Automatische Mapping-Verarbeitung auf Webdaten

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr. rer. nat.)

im Fachgebiet Informatik

vorgelegt von

Diplom-Informatiker Andreas Thor
geboren am 22. Februar 1979 in Bad Muskau

Die Annahme der Dissertation haben empfohlen:

1. Prof. Dr. Felix Naumann (Hasso-Plattner-Institut Potsdam)
2. Prof. Dr. Erhard Rahm (Universität Leipzig)
3. Prof. Dr. Gerhard Weikum (Max-Planck-Institut Saarbrücken)

Die Verleihung des akademischen Grades erfolgt auf Beschluss des Rates der Fakultät für Mathematik und Informatik vom 16.06.2008 mit dem Gesamtprädikat summa cum laude.

Danksagung

Mein größter Dank gilt meinem Betreuer Herrn Prof. Dr. Erhard Rahm, der mir in den vergangenen Jahren engagiert auf meinem wissenschaftlichen Weg mit Rat und Tat zur Seite stand. In zahllosen Gesprächen diskutierte er mit mir meine Ideen und Ergebnisse und trug so wesentlich zum Erfolg dieser Arbeit bei. Zusätzlich bewies er stets große Geduld mit mir, wenn er meine Entwürfe zu wissenschaftlichen Publikationen mit seinen Hinweisen und Anmerkungen verbesserte.

Diese Arbeit wäre ohne die finanzielle Unterstützung der Deutschen Forschungsgemeinschaft (DFG) nicht möglich gewesen. Dazu möchte ich mich stellvertretend bei Herrn Prof. Dr. Gerhard Brewka bedanken, der mir in seiner Funktion als Sprecher des Graduiertenkollegs „Wissensrepräsentation“ die Möglichkeit eröffnete, einen wesentlichen Teil meiner Arbeit als Stipendiat zu realisieren.

Des Weiteren möchte ich allen Kollegen der Abteilung Datenbanken am Institut für Informatik der Universität Leipzig für das sehr gute Arbeitsklima danken. In ganz besonders schöner Erinnerung wird mir unser jährliches Oberseminar an der Ostsee in Zingst bleiben.

Meinen Freunden danke ich für die große Unterstützung während meiner gesamten Promotionszeit, vor allem wenn sie mich durch aufmunternde Worte motivierten. Zu großem Dank bin ich auch meiner Schulfreundin Anja Munske verpflichtet, die viel Zeit und Energie in das Korrekturlesen dieser Arbeit gesteckt hat.

Besonderer Dank gilt natürlich meinen Eltern Ursula und Hartmut Thor, die mir stets alle Freiheiten auf meinem bisherigen Weg gelassen haben. Abschließend möchte ich mich bei meiner Frau Doreen Thor bedanken – vor allem für ihr Verständnis, dass ich viele Abende vor dem Rechner verbrachte, um diese Arbeit anzufertigen.

Leipzig, den 9. November 2007 Andreas Thor

Zusammenfassung

Das World Wide Web stellt eine Vielzahl von Informationen zur Verfügung. Neben dem Inhalt der Webseiten stellt ihre Verknüpfung durch Verweise zwischen ihnen eine wichtige Informationsquelle dar. So unterstützen z. B. Webseitenempfehlungen auf interessante andere Produkte Kunden großer E-Commerce-Websites bei der Suche nach dem passenden Produkt. Zusätzlich finden sich auf Seiten verschiedener Websites auch Informationen zum gleichen Objekt der realen Welt, so dass durch Links zwischen ihnen alle verfügbaren Informationen aus mehreren Datenquellen für entsprechende Analysen, z. B. einem Preisvergleich, erreichbar werden. In dieser Arbeit werden Verweise als sogenannte *Mappings* zusammengefasst, wobei ein Mapping eine Menge paarweiser Zuordnungen, sogenannter *Korrespondenzen*, zwischen Instanzdaten (Objekten) repräsentiert. Dabei wird mit dem Begriff Korrespondenz nicht nur die Verknüpfung gleicher Objekte gemeint, sondern allgemein eine semantische Beziehung zwischen zwei Objekten ausgedrückt. Innerhalb dieser Dissertation steht die Verarbeitung solcher Mappings, d. h. ihre Erzeugung, Optimierung und Verwendung, im Mittelpunkt.

Innerhalb einer Website können Korrespondenzen z. B. als Webseitenempfehlungen, sogenannte *Recommendations*, verwendet werden. Ausgehend von den vielen in der Literatur vorgeschlagenen Algorithmen zur Berechnung von Recommendations, sogenannte *Recommender*, stellt sich das Problem, welcher Recommender für welchen Nutzer unter welchen Umständen am nützlichsten ist. Durch eine gezielte und optimierte Auswahl sollen dabei die besten Recommendations bestimmt werden. Mit *AWESOME* wird dazu ein Ansatz vorgestellt, der eine solche adaptive Bestimmung von Recommendations zulässt. Durch die Aufzeichnung und Auswertung von Nutzer-Feedback können die Empfehlungen den Nutzerinteressen angepasst werden, so dass eine Steigerung der Recommendation-Qualität ermöglicht wird. Innerhalb der Arbeit werden insbesondere automatische Verfahren zur Verarbeitung des Nutzer-Feedbacks vorgestellt, so dass eine automatische Selbstoptimierung der Recommendations erzielt werden kann. Der AWESOME-Ansatz wurde innerhalb einer Website prototypisch implementiert und die Recommendation-Qualität bzgl. verschiedener Kriterien evaluiert. Dabei konnte insbesondere gezeigt werden, dass die automatische Adaption ähnliche Ergebnisse erzielt wie eine aufwändige, manuelle Optimierung der Recommendations.

Ein weiterer Schwerpunkt der Arbeit liegt in der Verarbeitung von Mappings zur Datenintegration. Dazu wird im zweiten Teil der Arbeit der Datenintegrationsansatz *iFuice* präsentiert, der – im Gegensatz zu schemabasierten Datenintegrationsansätzen – auf instanzbasierten Mappings zwischen Datenquellen aufbaut. Datenquellen und Mappings werden dabei mit Hilfe eines Domänenmodells semantisch annotiert, das u. a. die Typen der Objektinstanzen sowie die Art der durch Mappings definierten Beziehungen charakterisiert. Im Gegensatz zu den meist anfragebasierten Datenintegrationsansätzen eröffnet iFuice dem Nutzer die Möglichkeit, mittels Skripten ausführbare Datenintegrationsprozesse zu definieren. Innerhalb der Skripte kommen Operatoren zum Einsatz, die Objektinstanzen und Mappings in einer generischen Art und Weise verarbeiten. Dadurch können sowohl neue Mappings generiert als auch bereits bestehende durch entsprechende Kombination effektiv wieder verwendet werden. Ein weiterer Aspekt von iFuice ist die Möglichkeit der Informationsfusion, bei der als gleich erkannte Objektinstanzen zu sogenannten aggregierten Objekten zusammengefasst werden können. Die Verwendung von iFuice zur Informationsfusion wird insbesondere am Beispiel einer Zitierungsanalyse wissenschaftlicher Publikationen detailliert vorgestellt.

