEWITT, David ; GRAY, Jim: Parallel Database Systems: the Future of High Performance Database Systems. In: *Communications of the ACM* 35 (1992), Nr. 6, S. 85–98

[DG04] DEAN, Jeffrey ; GHEMAWAT, Sanjay: MapReduce: Simplified Data Processing on Large Clusters. In: *Proceedings of the Symposium on Operating System Design and Implementation (OSDI)*, 2004, S. 137–150

[Die06] DIETZ, Jan L. G.: *Enterprise Ontology: Theory and Methodology*. Berlin – Heidelberg – New York : Springer Verlag, 2006

[DMSW09] DENEV, Dimitar ; MAZEIKA, Arturas ; SPANIOL, Marc ; WEIKUM, Gerhard: SHARC: Framework for Quality-Conscious Web Archiving, VLDB Endowment, 2009, S. 586–597

[DNR+08] DOAN, AnHai ; NAUGHTON, Jeffrey F. ; RAMAKRISHNAN, Raghu ; BAID, Akanksha ; CHAI, Xiaoyong ; 0002, Fei C. ; CHEN, Ting ; CHU, Eric ; DEROSE, Pedro ; GAO, Byron J. ; GOKHALE, Chaitanya ; HUANG, Jiansheng ; SHEN, Warren ; VUONG, Ba-Quy: Information Extraction Challenges in Managing Unstructured Sata. In: *SIGMOD Record* 37 (2008), Nr. 4, S. 14–20

[Dom99] DOMINGOS, Pedro: MetaCost: a General Method for Making Classifiers Cost-Sensitive. In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Association for Computing Machinery (ACM), 1999, S. 155–164

[DSC+07] DEROSE, Pedro ; SHEN, Warren ; CHEN, Fei ; LEE, Yoonkyong ; BURDICK, Douglas ; DOAN, AnHai ; RAMAKRISHNAN, Raghu: DBLife: A Community Information Management Platform for the Database Research Community (Demo). In: *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2007, S. 169–172

[EIV07] ELMAGARMID, A.K. ; IPEIROTIS, P.G. ; VERYKIOS, V.S.: Duplicate Record Detection: A Survey. In: *IEEE Data Engineering Bulletin* 19 (2007), Nr. 1, S. 1

[ELB07] EGNER, Michael T. ; LORCH, Markus ; BIDDLE, Edd: UIMA GRID: Distributed Large-Scale Text Analysis. In: *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, 2007, S. 317–326

[EMR00] ESCUDERO, Gerard ; MÀRQUEZ, Lluís ; RIGAU, German: A Comparison between Supervised Learning Algorithms for Word Sense Disambiguation. In: *The Computing Research Repository (CoRR)* cs.CL/0009022 (2000)

[Fav04] FAVRE, Jean-Marie: Foundations of Meta-Pyramids: Languages vs. Metamodels - Episode II: Story of Thotus the Baboon1. In: *Language Engineering for Model-Driven Software Development*, Dagstuhl Seminar Proceedings, 2004

[FB86] FU, K S. ; BOOTH, T L.: Grammatical Inference: Introduction and Survey. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 8 (1986), Nr. 3, S. 343–359

[Fer09] FERNAU, Henning: Algorithms for Learning Regular Expressions from Positive Data. In: *Information and Computation* 207 (2009), Nr. 4, S. 521–541

[FGM05] FINKEL, Jenny R. ; GRENAGER, Trond ; MANNING, Christopher D.: Incorporating Non-Local Information into Information Extraction Systems by Gibbs Sampling. In: *Proceedings of the Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics (ACL), 2005

[FHRW09] FARFÁN, Fernando ; HRISTIDIS, Vagelis ; RANGANATHAN, Anand ; WEINER, Michael: XOntoRank: Ontology-Aware Search of Electronic Medical Records. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, Association for Computing Machinery (ACM), 2009, S. 820–831

[FL04] FERRUCCI, D. ; LALLY, A.: UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. In: *Natural Language Engineering* 10 (2004), Nr. 3-4, S. 327–348

[Fre00] FREITAG, Dayne: Machine Learning for Information Extraction in Informal Domains. In: *Machine Learning* 39 (2000), Nr. 2-3, S. 169–202

[FZ07] FANG, Hui ; ZHAI, ChengXiang: Probabilistic Models for Expert Finding. In: *Proceedings of the European Conference on Information Retrieval Research (ECIR)*, Springer Verlag, 2007, S. 418–430

[GCG+04] GRUHL, D. ; CHAVET, L. ; GIBSON, D. ; MEYER, J. ; PATTANAYAK, P. ; TOMKINS, A. ; ZIEN, J.: How to Build a WebFountain: An Architecture for Very Large-Scale Text Analytics. In: *IBM Systems Journal* 43 (2004), Nr. 1, S. 64–77

[GGR+00] GAROFALAKIS, Minos N. ; GIONIS, Aristides ; RASTOGI, Rajeev ; SESHADRI, S. ; SHIM, Kyuseok: XTRACT: A System for Extracting Document Type Descriptors from XML Documents. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Association for Computing Machinery (ACM), 2000, S. 165–176

[GHY02] GRISHMAN, Ralph ; HUTTUNEN, Silja ; YANGARBER, Roman: Information Extraction for Enhanced Access to Disease Outbreak Reports. In: *Journal of Biomedical Informatics* 35 (2002), Nr. 4, S. 236–246

[GIJ+01] GRAVANO, Luis ; IPEIROTIS, Panagiotis G. ; JAGADISH, H. V. ; KOUDAS, Nick ; MUTHUKRISHNAN, S. ; SRIVASTAVA, Divesh: Approximate String Joins in a Database (Almost) for Free. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, Morgan Kaufmann, 2001, S. 491–500

[GM03] GALLEY, Michel ; MCKEOWN, Kathleen: Improving Word Sense Disambiguation in Lexical Chaining. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Morgan Kaufmann, 2003, S. 1486–1488

[Gol67] GOLD, E. M.: Language Identification in the Limit. In: *Information and Control* 10 (1967), Nr. 5, S. 447–474

[Grü04] GRÜNWALD, Peter: A Tutorial Introduction to the Minimum Description Length Principle. In: *The Computing Research Repository (CoRR)* math.ST/0406077 (2004)

[Gri97] GRISHMAN, Ralph: Information Extraction: Techniques and Challenges. In: *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, International Summer School* Bd. 1299, Springer Verlag, 1997 (Lecture Notes in Computer Science), S. 10–27

[GS96] GRISHMAN, Ralph ; SUNDHEIM, Beth: Message Understanding Conference- 6: A Brief History. In: *Proceedings of the International Conference on Computational Linguistics (COLING)*, 1996, S. 466–471

[GSBS03] GUO, Lin ; SHAO, Feng ; BOTEV, Chavdar ; SHANMUGASUNDARAM, Jayavel: XRANK: Ranked Keyword Search over XML Documents. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2003, S. 16–27

[GT07] GETOOR, Lise ; TASKAR, Ben: *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. MIT Press, 2007

[GTA06] GORDON, Michael I. ; THIES, William ; AMARASINGHE, Saman: Exploiting Coarse-Grained Task, Data, and Pipeline Parallelism in Stream Programs. In: *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2006, S. 151–162

[GYLRS08] GAUDAN, S. ; YEPES, A. J. ; LEE, V. ; REBHOLZ-SCHUHMANN, D.: Combining Evidence, Specificity, and Proximity Towards the Normalization of Gene Ontology Terms in Text. In: *EURASIP Journal on Bioinformatics and Systems Biology* 2008 (2008), S. 1–9

[Hah83] HAHN, W. von: *Fachkommunikation: Entwicklung, linguistische Konzepte, betriebliche Beispiele*. Walter de Gruyter, 1983

[HAMA06] HASSELL, J. ; ALEMAN-MEZA, B. ; ARPINAR, I.B.: Ontology-Driven Automatic Entity Disambiguation in Unstructured Text. In: *Lecture Notes in Computer Science* 4273 (2006), S. 44

[Haw04] HAWKING, David: Challenges in Enterprise Search. In: *Proceedings of the Australasian Database Conference (ADC)*, Australian Computer Society, Inc., 2004, S. 15–24

[HBL+08] HAHN, Udo ; BUYKO, Ekaterina ; LANDEFELD, Rico ; MÜHLHAUSEN, Matthias ; POPRAT, Michael ; TOMANEK, Katrin ; WERMTER, Joachim: An Overview of JCoRe, the JULIE Lab UIMA Component Repository. In: *Proceedings of the LREC'08 Workshop Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP*, 2008, S. 1–7

[HCK+04] HAN, Kyoung-Soo ; CHUNG, Hoo-Jung ; KIM, Sang-Bum ; SONG, Young-In ; LEE, Joo-Young ; RIM, Hae-Chang: Korea University Question Answering System at TREC 2004. In: *Proceedings of the Text REtrieval Conference (TREC)*, National Institute of Standards and Technology (NIST), 2004

[HFL+08] HE, Bingsheng ; FANG, Wenbin ; LUO, Qiong ; GOVINDARAJU, Naga K. ; WANG, Tuyong: Mars: a MapReduce Framework on Graphics Processors. In: *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Association for Computing Machinery (ACM), 2008, S. 260–269

[HNW06] HEGEWALD, Jan ; NAUMANN, Felix ; WEIS, Melanie: XStruct: Efficient Schema Extraction from Multiple and Large XML Documents. In: *Proceedings of the International ICDE Workshop on Schema and Data Management (XSDM)*, 2006, S. 81

[HP02] HRISTIDIS, Vagelis ; PAPAKONSTANTINOU, Yannis: DISCOVER: Keyword Search in Relational Databases. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, Morgan Kaufmann, 2002, S. 670–681

[HSL07] HAKENBERG, Jörg ; SCHROEDER, Michael ; LESER, Ulf: Consensus Pattern Alignment to Find Protein-Protein Interactions in Text. In: *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, 2007

[Huf95] HUFFMAN, Scott B.: Learning Information Extraction Patterns from Examples. In: *Learning for Natural Language Processing*, Springer Verlag, 1995, S. 246–260

[Hul00] HULL, D.A.: Xerox TREC-8 Question Answering Track Report. In: *Overview of the Ninth Text REtrieval Conference (TREC)* Bd. 21, 2000

[HZ09] HAN, Xianpei ; ZHAO, Jun: Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge. In: *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, Association for Computing Machinery (ACM), 2009, S. 215–224