Abschließend wird im letzten Teil der Arbeit das *MOMA*-Framework präsentiert, das auf den wichtigen Aspekt des Object Matchings, d. h. dem Erkennen von Abbildern der gleichen Objekte der realen Welt in (verschiedenen) Datenquellen, abstellt. Das MOMA-Framework setzt auf dem iFuice-Ansatz auf und verwendet insbesondere dessen Operatoren und Datenstrukturen. MOMA unterstützt die Erstellung sogenannter Match-Workflows, die u. a. verschiedene Match-Verfahren, z. B. durch Berechnung syntaktischer Ähnlichkeiten von Attributwerten, integrieren können. Das Ergebnis eines Match-Workflows ist jeweils ein Mapping, das wiederum zur Informationsfusion innerhalb von iFuice genutzt werden kann. Wichtiges Kennzeichen des MOMA-Frameworks ist die Möglichkeit, existierende Mappings miteinander kombinieren und dadurch neue Mappings ableiten zu können. Mapping-Kombinationen können als sogenannte Match-Strategien definiert werden, welche flexibel für verschiedene Datenquellen eingesetzt werden können. Mit Hilfe einer prototypischen Implementation wurden verschiedene Match-Strategien unter Verwendung realer Datenquellen aus dem Bereich bibliografischer Daten evaluiert.

Inhaltsverzeichnis

Teil I

Einleitung

1 Einleitung

1.1 Einführung

Eine aktuelle Netcraft-Studie[1] besagt, dass es mehr als 60 Millionen aktive Websites im World Wide Web gibt (Stand: Oktober 2007) – mit stark steigender Tendenz. Zusätzlich erhöht sich auch die Menge der Informationen innerhalb der einzelnen Websites stetig. So entstehen z. B. monatlich ca. 10.000 neue Publikationseinträge auf der Website der DBLP Bibliography [197]. Diese rasante Vergrößerung des Webs rückt das Management von Webdaten in den Fokus der aktuellen Forschung, damit Nutzer mit den großen Datenmengen umgehen können. Die Bandbreite reicht dabei von Suchmaschinen, um mittels Stichworten nach relevanten Seiten zu suchen [24], über die automatische Klassifikation von Webdokumenten in Hierarchien [53], die Erstellung von Informationsportalen [179] bis hin zur dynamischen Kopplung verschiedener Web-Services innerhalb sogenannter Mashups [121].

Ein wichtiger Aspekt der Datenquellen sind ihre Verknüpfungen untereinander mittels Verweisen, welche Daten in der gleichen oder einer anderen Datenquelle referenzieren. Verweise bieten damit eine gezielte Möglichkeit, schnell zu weiteren relevanten Informationen zu gelangen und können z. B. als Hyperlinks innerhalb von HTML-Seiten oder durch RDF-Tripel [44], welche eine Grundlage des Semantic Webs [13] bilden, repräsentiert sein. Sie können sowohl von (menschlichen) Nutzern[2] verwen-

[1] http://news.netcraft.com/archives/2007/10/11/october_2007_web_server_survey.html

[2] In dieser Arbeit gelten grammatikalisch maskuline Personenbezeichnungen gleichermaßen für Personen weiblichen und männlichen Geschlechts.

det werden, welche z. B. den Hyperlinks innerhalb einer Webseite folgen und sich die referenzierten Seiten betrachten, als auch von Anwendungen genutzt werden, die automatisch die referenzierten Informationen verarbeiten, um sie z. B. zusammenzufassen.

In dieser Arbeit werden Verweise als sogenannte *Mappings*[3] zusammengefasst. Ein Mapping ist eine Menge paarweiser Zuordnungen, sogenannter *Korrespondenzen*, zwischen Instanzdaten. Dabei wird in dieser Arbeit mit dem Begriff Korrespondenz nicht nur die Verknüpfung gleicher Objekte gemeint, sondern allgemein eine Beziehung zwischen zwei Objekten ausgedrückt. Objekte können dabei einzelne Webseiten sein, aber auch andere Entitäten, z. B. Datensätze einer Datenbank oder Suchergebnisse von Suchmaschinen. Aus einer „Berechnungssicht“ lässt sich ein Mapping als eine Funktion charakterisieren, die für ein oder mehrere Eingabeobjekte alle *korrespondierenden* Ausgabeobjekte ermittelt. Bei der „Ausführung“ des Mappings können unterschiedliche Verfahren angewendet werden, z. B. SQL-Anfragen an Datenbanken, Informationsextraktionstechniken bei Webseiten, um z. B. bestehende Hyperlinks zu extrahieren, oder Web-Service-Aufrufe. Aus einer „Datensicht“ lässt sich ein Mapping als zweispaltige Tabelle darstellen, wobei jede Zeile die beiden korrespondierenden Objekte definiert.

Die manuelle Erstellung solcher instanzbasierten Korrespondenzen ist sehr zeitaufwändig und skaliert schlecht mit der Anzahl und Größe der Datenquellen. Als eine Möglichkeit haben sich dabei kollaborative Ansätze entwickelt, in der eine Nutzergemeinde gemeinschaftlich eine Datenbasis erstellt und verändert (z. B. Wikis, Web-Content-Management-Systeme). Korrespondenzen können dabei explizit definiert werden, z. B. in Form von Hyperlinks innerhalb eines Wikis, oder implizit durch die Verwendung gleicher Begriffe bei der Annotierung von Objekten (z. B. Fotos bei Flickr[4]). In dieser Arbeit liegt der Fokus jedoch auf der automatischen Verarbeitung solcher Korrespondenzen, was die Erstellung neuer sowie die Verwendung bestehender Mappings umfasst.