[IBY+07] ISARD, Michael ; BUDIU, Mihai ; YU, Yuan ; BIRRELL, Andrew ; FETTERLY, Dennis: Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In: *SIGOPS Operating Systems Review* 41 (2007), Nr. 3, S. 59–72

[IN07] INMON, William ; NESAVICH, Anthony: *Tapping into Unstructured Data: Integrating Unstructured Data and Textual Analytics into Business Intelligence*. Upper Saddle River, NJ, USA : Prentice-Hall Press, 2007

[IR06] IDE, Nancy ; ROMARY, Laurent: Representing Linguistic Corpora and Their Annotations. In: *Proceedings of the Language Resources and Evaluation Conference (LREC)*, 2006

[IS07] IDE, Nancy ; SUDERMAN, Keith: GrAF: A Graph-Based Format for Linguistic Annotations. In: *Proceedings of the Linguistic Annotation Workshop (LAW)*, Association for Computational Linguistics (ACL), 2007, S. 1–8

[IV98] IDE, Nancy ; VERONIS, Jean: Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. In: *Computational Linguistics* 24 (1998), Nr. 1, S. 1–40

[Jav05] JAVED, Faizan: Inferring Context-Free Grammars for Domain-Specific Languages. In: *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Application*, Association for Computing Machinery (ACM), 2005, S. 242–243

[JHM04] JOHNSTON, Wesley M. ; HANNA, J. R. P. ; MILLAR, Richard J.: Advances in Dataflow Programming Languages. In: *ACM Computing Surveys* 36 (2004), Nr. 1, S. 1–34

[JKR+06] JAYRAM, T. S. ; KRISHNAMURTHY, Rajasekar ; RAGHAVAN, Sriram ; VAITHYANATHAN, Shivakumar ; ZHU, Huaiyu: Avatar Information Extraction System. In: *IEEE Data Engineering Bulletin* 29 (2006), Nr. 1, S. 40–48

[JPS05] JASTRZEBSKI, Lukasz ; PIASECKI, Maciej ; STRZELECKI, Grzegorz: Distributed Service - Oriented Architecture for Information Extraction System "Semanta". In: *Proceedings of the International Conference on Intelligent Systems Design and Applications (ISDA)*, 2005, S. 61–66

[JWR00] JONES, Karen S. ; WALKER, Steve ; ROBERTSON, Stephen E.: A Probabilistic Model of Information Retrieval: Development and Comparative Experiments - Part 2. In: *Information Processing and Management* 36 (2000), Nr. 6, S. 809–840

[KA96] KWOK, Yu-Kwong ; AHMAD, Ishfaq: Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors. In: *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 7 (1996), Nr. 5, S. 506–521

[KAP09] KLUEGL, Peter ; ATZMUELLER, Martin ; PUPPE, Frank: TextMarker: A Tool for Rule-Based Information Extraction. In: *Proceedings of the Biennial GSCL Conference, 2nd UIMA@GSCL Workshop*, Gunter Narr Verlag, 2009, S. 233–240

[Kir07] KIRK, David: NVIDIA Cuda Software and GPU Parallel Computing Architecture. In: *Proceedings of the International Symposium on Memory Management (ISMM)*, Association for Computing Machinery (ACM), 2007, S. 103–104

[KK03] KUHLINS, Stefan ; KORTHAUS, Axel: A Multithreaded Java Framework for Information Extraction in the Context of Enterprise Application Integration. In: *Proceedings of the International Symposium on Information and Communication Technologies (ISICT)*, Trinity College Dublin, 2003, S. 518–523

[KLR+08] KRISHNAMURTHY, Rajasekar ; LI, Yunyao ; RAGHAVAN, Sriram ; REISS, Frederick ; VAITHYANATHAN, Shivakumar ; ZHU, Huaiyu: SystemT: a System for Declarative Information Extraction. In: *SIGMOD Record* 37 (2008), Nr. 4, S. 7–13

[KM95] KIM, Jun-Tae ; MOLDOVAN, Dan I.: Acquisition of Linguistic Patterns for Knowledge-Based Information Extraction. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 7 (1995), Nr. 5, S. 713–724

[KM08] KUDLUR, Manjunath ; MAHLKE, Scott: Orchestrating the Execution of Stream Programs on Multicore Platforms. In: *ACM SIGPLAN Notices* 43 (2008), Nr. 6, S. 114–124

[KMU95] KILPELAINEN, Pekka ; MANNILA, Heikki ; UKKONEN, Esko: MDL Learning of Unions of Simple Pattern Languages from Positive Examples. In: *Proceedings of the Second European Conference on Computational Learning Theory (EuroCOLT)*, Springer Verlag, 1995, S. 252–260

[KNS+08] KANO, Yoshinobu ; NGUYEN, Ngan ; SÆTRE, Rune ; YOSHIDA, Kazuhiro ; MIYAO, Yusuke ; TSURUOKA, Yoshimasa ; MATSUBAYASHI, Yuichiro ; ANANIADOU, Sophia ; TSUJII, Junichi: Filling the Gaps between Tools and Users: a Tool Comparator, Using Protein-Protein Interaction as an Example. In: *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, 2008

[KNTM08] KALASHNIKOV, Dmitri V. ; NURAY-TURAN, Rabia ; MEHROTRA, Sharad: Towards Breaking the Quality Curse.: a Web-Querying Approach to Web People Search. In: *Proceeding of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Association for Computing Machinery (ACM), 2008, S. 27–34

[KPC+05] KACHOLIA, Varun ; PANDIT, Shashank ; CHAKRABARTI, Soumen ; SUDARSHAN, S. ; DESAI, Rushi ; KARAMBELKAR, Hrishikesh: Bidirectional Expansion For Keyword Search on Graph Databases. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, Association for Computing Machinery (ACM), 2005, S. 505–516

[KSI+08] KASNECI, Gjergji ; SUCHANEK, Fabian M. ; IFRIM, Georgiana ; ELBASSUONI, Shady ; RAMANATH, Maya ; WEIKUM, Gerhard: NAGA: Harvesting, Searching and Ranking Knowledge. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Association for Computing Machinery (ACM), 2008, S. 1285–1288

[LB10] LAI, Catherine ; BIRD, Steven: Querying Linguistic Trees. In: *Journal of Logic, Language and Information* 19 (2010), Nr. 1, S. 53–73

[LBB08] LÖSER, Alexander ; BARCZYNSKI, Wojciech ; BRAUER, Falk: What's the Intention Behind Your Query? A few Observations From a Large Developer Community. In: *Proceedings of the ESWC International Workshop on Identity and Reference on the Semantic Web (IRSW)*, 2008

[LCW07] LEITAO, L. ; CALADO, P. ; WEIS, M.: Structure-Based Inference of XML Similarity for Fuzzy Duplicate Detection. In: *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, 2007, S. 293–302

[Les86] LESK, Michael: Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In: *Proceedings of the International Conference on Systems Documentation*, Association for Computing Machinery (ACM), 1986, S. 24–26

[LFGP02] LAPRUN, Christophe ; FISCUS, Jonathan ; GAROFOLO, John ; PAJOT, Sylvain: Recent Improvements to the ATLAS Architecture. In: *Proceedings of the International Conference on Human Language Technology Research*, Morgan Kaufmann, 2002, S. 263–268

[LKR+08] LI, Yunyao ; KRISHNAMURTHY, Rajasekar ; RAGHAVAN, Sriram ; VAITHYANATHAN, Shivakumar ; JAGADISH, H. V.: Regular Expression Learning for Information Extraction. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2008, S. 21–30

[LLZ07] LUO, Yi ; LIN, Xuemin ; 0011, Wei W. ; ZHOU, Xiaofang: Spark: Top-k Keyword Query in Relational Databases. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Association for Computing Machinery (ACM), 2007, S. 115–126

[LM09] LUIS, Tiago ; MATOS, David M.: High-Performance High-Volume Layered Corpora Annotation. In: *Proceedings of the Linguistic Annotation Workshop (ACL-IJCNLP)*. Morristown, NJ, USA : Association for Computational Linguistics (ACL), 2009, S. 99–107

[LMP01] LAFFERTY, John D. ; MCCALLUM, Andrew ; PEREIRA, Fernando C. N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Proceedings of the International Conference on Machine Learning (ICML)*, Morgan Kaufmann, 2001, S. 282–289

[LN06] LESER, Ulf ; NAUMANN, Felix: *Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. dpunkt.verlag, Heidelberg, 2006

[LOF+08] LI, Guoliang ; OOI, Beng C. ; FENG, Jianhua ; WANG, Jianyong ; ZHOU, Lizhu: EASE: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-Structured and Structured Data. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Association for Computing Machinery (ACM), 2008, S. 903–914

[LSH08] LACLAVIK, Michal ; SELENG, Martin ; HLUCHÝ, Ladislav: Towards Large Scale Semantic Annotation Built on MapReduce Architecture. In: *Proceedings of the International Conference on Conceptual Structures (ICCS)*, 2008, S. 331–338

[Luh59] LUHN, H.P.: Auto-Encoding of Documents for Information Retrieval Systems. In: *Modern Trends in Documentation* (1959), S. 45–58

[LV97] LI, M. ; VITANYI, PMB: *An Introduction to Kolmogorov Complexity and its Applications*. Berlin – Heidelberg – New York : Springer Verlag, 1997

[LYMC06] LIU, Fang ; YU, Clement T. ; MENG, Weiyi ; CHOWDHURY, Abdur: Effective Keyword Search in Relational Databases. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Association for Computing Machinery (ACM), 2006, S. 563–574

[MBBL09] MUTHMANN, Klemens ; BARCZYNSKI, Wojciech ; BRAUER, Falk ; LÖSER, Alexander: Near-Duplicate Detection for Web-Forums. In: *Proceedings of the International Database Engineering and Applications Symposium (IDEAS)*, Association for Computing Machinery (ACM), 2009, S. 142–151

[MC03] MAYNARD, Diana ; CUNNINGHAM, Hamish: Multilingual Adaptations of ANNIE, a Reusable Information Extraction Tool. In: *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Association for Computational Linguistics (ACL), 2003, S. 219–222

[MCF03] MELLOR, Stephen J. ; CLARK, Anthony N. ; FUTAGAMI, Takao: Guest Editors' Introduction: Model-Driven Development. In: *IEEE Software* 20 (2003), Nr. 5, S. 14–18

[MCM03] MCCRUM, R. ; CRAN, W. ; MACNEIL, R.: *The Story of English*. Penguin Group USA, 2003