Innerhalb dieser Arbeit werden drei Aspekte der automatischen Mapping-Verarbeitung betrachtet, die sich u. a. aus den unterschiedlichen Interessen und Kenntnissen von Nutzern ableiten lassen. Ein Website-Besucher möchte z. B. auf jeder Seite Links auf für ihn relevante andere Seiten (der gleichen Website) erhalten. Ziel ist daher das automatische, dynamische Generieren und Auswählen der „besten“ Links, d. h. der besten Korrespondenzen für die aktuelle Seite. Ein „fortgeschrittener“ Nutzer möchte die vorhandenen Korrespondenzen nutzen, um gezielt die dadurch verknüpften Informationen in den verschiedenen Datenquellen „zusammenzusetzen“. Sein Ziel ist u. a. die Beantwortung von Fragen, die nur unter Einbeziehung mehrerer Datenquellen beantwortet werden können, z. B. welche die am häufigsten zitierten Publikationen der letzten 10 Jahre im Datenbankbereich sind. Schließlich möchte ein „Datenspezialist“

[3] *engl.* mapping [comp.] = Abbildung, Zuordnung
[4] http://www.flickr.com/

neue Verknüpfungen effizient erstellen. Insbesondere geht es um die Identifikation gleicher Objekte (der realen Welt) in der gleichen oder verschiedenen Datenquellen, so dass durch eine Korrespondenz zwischen ihnen alle in den einzelnen Datenquellen vorliegenden Informationen zum gleichen Realweltobjekt verfügbar gemacht werden. Aus den drei geschilderten Nutzeranforderungen ergeben sich die folgenden, in dieser Arbeit behandelten, Themen.

Webseitenempfehlungen: Produktempfehlungen sind ein wichtiger Faktor für den Erfolg großer E-Commerce-Websites. Die große Fülle der angebotenen Produkte erschwert die Navigation für die Website-Nutzer. Hierarchische Einordnungen der Produkte oder entsprechende Suchmaschinen können den Nutzern insbesondere dann nur begrenzt helfen, wenn der Nutzer nicht genau weiß, was er sucht, sondern vielmehr auf der Suche nach interessanten Produkten ist. Entsprechende Empfehlungen, sogenannte *Recommendations*, ermöglichen es, dem Nutzer Vorschläge für potenziell interessante Seiten bzw. Produkte zu geben. Eine manuelle Bestimmung von Recommendations ist bei großen Websites mit einem hohen Aufwand verbunden. Daher wurden in der Literatur viele verschiedene Verfahren – sogenannte *Recommender* – zur automatischen Bestimmung von Recommendations vorgestellt, welche sowohl unterschiedliche Informationen zur Berechnung heranziehen als auch unterschiedliche Algorithmen anwenden. Ein bekanntes Beispiel ist „Kunden, die dieses Produkt gekauft haben, haben auch ... gekauft“, welches auf der Website des Online-Händlers Amazon[5] zu finden ist. Andere Verfahren empfehlen z. B. neue Produkte, zu einem Produkt ähnliche Produkte oder die am meisten verkauften Produkte der letzten Tage. Ziel ist es, dem Nutzer möglichst immer die „beste“ Recommendation zu präsentieren, so dass ein möglichst großer Umsatz realisiert wird. Dazu muss aus der Fülle möglicher Recommendations, die von den einzelnen Recommendern berechnet werden, die „wahrscheinlich beste“ Recommendation ausgewählt werden.

Informationsfusion: Ein mögliches Szenario für die Mapping-Verarbeitung zur dynamischen Informationsfusion sei innerhalb des Bereichs bibliografischer Daten dargestellt. Ein Nutzer möchte z. B. über die in seinem Adressbuch gespeicherten Kollegen erfahren, welche neuen Publikationen sie in letzter Zeit veröffentlicht haben. Dazu müsste er für jeden Namen den passenden Autoreneintrag bei einer entsprechenden Datenquelle, z. B. DBLP Bibliography [2], finden und aus der Liste der zugehörigen Publikationen die neuesten extrahieren. Neben diesem zeitaufwändigen Vorgehen sollten abschließend noch doppelte Einträge entfernt werden, wenn zwei oder mehr Autoren an der gleichen Publikation beteiligt waren. Ziel ist es, dieses Vorgehen unter Verwendung entsprechender Mappings zu automatisieren. Ein erstes Mapping verknüpfe dabei in diesem Beispiel jeden Adressbucheintrag mit dem zugehörigen DBLP-

[5] http://www.amazon.de

Autor, während ein zweites Mapping zu jedem DBLP-Autoren die zugehörigen DBLP-Publikationen bereitstellt. Der Nutzer soll in die Lage versetzt werden, ein Verarbeitungsprogramm zu definieren, das diese Mappings traversiert und die gefundenen Publikationen nach dem Publikationsjahr ordnet. Gibt es zusätzlich Korrespondenzen der DBLP-Publikationen zu einer anderen Datenquelle, z. B. Google Scholar (GS) [3], die für jede Publikation X eine Zitierungszahl bereithält, d. h. die Anzahl der Publikationen Y, die X zitieren, sollen diese Informationen einfach in das bestehende Verarbeitungsprogramm aufgenommen werden. Dadurch können die gefundenen DBLP-Publikationen nach der GS-Zitierungszahl sortiert werden, um „einflussreiche“ wissenschaftliche Arbeiten zu identifizieren.

Object Matching: Während im oben geschilderten Szenario „lediglich“ bestehende Korrespondenzen verwendet, d. h. ausgewählt bzw. traversiert, werden, sollen Nutzer auch in die Lage versetzt werden, neue Korrespondenzen effizient zu erstellen. Dabei steht insbesondere das Object Matching, d. h. das Finden gleicher Objekte (meist in verschiedenen Datenquellen), im Vordergrund. Solche Korrespondenzen sind Voraussetzung für die oben genannte Informationsfusion. Ein Beispiel ist die Identifikation gleicher Publikationen bei DBLP und Google Scholar. Hierbei können neben dem Vergleich von Attributwerten, z. B. der syntaktischen Ähnlichkeit der Publikationstitel, auch bestehende Korrespondenzen (wieder-) verwendet werden. Gibt es z. B. für eine DBLP-Publikation D bereits eine Korrespondenz einer Publikation A einer anderen Datenquelle, z. B. ACM [1], die wiederum eine Korrespondenz zu einer Publikation G bei Google Scholar besitzt, kann durch die Verknüpfung beider Korrespondenzen eine Korrespondenz zwischen D und G, d. h. zwischen der DBLP- und der Google-Scholar-Publikation, erreicht werden. Auch hier ist das Ziel, einem Nutzer die Möglichkeit zur Definition und Ausführung von Verarbeitungsprogrammen zu geben, welche für eine Menge von Objekten zielgerichtet Korrespondenzen zwischen den jeweils gleichen Objekten ermitteln.

Die genannten Themengebiete werden in gesonderten Kapiteln – Kapitel 2 für Webseitenempfehlungen bzw. Kapitel 6 für Informationsfusion und Object Matching – näher vorgestellt, wobei vor allem ein Überblick über bisherige Arbeiten gegeben wird, sowie offene, in der vorliegenden Dissertation bearbeitete Probleme dargestellt werden.

1.2 Wissenschaftlicher Beitrag

Der wissenschaftliche Beitrag dieser Dissertation besteht aus den folgenden drei Arbeiten zur automatischen Mapping-Verarbeitung.

Adaptive Webseitenempfehlungen: Es wird mit AWESOME (Adaptive Website Recommendations) ein Ansatz zur Erstellung, Evaluation und automatischen Optimierung von Recommendations vorgestellt. Dazu wird eine Architektur präsentiert, welche die einfache Integration verschiedener Recommender ermöglicht, sowie Nutzer-Feedback, d. h. ob Nutzer die präsentierten Recommendations angeklickt haben oder nicht, automatisch aufzeichnet. Die Auswertung des Feedbacks für präsentierte Recommendations ermöglicht eine vergleichende Evaluation von Recommendern. Zusätzlich wird ein auf Selektionsregeln basierender Ansatz zur dynamischen Auswahl von Recommendern präsentiert. Dazu werden u. a. automatische Verfahren zur Generierung dieser Regeln vorgestellt, so dass der AWESOME-Ansatz eine vollautomatische Optimierung der Recommendations zur Steigerung der Recommendation-Qualität ermöglicht. Der AWESOME-Ansatz wurde prototypisch mit Hilfe einer Website implementiert und evaluiert. Dabei konnte gezeigt werden, dass eine adaptive Feedback-basierte Auswahl der Recommender die Recommendation-Qualität signifikant steigern kann. In der durchgeführten Evaluation erreichte die vollautomatische Selbstoptimierung mittels maschinellem Lernen vergleichbare Ergebnisse wie eine aufwändige, manuelle Optimierung.

Dynamische Informationsfusion: Der Datenintegrationsansatz iFuice (Information Fusion utilizing Instance Correspondences and Peer Mappings) stellt einen neuen Ansatz zur flexiblen Integration verschiedener Datenquellen dar. Im Gegensatz zu anderen, meist schemabasierten, Ansätzen stehen bei iFuice Peer-to-Peer-artige Mappings, d. h. Korrespondenzen zwischen Objektinstanzen, zwischen den Datenquellen im Mittelpunkt. Dabei wird die Semantik der Datenquellen und Mappings in einem sogenannten Domänenmodell reflektiert. Im Zentrum der Datenverarbeitung steht eine Skriptsprache mit High-Level-Operatoren, mit deren Hilfe Nutzer Workflows zur dynamischen Informationsfusion definieren können. Innerhalb des iFuice-Ansatzes wird eine Mediator-basierte Architektur zur Verarbeitung solcher Workflows vorgestellt. Mit Hilfe einer Beispielimplementation wird im Rahmen eines konkreten Anwendungsfalls (Zitierungsanalyse wissenschaftlicher Publikationen) die Einsatzfähigkeit des Ansatzes aufgezeigt.

Mapping-basiertes Object Matching: Das Matching-Framework MOMA (Mapping-based Object Matching) baut auf dem iFuice-Ansatz auf und dient der Erstellung von Workflows zum Object Matching, einem entscheidenden Aspekt der Datenintegration hinsichtlich der Datenqualität. Kernidee von MOMA ist die Kombination von Mappings zur Berechnung neuer Mappings, die – als sogenannte Same-Mappings – Object-Matching-Ergebnisse repräsentieren können. Dazu wird eine Architektur vorgestellt, welche auf der einen Seite die einfache Integration verschiedener Match-Verfahren ermöglicht und auf der anderen Seite die jeweiligen Ergebnisse, d. h. Mappings, flexibel miteinander kombinieren kann. Die Mapping-Verarbeitung wird dabei über iFuice-Operatoren realisiert.

Es werden dazu insbesondere verschiedene Kombinationsstrategien erarbeitet, die in unterschiedlichen Einsatzszenarien zur Anwendung kommen können. Die Effektivität der präsentierten Strategien wird anhand realer Daten evaluiert.

Die in der vorliegenden Arbeit dargestellten Ergebnisse wurden bereits als begutachtete Beiträge bei international führenden Konferenzen, Workshops und Journals publiziert. Der AWESOME-Ansatz wurde erstmalig bei der VLDB-Konferenz [191] vorgestellt. Die Arbeit wurde als eine der besten vier Papiere der Konferenz ausgewählt und erschien daher in einer längeren Version im VLDB-Journal des Folgejahres [189]. Der Datenintegrationsansatz iFuice wurde beim internationalen WebDB-Workshop publiziert [158]. Mit Hilfe von iFuice wurde eine Zitierungsanalyse wissenschaftlicher Publikationen aus dem Datenbankbereich durchgeführt, welche in SIGMOD Record erschien [157]. Innerhalb des internationalen IIWeb-Workshops wurden Konzept und Funktionsweise einer Online-Applikation zur Zitierungsanalyse veröffentlicht [188]. Architektur und Funktionsweise des MOMA-Frameworks wurden im Rahmen der CIDR-Konferenz vorgestellt [192].

1.3 Aufbau der Arbeit

Der Kern der vorliegenden Arbeit gliedert sich in zwei Teile. Der erste Teil – Adaptive Webseitenempfehlungen – stellt mit *AWESOME* einen Ansatz für adaptive Recommendations vor und gliedert sich in die folgenden Kapitel:

Kapitel 2 gibt eine Einführung in die Thematik von Recommender-Systemen. Neben der Vorstellung relevanter Literatur erfolgt insbesondere eine Klassifikation, mit deren Hilfe Recommender nach verschiedenen Kriterien eingeordnet werden können. Abschließend werden ausgewählte offene Probleme vorgestellt, die innerhalb der Arbeit angegangen werden.