[MCN06] MINKOV, Einat ; COHEN, William W. ; NG, Andrew Y.: Contextual Search and Name Disambiguation in Email Using Graphs. In: *Proceeding of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, ACM, 2006, S. 27–34

[Mil95] MILLER, George A.: WordNet: A Lexical Database for English. In: *Communications of the ACM* 38 (1995), Nr. 11, S. 39–41

[MK07] MICHELSON, M. ; KNOBLOCK, C. A.: Unsupervised Information Extraction from Unstructured, Ungrammatical Data Sources on the World Wide Web. In: *International Journal on Document Analysis and Recognition* 10 (2007), Nr. 3, S. 211–226

[MKHV09] MICHELAKIS, Eirinaios ; KRISHNAMURTHY, Rajasekar ; HAAS, Peter J. ; VAITHYANATHAN, Shivakumar: Uncertainty Management in Rule-Based Information Extraction Systems. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Association for Computing Machinery (ACM), 2009, S. 101–114

[MKS04] MÜLLER, Hans-Michael M. ; KENNY, Eimear E. ; STERNBERG, Paul W.: Textpresso: An Ontology-Based Information Retrieval and Extraction System for Biological Literature. In: *PLoS Biology* 2 (2004), Nr. 11, S. 309

[MM02] MICHAILIDIS, P. D. ; MARGARITIS, K.G.: On-Line Approximate String Searching Algorithms: Survey and Experimental Results. In: *International Journal of Computer Mathematics* 79 (2002), Nr. 8, S. 867–888

[MM04] MUKHERJEE, Rajat ; MAO, Jianchang: Enterprise Search: Tough Stuff. In: *ACM Queue* 2 (2004), Nr. 2, S. 36–46

[MP94] MURPHY, Patrick M. ; PAZZANI, Michael J.: Exploring the Decision Forest: An Empirical Investigation of Occam's Razor in Decision Tree Induction. In: *Journal of Artificial Intelligence Research (JAIR)* 1 (1994), S. 257–275

[MRS08] MANNING, Christopher D. ; RAGHAVAN, Prabhakar ; SCHÜTZE, Hinrich: *Introduction to Information Retrieval*. 1. Cambridge University Press, 2008

[MS06] MANSURI, Imran R. ; SARAWAGI, Sunita: Integrating Unstructured Data into Relational Databases. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, IEEE Computer Society, 2006, S. 29

[MSM94] MARCUS, Mitchell P. ; SANTORINI, Beatrice ; MARCINKIEWICZ, Mary A.: Building a Large Annotated Corpus of English: The Penn Treebank. Cambridge, MA, USA : MIT Press, 1994, S. 313–330

[MSM06] MANGOLD, Christoph ; SCHWARZ, Holger ; MITSCHANG, Bernhard: u38: A Framework for Database-Supported Enterprise Document-Retrieval. In: *Proceedings of the International Database Engineering and Applications Symposium (IDEAS)*, IEEE Computer Society, 2006, S. 193–200

[MTT+98] MANDALA, Rila ; TOKUNAGA, Takenobu ; TANAKA, Hozumi ; OKUMURA, Akitoshi ; SATOH, Kenji: Ad Hoc Retrieval Experiments Using WordNet and Automatically Constructed Thesauri. In: *Proceedings of the Text REtrieval Conference (TREC)*, National Institute of Standards and Technology (NIST), 1998, S. 414–419

[Mud98] MUDDAMALLE, Manikya R.: Natural Language versus Controlled Vocabulary in Information Retrieval: a Case Study in Soil Mechanics. In: *Journal of the American Society for Information Science* 49 (1998), Nr. 10, S. 881–887

[Mus99] MUSLEA, Ion: Extraction Patterns for Information Extraction Tasks: A Survey. In: *Proceedings of the AAAI Workshop on Machine Learning for Information Extraction*, American Association for Artificial Intelligence (AAAI), 1999, S. 1–6

[Nad05] NADEAU, D.: Balie: Baseline Information Extraction / Technical Report, School of Information Technology and Engineering, University of Ottawa. 2005. – Forschungsbericht

[Nav01] NAVARRO, Gonzalo: A Guided Tour to Approximate String Matching. In: *ACM Computing Surveys* 33 (2001), Nr. 1, S. 31–88

[Nav09] NAVIGLI, Roberto: Word Sense Disambiguation: A Survey. In: *ACM Computing Surveys* 41 (2009), Nr. 2, S. 1–69

[NHT+02] NAUMANN, F. ; HO, C.T. ; TIAN, X. ; HAAS, L. ; MEGIDDO, N.: Attribute Classification Using Feature Analysis. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, 2002, S. 271–271

[NL07] NAVIGLI, Roberto ; LAPATA, Mirella: Graph Connectivity Measures for Unsupervised Word Sense Disambiguation. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Morgan Kaufmann, 2007, S. 1683–1688

[NL10] NAVIGLI, Roberto ; LAPATA, Mirella: An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (2010), S. 678–692

[NR04] NAVARRO, Gonzalo ; RAFFINOT, Mathieu: New Techniques for Regular Expression Searching. In: *Algorithmica* 41 (2004), Nr. 2, S. 89–116

[NS07] NADEAU, David ; SEKINE, Satoshi: A Survey of Named Entity Recognition and Classification. In: *Linguisticae Investigationes* 30 (2007), Nr. 1, S. 3–26

[NTL10] NGUYEN, Quang L. ; TIKK, Domonkos ; LESER, Ulf: Simple Tricks for Improving Pattern-Based Information Extraction from the Biomedical Literature. In: *Journal of Biomedical Semantics* 1 (2010), Nr. 1, S. 9

[NV03] NAVIGLI, R. ; VELARDI, P.: An Analysis of Ontology-Based Query Expansion Strategies. In: *Proceedings of the IJCAI Workshop on Adaptive Text Extraction and Mining*, 2003

[ORS+08] OLSTON, Christopher ; REED, Benjamin ; SRIVASTAVA, Utkarsh ; KUMAR, Ravi ; TOMKINS, Andrew: Pig Latin: a Not-so-Foreign Language for Data Processing. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Association for Computing Machinery (ACM), 2008, S. 1099–1110

[OVG+09] OWYANG, Jeremiah K. ; VANBOSKIRK, Shar ; GLASS, Sarah ; OVERBY, Christine S. ; YOUNG, G. O. ; POLANCO, Angie: *The Forrester Wave: Community Platforms, Q1 2009*. 2009

[Ped01] PEDERSEN, Ted: A Decision Tree of Bigrams is an Accurate Predictor of Word Sense. In: *Proceedings of the Conference on Natural Language Learning (NAACL)*, Association for Computational Linguistics (ACL), 2001, S. 1–8

[PH00] PAREKH, Rajesh ; HONAVAR, Vasant: Automata Induction, Grammar Inference, and Language Acquisition. In: *The Handbook of Natural Language Processing*. New York, Ny, USA : Marcel Dekker Inc., 2000, S. 727–764

[PHL05] PLAKE, Conrad ; HAKENBERG, Jörg ; LESER, Ulf: Optimizing Syntax Patterns for Discovering Protein-Protein Interactions. In: *Proceedings of the ACM Symposium on Applied Computing (SAC)*, 2005, S. 195–201

[Pit89] PITT, Leonard: Inductive Inference, DFAs, and Computational Complexity. In: *Proceedings of the International Workshop on Analogical and Inductive Inference (AII)*, Springer Verlag, 1989, S. 18–44

[PNLH09] PALAGA, Peter ; NGUYEN, Long ; LESER, Ulf ; HAKENBERG, Jörg: High-Performance Information Extraction with AliBaba. In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*, Association for Computing Machinery (ACM), 2009, S. 1140–1143

[PRÁ+02] PAN, Alberto ; RAPOSO, Juan ; ÁLVAREZ, Manuel ; MONTOTO, Paula ; ORJALES, Vicente ; HIDALGO, Justo ; ARDAO, Lucía ; MOLANO, Anastasio ; VIÑA, Ángel: The Denodo Data Integration Platform. In: *VLDB*, 2002, S. 986–989

[Pv08] PAJAS, Petr ; ŠTĚPÁNEK, Jan: Recent Advances in a Feature-rich Framework for Treebank Annotation. In: *Proceedings of the International Conference on Computational Linguistics (COLING)*, Association for Computational Linguistics (ACL), 2008, S. 673–680

[PWN06] PUHLMANN, Sven ; WEIS, Melanie ; NAUMANN, Felix: XML Duplicate Detection Using Sorted Neighborhoods. In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*, 2006, S. 773–791

[QHC06] QU, Yuzhong ; HU, Wei ; CHENG, Gong: Constructing Virtual Documents for Ontology Matching. In: *Proceedings of the International World Wide Web Conference (WWW)*, Association for Computing Machinery (ACM), 2006, S. 23–31

[RB01] RAHM, Erhard ; BERNSTEIN, Philip A.: A Survey of Approaches to Automatic Schema Matching. In: *VLDB Journal* 10 (2001), Nr. 4, S. 334–350

[RBJ06] RAMAKRISHNAN, Ganesh ; BALAKRISHNAN, Sreeram ; JOSHI, Sachindra: Entity Annotation Based on Inverse Index Operations. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics (ACL), 2006, S. 492–500

[RD00] RAHM, Erhard ; DO, Hong H.: Data Cleaning: Problems and Current Approaches. In: *IEEE Data Engineering Bulletin* 23 (2000), Nr. 4, S. 3–13

[RH01] RAMAN, Vijayshankar ; HELLERSTEIN, Joseph M.: Potter's Wheel: An Interactive Data Cleaning System. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, Morgan Kaufmann, 2001, S. 381–390

[Ril93] RILOFF, Ellen: Automatically Constructing a Dictionary for Information Extraction Tasks. In: *Proceedings of the National Conference on Artificial Intelligence*, American Association for Artificial Intelligence (AAAI), 1993, S. 811–816

[RRK+08] REISS, Frederick ; RAGHAVAN, Sriram ; KRISHNAMURTHY, Rajasekar ; ZHU, Huaiyu ; VAITHYANATHAN, Shivakumar: An Algebraic Approach to Rule-Based Information Extraction. In: *Proceedings of the International Conference on Data Engineering (ICDE)*, IEEE Computer Society, 2008, S. 933–942

[RRP+07] RANGER, Colby ; RAGHURAMAN, Ramanan ; PENMETSA, Arun ; BRADSKI, Gary R. ; KOZYRAKIS, Christos: Evaluating MapReduce for Multi-Core and Multiprocessor Systems. In: *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, 2007, S. 13–24