Kapitel 3 beschreibt die wesentlichen Ideen und die Architektur des AWESOME-Ansatzes. Dazu werden die wichtigsten Komponenten und ihre Funktionsweise vorgestellt. Ein weiterer Schwerpunkt liegt auf dem Aufbau des Data Warehouses und dem zugehörigen Datenimport. Zusätzlich wird die prototypische Beispielimplementation skizziert, mit deren Hilfe die allgemeine Vorgehensweise zur Realisierung von AWESOME für beliebige Websites dargestellt wird.

Kapitel 4 definiert einen Ansatz zur dynamischen Bestimmung von Recommendations mit Hilfe von Selektionsregeln. Neben der manuellen Regelerstellung werden u. a. auch zwei automatische Verfahren zur Generierung solcher Regeln vorgestellt, die eine automatische Selbstoptimierung der Recommendations ermöglichen.

Kapitel 5 widmet sich der Evaluation des AWESOME-Ansatzes. Ausgehend von konkreten Evaluationsdaten der prototypischen Implementation wird das prinzipielle Vorgehen zur Evaluation der Recommendation-Qualität für eine beliebige Website aufgezeigt. Dazu werden unterschiedliche Arten der Evaluation vorgestellt, u. a. die kontextbasierte Recommender-Evaluation sowie die vergleichende Evaluation der in Kapitel 4 vorgestellten Selektionsstrategien.

Im zweiten Teil – Mapping-basierte Datenintegation – werden der Datenintegrationsansatz *iFuice* sowie das darauf aufbauende Object-Matching-Framework *MOMA* präsentiert. Die Darstellung gliedert sich in die folgenden Kapitel:

Kapitel 6 gibt einen Überblick über Datenintegration und deren Bedeutung. Der Schwerpunkt wird dabei auf Datenintegrationsansätze sowie auf die Problematik des Object Matchings, d. h. der Identifikation verschiedener Datensätze, die das gleiche Objekt der realen Welt beschreiben, gelegt. Das Kapitel schließt mit der Darstellung offener Probleme ab, die in den nachfolgenden Kapiteln bearbeitet werden.

Kapitel 7 stellt den neuartigen Datenintegrationsansatz iFuice vor. Nach der Darstellung der wesentlichen Ideen, inklusive einer Abgrenzung zu den bisherigen Ansätzen, werden Datenstrukturen und Operatoren charakterisiert, welche zu Skripten kombiniert werden können. Das Kapitel wird durch die Vorstellung der iFuice-Mediator-Architektur abgeschlossen.

Kapitel 8 stellt am Beispiel einer Zitierungsanalyse die Verwendung von iFuice zur Datenintegration vor. Nach einer Einführung in die wesentlichen Aspekte einer Zitierungsanalyse erfolgt die Darstellung einer kompletten Beispielanalyse, welche durch Skripte und entsprechende Abbildungen illustriert wird. Abschließend werden zwei auf iFuice basierende Applikationen vorgestellt, die für Zitierungsanalysen genutzt werden können.

Kapitel 9 stellt die Ideen und Konzepte des Object-Matching-Frameworks MOMA vor. Der Schwerpunkt der Darstellung liegt auf den Kombinationsmöglichkeiten von Mappings, welche in sogenannten Match-Workflows zum Einsatz kommen.

Kapitel 10 evaluiert die in Kapitel 9 vorgestellten Strategien zum Object Matching am Beispiel realer Daten aus der bibliografischen Domäne. Dabei werden die Eigenschaften der Match-Strategien und ihre Eignung innerhalb von Match-Workflows beschrieben.

Nach der abschließenden Zusammenfassung der Ergebnisse, inklusive Ausblick auf zukünftige Arbeiten, wird im Anhang die iFuice-Skriptsprache detailliert vorgestellt. Neben der formalen Definition des Aufbaus von Skripten und Prozeduren findet

sich eine vollständige Definition aller Operatoren inklusive Aufrufparametern. Zur Illustration sind an vielen Stellen Beispiele angegeben.

Teil II

Adaptive Webseitenempfehlungen

2

Recommender-Systeme

Dieses Kapitel beschäftigt sich mit Recommender-Systemen, d. h. Systemen zur Berechnung von Recommendations. Nach einer Einführung in die Thematik, welche die wesentlichen Grundbegriffe sowie die Bedeutung der Bestimmung von Recommendations darstellt (Abschnitt 2.1), erfolgt zunächst ein Abriss der relevanten Literatur (Abschnitt 2.2). Anschließend erfolgt die Vorstellung einer vollständigen dreistufigen Recommender-Klassifikation, welche die Berechnungsverfahren auf Basis der dabei verwendeten Informationen einordnet (Abschnitt 2.3). Das Kapitel wird durch die Darstellung der innerhalb der Arbeit bearbeiteten Probleme abgerundet (Abschnitt 2.4).

2.1 Motivation

Große Websites konfrontieren ihre Nutzer mit einer unüberschaubaren Menge von Informationen. Mit einer *Website* wird dabei eine komplette Internetpräsenz im World Wide Web beschrieben, z. B. der E-Commerce-Shop Amazon[6] oder die Enzyklopädie Wikipedia[7]. Die Größe einer Website lässt sich u. a. durch die Anzahl ihrer *Webseiten*[8], die in dieser Arbeit auch kurz Seiten genannt werden, charakterisie-

[6] http://www.amazon.de

[7] http://de.wikipedia.org

[8] In der deutschen Umgangssprache wird mit dem Begriff Webseite sowohl eine einzelne aufgerufene Seite als auch eine gesamte Internetpräsenz bezeichnet. Zur Differenzierung werden in dieser Arbeit der (englische) Begriff Website sowie der (deutsche) Begriff Webseite (*engl.* web page) verwendet.

ren. Amazon bietet z. B. aktuell ca. 3 Millionen deutschsprachige Bücher an[9], wobei jedes Produkt auf einer eigenen Webseite im Detail u. a. mit Preisangabe, Produktbeschreibung und Rezensionen präsentiert wird. Auch nichtkommerzielle Websites können eine große Anzahl von Seiten innerhalb der Website aufweisen. So hat z. B. die deutschsprachige Wikipedia mehr als 600.000 Artikel[10] (Stand: Oktober 2007).