[RS95] RICHARDSON, R. ; SMEATON, A. F.: Using WordNet in a Knowledge-Based Approach to Information Retrieval. In: *Proceedings of the BCS-IRSG Colloquium, Crewe*, 1995

[RSA04] ROCHA, Cristiano ; SCHWABE, Daniel ; ARAGAO, Marcus P.: A Hybrid Approach for Searching in the Semantic Web. In: *Proceedings of the International World Wide Web Conference (WWW)*, Association for Computing Machinery (ACM), 2004, S. 374–383

[RW99] ROBERTSON, Stephen E. ; WALKER, Steve: Okapi/Keenbow at TREC-8. In: *Proceedings of the Text REtrieval Conference (TREC)*, National Institute of Standards and Technology (NIST), 1999

[Sak97] SAKAKIBARA, Yasubumi: Recent Advances of Grammatical Inference. In: *Theoretical Computer Science - Special Issue on Algorithmic Learning Theory* 185 (1997), Nr. 1, S. 15–45

[Sar08] SARAWAGI, Sunita: Information Extraction. In: *Foundations and Trends in Databases* 1 (2008), Nr. 3, S. 261–377

[SCDZ06] SHAN, Jiulong ; CHEN, Yurong ; DIAO, Qian ; ZHANG, Yimin: Parallel Information Extraction on Shared Memory Multi-Processor System, IEEE Computer Society, 2006, S. 311–318

[SDM+08] SHEN, W. ; DEROSE, P. ; MCCANN, R. ; DOAN, A.H. ; RAMAKRISHNAN, R.: Toward Best-Effort Information Extraction. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2008, S. 1031–1042

[SDNR07] SHEN, W. ; DOAN, A.H. ; NAUGHTON, J.F. ; RAMAKRISHNAN, R.: Declarative Information Extraction Using Datalog with Embedded Extraction Predicates. In: *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2007, S. 1033–1044

[SLH09] SCHIEMANN, Torsten ; LESER, Ulf ; HAKENBERG, Jörg: Word Sense Disambiguation in Biomedical Applications: A Machine Learning Approach. In: *Information Retrieval in Biomedicine: Natural Language Processing for Knowledge Integration*. Hershey, PA, USA : IGI Global, 2009

[Sod99] SODERLAND, Stephen: Learning Information Extraction Rules for Semi-Structured and Free Text. In: *Machine Learning* 34 (1999), Nr. 1-3, S. 233–272

[SOT03] STOKOE, Christopher ; OAKES, Michael P. ; TAIT, John: Word Sense Disambiguation in Information Retrieval Revisited. In: *Proceeding of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Association for Computing Machinery (ACM), 2003, S. 159–166

[SP95] SCHÜTZE, H. ; PEDERSEN, J. O.: Information Retrieval Based on Word Senses. In: *Proceedings of the Symposium on Document Analysis and Information Retrieval*, 1995, S. 161–175

[SP03] SHA, Fei ; PEREIRA, Fernando C. N.: Shallow Parsing with Conditional Random Fields. In: *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, Association for Computational Linguistics (ACL), 2003, S. 134–141

[Sue09] SUEL, Torsten: Geo-Targeted Web Search. In: *Encyclopedia of Database Systems*. Berlin – Heidelberg – New York : Springer Verlag, 2009, S. 1251–1255

[SVC06] SOBOROFF, Ian ; VRIES, Arjen P. ; CRASWELL, Nick: Overview of the TREC 2006 Enterprise Track. In: *Proceedings of the Text REtrieval Conference (TREC)*, National Institute of Standards and Technology (NIST), 2006

[TAC06] TURMO, Jordi ; AGENO, Alicia ; CATALÀ, Neus: Adaptive Information Extraction. In: *ACM Computing Surveys* 38 (2006), Nr. 2, S. 4

[TBBA10] THOLLOT, Raphael ; BRAUER, Falk ; BARCZYNSKI, Wojciech ; AUFAURE, Marie-Aude: Text-to-Query: Dynamically Building Structured Analytics to Illustrate Textual Content. In: *Proceedings of the EDBT International Workshop on Business IntelligencE and the WEB (BEWEB)*, Association for Computing Machinery (ACM), 2010

[TKA02] THIES, William ; KARCZMAREK, Michal ; AMARASINGHE, Saman P.: StreamIt: A Language for Streaming Applications. In: *Proceedings of the International Conference on Compiler Construction (CC)*, Springer Verlag, 2002, S. 179–196

[TKMS03] TOUTANOVA, Kristina ; KLEIN, Dan ; MANNING, Christopher D. ; SINGER, Yoram: Feature-rich Part-of-Speech Tagging with a Cyclic Dependency Network. In: *Proceedings of the Conference on Natural Language Learning (NAACL)*, Association for Computational Linguistics (ACL), 2003, S. 173–180

[TKSDM03] TJONG KIM SANG, Erik F. ; DE MEULDER, Fien: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In: *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, Association for Computational Linguistics (ACL), 2003, S. 142–147

[TOKG03] TROUSSOV, Alexander ; O'DONOVAN, Brian ; KOSKENNIEMI, Seppo ; GLUSHNEV, Nikolay: Per-Node Optimization of Finite-State Mechanisms for Natural Language Processing. In: *Proceedings of the International Conference Computational Linguistics and Intelligent Text Processing (CICLing)*, 2003, S. 221–224