Die große Anzahl der Webseiten erschwert die Navigation für die Website-Nutzer, die einzelne Seiten mittels eines Webbrowsers aufrufen. Ein *Pageview* umfasst dabei die komplette Darstellung einer aufgerufenen Webseite im Webbrowser des Nutzers. Während eines Website-Besuchs – einer sogenannten *Session* – eines Nutzers, betrachtet dieser einen oder mehrere Pageviews. Eine Session beschreibt dabei eine Liste zeitlich zusammenhängender Pageviews (z. B. innerhalb einer halben Stunde), die ein Nutzer auf einer Website aufgerufen und (vermutlich) betrachtet hat.

Um Nutzer innerhalb ihrer Session bei der Navigation zu unterstützen, bieten große Websites i. Allg. entsprechende Navigationshilfen an, da insbesondere im E-Commerce-Bereich z. B. nicht gefundene Produkte auch nicht gekauft werden (können). Ein Beispiel sind hierarchische Einordnungen der Seiten, z. B. durch Produktkategorien oder thematische Rubriken, die eine zielgerichtete Navigation erlauben. Mittels in Websites integrierter Suchmaschinen können Nutzer auch nach Informationen innerhalb der Website suchen. Solche Ansätze können dem Nutzer aber nur dann helfen, wenn er genau weiß, wonach er sucht. Sie eröffnen keine Möglichkeit, dem Nutzer potenziell interessante Seiten oder Produkte anzubieten, die außerhalb seiner durchgeführten Navigation oder Suche liegen.

An diesem Punkt setzen Recommendations an. Eine *Recommendation* ist eine Webseite R, die bei einem Pageview innerhalb einer anderen Webseite S dargestellt wird. Die Repräsentation der Recommendation erfolgt dabei mittels eines Hyperlinks auf R, der – optional zusammen mit einem kurzen Inhaltsangabe von R – innerhalb von S dargestellt wird. Innerhalb der vorliegenden Arbeit bezeichnen Recommendations dabei Webseiten der gleichen Website. Dieser Spezialfall stellt auf den typischen Einsatz von Recommendations zur Unterstützung der Nutzer bei der Navigation innerhalb einer Website ab. Im Rahmen einer Kooperation verschiedener Websites sind natürlich auch Recommendations auf Seiten anderer Websites möglich.

Eine manuelle Bestimmung von Recommendations ist bei großen Websites auf Grund des damit verbundenen Aufwands kaum möglich. Daher werden zu diesem Zwecke sogenannte *Recommender* eingesetzt. Ein Recommender bezeichnet ein Verfahren, dass für jede Webseite S einer Website – unter Berücksichtigung des aktuellen Kontexts (siehe unten) – eine Liste verschiedener Recommendations R ermittelt. Dabei wird die erste Recommendation der Liste als beste Recommendation des Recommenders interpretiert, die zweite als zweitbeste usw. Ein bekanntes Beispiel aus dem E-Commerce-Bereich ist „Kunden, die dieses Produkt gekauft haben, haben

[9] Ergebnis einer entsprechenden Suchanfrage im Oktober 2007
[10] http://de.wikipedia.org/wiki/Wikipedia

auch ... gekauft". Andere Verfahren empfehlen z. B. neue Produkte, zu einem Produkt ähnliche Produkte oder die am meisten verkauften Produkte der letzten Tage. Analog können z. B. nichtkommerzielle Websites die neuesten oder die am häufigsten aufgerufenen Seiten als Recommendation präsentieren. Verschiedene Recommender können dabei sowohl unterschiedliche Informationen als auch unterschiedliche Algorithmen zur Berechnung der Recommendations heranziehen. Eine Übersicht über die vielen in der Literatur vorgeschlagenen Verfahren zur Bestimmung von Recommendations wird in Abschnitt 2.2 gegeben.

Im Gegensatz zu den durch den Website-Administrator definierten (statischen) Links innerhalb einer Seite werden Recommendations in Abhängigkeit vom aktuellen Kontext dynamisch während eines Pageviews in die Webseite S „eingebaut" und angezeigt. Der *Kontext* umfasst dabei alle Informationen, die für einen Pageview – innerhalb dessen die Recommendation dargestellt werden soll – zur Verfügung stehen. Dabei lassen sich drei Arten von Informationen unterscheiden:

Informationen zur Webseite beschreiben die aufgerufene Seite (URL) sowie abgeleitete (zusätzliche) Informationen, z. B. die Einordnung in thematische Kategorien oder das Datum der letzten Änderung.

Informationen zum Nutzer umfassen sowohl „technische" Angaben zur Charakterisierung des Nutzers, d. h. IP-Adresse des verwendeten Rechners (aus denen regionale Informationen, z. B. Land oder Stadt, abgeleitet werden können) oder Agent-Name des verwendeten Browsers sowie auch evtl. zur Verfügung stehende nutzerbezogene Angaben, z. B. welche Seiten oder Produkte er bei seinem letzten Website-Besuch angesehen bzw. gekauft hat.

Weitere Informationen charakterisieren den Pageview unabhängig von der Seite und dem Nutzer, also z. B. Datum und Uhrzeit des Pageviews.

Als Beispiel wird ein Zugriff vom Arbeitsplatzrechner des Autors auf die Startseite der Website der Abteilung Datenbanken[11] betrachtet. Für den Pageview stehen dann u. a. die in Tabelle 2.1 aufgeführten Kontextinformationen zur Verfügung. Als Informationen zur aufgerufenen Seite wird die URL vermerkt. Zusätzlich können die Seiten einer Website kategorisiert sein, so dass im Beispiel die Startseite den Seitentyp „Navigation" erhält. Für den Nutzer steht zunächst die IP-Adresse seines Rechners zur Verfügung, welcher durch entsprechende Dienste[12] regionale Informationen, z. B. das Land, zugeordnet werden können. Kann der Nutzer als wiederkehrender Nutzer identifiziert werden, steht u. a. seine persönliche „Nutzungshistorie" der Website zur Verfügung. Diese kann, wie in Tabelle 2.1 angegeben, z. B. durch den Seitentyp (im Beispiel: Research) charakterisiert sein, zu dem die meisten aufgerufenen Seiten früherer Besuche gehören. Als weitere Kontextinformationen stehen

[11] http://dbs.uni-leipzig.de
[12] z. B. http://www.ripe.net/

Typ	Information	
Website	URL	index.html
	PageType	Navigation
Nutzer	IP	139.18.13.14
	Country	Germany
	History	Research
Sonstiges	Date	2007/08/22 11:15 AM
	DateType	Business

Tabelle 2.1: Beispiel für Kontextinformationen eines Pageviews

Datum und Uhrzeit des Pageviews zur Verfügung. Diese können z. B. danach charakterisiert werden, ob der Zeitpunkt in eine typische Arbeitszeit, z. B. Mo-Fr von 9-18 Uhr, fällt oder nicht.

2.1.1 Bedeutung

Insbesondere im E-Commerce-Bereich ist die wirtschaftliche Bedeutung von Recommendations hervorzuheben. Der Einsatz eines Recommender-Systems unterstützt die Nutzer bzw. Käufer beim Auffinden der für sie interessanten Produkte und verfolgt dadurch die folgenden drei Ziele eines Händlers im E-Commerce-Bereich [169].

Nutzer zu Kunden: Nutzer erreichen die Website eines Händlers nicht immer mit einer Kaufabsicht, z. B. wenn sie sich nur allgemein über eine Produktgruppe informieren wollen. Recommendations können den Nutzern helfen, die für sie interessantesten Produkte schnell zu identifizieren, so dass die Kundenneigung für einen Kauf erhöht wird.

Cross-Selling: Hat ein Kunde sich bereits für den Kauf eines Produkts entschieden, können ihm gezielt Angebote für zusätzliche Produkte gemacht werden (Cross-Selling). Eine Möglichkeit sind Zubehörprodukte, also z. B. DVD-Rohlinge beim Kauf eines DVD-Laufwerks. Alternativ können auch andere niedrigpreisige Produkte angeboten werden, mit denen der Gesamtverkaufspreis gezielt über eine Schwelle, z. B. ab der ein kostenloser Versand erfolgt, gehoben wird.

Kundenloyalität: Ähnlich dem klassischen Einzelhandel mit dem „Händler des Vertrauens“ können Recommendations die Loyalität des Kunden zu einem Händler verstärken. Sobald sich Händler in wichtigen Aspekten, u. a. Qualität der Produkte, Lieferzeit, Preisgestaltung und Service, nicht mehr von ihren Wettbewerbern unterscheiden, können kundenbezogene Recommendations einen entscheidenden Mehrwert generieren. Während der Einzelhändler

vor Ort den jeweiligen Geschmack seiner Stammkunden kennt, so „kennt der Online-Händler“ die komplette Kaufhistorie eines Kunden, d. h. welche Produkte er wann gekauft hat, und kann daraus gezielt neue Produkte empfehlen. Das über den Kunden verfügbare Wissen über seine Interessen, evtl. durch Lieblings- oder Wunschlisten vom Kunden explizit angegeben, kann ein entscheidender Wettbewerbsvorteil sein, der durch Recommendations ausgespielt wird.

Das Ziel des Recommender-Einsatzes im E-Commerce-Bereich ist die direkte Steigerung von Umsatz und Gewinn. Der Erfolg der Recommendations kann auch z. T. evaluiert werden, z. B. ob gegebene Produktempfehlungen angeklickt und später gekauft wurden. Auch bei nichtkommerziellen Websites, z. B. Websites für aktuelle Nachrichten, spielen Recommendations eine wichtige Rolle. Analog zur Fülle der angebotenen Produkte bei E-Commerce-Websites muss der Nutzer aus der Vielzahl der Informationen, z. B. der täglich erscheinenden Nachrichten, die für ihn relevanten auswählen. Dazu lässt z. B. die Website des Nachrichtenmagazins *Der Spiegel*[13] Lesern die Möglichkeit, Artikel durch Punktzahlen zu bewerten, so dass anderen Nutzern von Lesern empfohlene Artikel präsentiert werden können. Das Ziel ist, analog zur Kundenloyalität, der Aufbau einer Nutzerbindung, so dass Nutzer regelmäßig die Website aufsuchen. Der Wert dieser – zunächst monetär nicht erfassbaren – Bindung wird beim Thema Werbung deutlich. Analog zu den Preisen für Werbespots im Fernsehen oder Anzeigen in Zeitungen, die sich u. a. nach den Zuschauerzahlen bzw. der Auflage richten, orientiert sich die Bezahlung von Anzeigenwerbung auf Websites, z. B. durch Banner, u. a. an der Anzahl der Besucher, welche die Werbung gesehen („views“) oder angeklickt („clicks“) haben. Ein großer und regelmäßig wiederkehrender Nutzerkreis dient somit als Basis für Werbeeinnahmen.

2.2 Related Work: Recommender

Die Personalisierung von Websites ermöglicht Nutzern schnell und zielgerichtet die für sie relevanten Inhalte zu erreichen. Bei manueller Personalisierung gibt der Nutzer explizit seine Interessen an und erhält daraufhin ein entsprechendes Informationsangebot. Beispiele dafür sind MyYahoo![14] oder die personalisierte Startseite der Suchmaschine Google[15]. Bei der automatischen Personalisierung werden Verfahren angewendet, welche z. B. das Nutzungsverhalten analysieren, um die Website entsprechend aufzubereiten. Dies kann z. B. zu einer Reorganisation der Website führen [148, 149].

Die häufigste Form der Personalisierung stellen jedoch Recommender dar, deren

[13] http://www.spiegel.de
[14] http://my.yahoo.com
[15] http://www.google.com/ig

Recommendations zusätzlich innerhalb der Webseiten dargestellt werden. Recommender-Systeme sind ein wichtiges Forschungsgebiet der letzten Jahre, welches im Wesentlichen in den Bereichen (Web) Data Mining [107] und Human Computer Interaction angesiedelt ist. Die zunehmende Verbreitung von Recommender-Systemen führt mittlerweile auch zu Missbrauchsversuchen [26], da manipulierte Recommendations insbesondere im E-Commerce-Bereich erhebliche Auswirkungen haben können.

Das Ziel von Recommender-Systemen ist es, die *besten* Recommendations zu ermitteln und dem Nutzer zu präsentieren. Die grundsätzliche Schwierigkeit zeigt sich darin, dass die Frage, welche die beste Recommendation ist, nicht allgemein beantwortbar ist. Es spielen offenbar mehrere Kriterien eine Rolle, z. B. ob es sich um einen erfahrenen Nutzer handelt, dem eine Recommendation präsentiert wird, oder ob der Nutzer die Website zum ersten Mal besucht. Im ersten Fall sind z. B. Hinweise auf neue oder veränderte Seiten vielversprechend, wohingegen einem neuen Nutzer durch Recommendations auf häufig angesehene (aufgerufene) Seiten ein Überblick verschafft werden kann.