[TT04] TSURUOKA, Yoshimasa ; TSUJII, Jun ichi: Improving the Performance of Dictionary-Based Approaches in Protein Name Recognition. In: *Journal of Biomedical Informatics* 37 (2004), Nr. 6, S. 461–470

[TUD+09] TOUPIKOV, N. ; UMBRICH, J. ; DELBRU, R. ; HAUSENBLAS, M. ; TUMMARELLO, G.: DING! Dataset Ranking Using Formal Descriptions. In: *Proceedings of the WWW Workshop on Linked Data on the Web (LDOW2009)*, 2009

[TVA07] TSATSARONIS, George ; VAZIRGIANNIS, Michalis ; ANDROUTSOPOULOS, Ion: Word Sense Disambiguation with Spreading Activation Networks Generated from Thesauri. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Morgan Kaufmann, 2007, S. 1725–1730

[VFC05] VALLET, David ; FERNÁNDEZ, Miriam ; CASTELLS, Pablo: An Ontology-Based Information Retrieval Model. In: *Proceedings of the European Semantic Web Conference (ESWC)*, Springer Verlag, 2005, S. 455–470

[Vin06] VINOSKI, S.: Advanced Message Queuing Protocol. In: *IEEE Internet Computing* 10 (2006), Nr. 6, S. 87–89

[Voo93] VOORHEES, Ellen M.: Using WordNet to Disambiguate Word Senses for Text Retrieval. In: *Proceeding of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Association for Computing Machinery (ACM), 1993, S. 171–180

[VSS02] VASSILIADIS, Panos ; SIMITSIS, Alkis ; SKIADOPOULOS, Spiros: Conceptual Modeling for ETL Processes. In: *Proceedings of the ACM International Workshop on Data Warehousing and OLAP*, 2002, S. 14–21

[Whi08] WHITE, Bebo: Amy Langville and Carl Meyer, Google's Page Rank and Beyond: The Science of Search Engine Rankings. In: *Information Retrieval* 11 (2008), Nr. 5, S. 471–472

[WVV+01] WACHE, H. ; VÖGELE, T. ; VISSER, U. ; STUCKENSCHMIDT, H. ; SCHUSTER, G. ; NEUMANN, H. ; HÜBNER, S.: Ontology-Based Integration of Information - A Survey of Existing Approaches. In: *Proceedings of the IJCAI Workshop on Ontologies and Information Sharing*, 2001, S. 108–117

[WXLZ09] WANG, Wei ; XIAO, Chuan ; LIN, Xuemin ; ZHANG, Chengqi: Efficient Approximate Entity Extraction with Edit Distance Constraints. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Association for Computing Machinery (ACM), 2009, S. 759–770

[XP08] XU, Yu ; PAPAKONSTANTINOU, Yannis: Efficient LCA Based Keyword Search in XML Data. In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*, 2008, S. 535–546

[Yan03] YANGARBER, Roman: Counter-Training in Discovery of Semantic Patterns. In: *Proceedings of the Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics (ACL), 2003, S. 343–350

[YQC10] YU, J.X. ; QIN, L. ; CHANG, L.: Keyword Search in Relational Databases: A Survey. In: *IEEE Data Engineering Bulletin* 33 (2010), Nr. 1, S. 67–78

[YZHL05] YU, Hao ; ZHU, Xiaoyan ; HUANG, Minlie ; LI, Ming: Discovering Patterns to Extract Protein-Protein Interactions from the Literature: Part II. In: *Bioinformatics* 21 (2005), Nr. 15, S. 3294–3300

[ZD10] ZHOU, Jin ; DEMSKY, Brian: Bamboo: a Data-Centric, Object-Oriented Approach to Many-Core Software. In: *ACM SIGPLAN Notices* 45 (2010), Nr. 6, S. 388–399

[ZE01] ZADROZNY, Bianca ; ELKAN, Charles: Learning and Making Decisions when Costs and Probabilities Are Both Unknown. In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Association for Computing Machinery (ACM), 2001, S. 204–213

[ZL04] ZHAI, Chengxiang ; LAFFERTY, John: A Study of Smoothing Methods for Language Models Applied to Information Retrieval. In: *ACM Transactions on Information Systems (TOIS)* 22 (2004), Nr. 2, S. 179–214

[ZSWG09] ZHENG, Shuyi ; SONG, Ruihua ; WEN, Ji-Rong ; GILES, C. L.: Efficient Record-Level Wrapper Induction. In: *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, Association for Computing Machinery (ACM), 2009, S. 47–56

[ZWX+07] ZHOU, Qi ; WANG, Chong ; XIONG, Miao ; WANG, Haofen ; YU, Yong: SPARK: Adapting Keyword Query to Semantic Search. In: *Proceedings of the International Semantic Web Conference (ISWC)*, 2007, S. 694–707

Abbildungsverzeichnis

Tabellenverzeichnis

Beispielverzeichnis

Algorithmenverzeichnis

Selbständigkeitserklärung

Hiermit versichere ich, die vorliegende Dissertation eigenständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel, angefertigt zu haben. Alle öffentlichen Quellen sind als solche kenntlich gemacht. Die vorliegende Arbeit ist in dieser oder anderer Form zuvor nicht als Prüfungsarbeit zur Begutachtung vorgelegt worden.

Dresden, den 13.10.2010