Grundsätzlich lassen sich die drei folgenden Schwerpunkte für Recommender-Systeme identifizieren, die z. T. in den nachfolgenden Abschnitten näher beleuchtet werden.

Berechnungsmodell: Soll die Bestimmung von Recommendations automatisch erfolgen, muss festgelegt werden, mit welchem Algorithmus und welchen Daten die Empfehlungen ermittelt werden sollen. Die Vielzahl der Algorithmen für Recommender wird in Abschnitt 2.2.1 näher vorgestellt, welche sich innerhalb hybrider Recommender-Systeme (Abschnitt 2.2.2) kombinieren lassen.

Präsentation: Nach ihrer Berechnung müssen die ermittelten Recommendations dem Nutzer präsentiert werden, d. h. in die aufgerufene Webseite „eingebaut“ werden. Dabei sollen die Recommendations natürlich deutlich sichtbar sein, auf der anderen Seite aber auch nicht zu aufdringlich wirklich und dadurch den Nutzer „verschrecken“. Die Recommendation-Präsentation beschäftigt sich daher im Wesentlichen mit dem Layout der Website sowie der kognitiven Erfassung durch Menschen (siehe z. B. [76]), was nicht Schwerpunkt dieser Arbeit ist.

Evaluation des Erfolgs: Wie oben beschrieben, ist die Bedeutung von Recommendations insbesondere im E-Commerce-Bereich mit einem wirtschaftlichen Ziel verknüpft. Daher ist eine Evaluation der Recommender-Systeme bzw. der präsentierten Recommendations nötig, um den Einfluss von Empfehlungen zu bestimmen und ggf. zu verbessern. Daher werden in Abschnitt 2.2.3 entsprechende Ansätze zur Evaluation vorgestellt.

2.2.1 Verfahren zur Berechnung von Recommendations

Eine Übersicht über Recommender findet sich in [4, 95, 108, 150, 182]. Im Wesentlichen werden zwei Arten von Recommender-Systemen unterschieden:

Inhaltsbasiert: Die für einen Nutzer assoziierten Inhalte, z. B. welche Seiten er besucht hat, werden analysiert. Es werden ähnliche Seiten als Empfehlung ermittelt, also z. B. diejenigen Seiten, die mit möglichst vielen der bisher besuchten Seiten eine große Ähnlichkeit aufweisen.

Kollaborativ (Collaborative Filtering): Für einen Nutzer werden zunächst ähnliche Nutzer identifiziert, wobei die Ähnlichkeit je nach Domäne über eine ähnliche Bewertung der gleichen Produkte, z. B. Filme, oder über ein ähnliches Navigationsverhalten bestimmt wird. Nur die von den ähnlichen Nutzern zur Verfügung stehenden Informationen, z. B. welche Seiten sie besucht haben, werden zur Berechnung von Empfehlungen verwendet.

Zusätzlich gibt es noch weitere Arten, z. B. wissensbasierte Recommender, welche domänenspezifisches Hintergrundwissen verwenden. Hybride Systeme, welche mehrere Recommender kombinieren, werden gesondert im Abschnitt 2.2.2 diskutiert.

Inhaltsbasierte Verfahren stammen aus dem Text/Data-Mining-Bereich und versuchen Ähnlichkeiten zwischen Seiten zu bestimmen und auszuwerten. Eine häufig angewendetes und aus dem Suchmaschinenbereich bekanntes Ähnlichkeitsmaß ist TF·IDF[16] [180].

Im Bereich kollaborativer Recommender-Systeme zeigen sich drei große Einsatzbereiche, die im Folgenden kurz vorgestellt werden.

Nutzerbewertungen: Die ersten Recommender-Systeme entstanden innerhalb einer aktiven Nutzergemeinde, welche jeweils mit Hilfe von Bewertungen Nutzerprofile aufgebaut hat. Beispiele sind GroupLens für Nachrichten des UseNets [104], Movielens für Filme [128], Tapestry für E-Mails [69] sowie Ringo für Musik [174]. Für einen Nutzer können z. B. auf Grund seiner Filmbewertungen diejenigen anderen Nutzer identifiziert werden, welche die größte Übereinstimmung in der Bewertung haben. Ausgehend von den so identifizierten ähnlichen Nutzern werden dann gut bewertete Filme empfohlen, die der Nutzer selbst noch nicht bewertet (bzw. gesehen) hat.

Nutzungsverhalten: Obige Nutzerbewertungen setzen durch die Bewertungen einen entsprechenden Aufwand für Nutzer voraus. Um auch ohne zusätzlichen Nutzeraufwand Recommendations bestimmen zu können, wurde in den

[16] TF·IDF: term frequency inverse document frequency

letzten Jahren verstärkt das Webnutzungsverhalten ausgewertet (Web Usage Mining, siehe [151] für eine Übersicht).

Eine Möglichkeit besteht darin, das Nutzungsverhalten direkt im Browser aufzuzeichnen. So wird z. B. bei Letizia [116] ein entsprechendes Programm in den Browser integriert, das das Nutzungsverhalten aufzeichnet. Auf jeder Seite überprüft es den Inhalt der verlinkten Seiten und gibt – auf Grund des Inhalts der bisher besuchten Seiten – Empfehlungen für potenziell interessante Seiten. Dieser Ansatz funktioniert für beliebige Websites, ist aber auf einen Nutzer beschränkt. Um auch das Nutzungsverhalten anderer Nutzer für Empfehlungen verwenden zu können, speichert WebWatcher [96] alle Navigationsschritte von Nutzern zentral ab. Dabei ist es nötig, eine Session auf der zentralen WebWatcher-Startseite zu beginnen. Anschließend kann ein Nutzer sich z. B. für alle Links auf einer Seite die Anzahl der Nutzer anzeigen lassen, die diesen Link angeklickt haben, oder sich – bzgl. der aktuellen Seite – ähnliche Seiten vorschlagen lassen.

Um auch Nutzungsverhalten ohne Installation eines Browser-Plugins (Letizia) oder Verwendung einer zentralen Einstiegsseite (WebWatcher) zu verwenden, werden die Logfiles der Webserver ausgewertet. Logfiles speichern die Zugriffe auf eine Website, d. h. welche Dateien (u. a. HTML-Seiten, Bilder, Stylesheets) wann von welchem Rechner angefordert wurden. Einfache statistische Verfahren, z. B. Footprints [200], bestimmen für jede Seite den häufigsten Nachfolger, d. h. zu welcher Seite die meisten anderen Nutzer von der aktuellen Seite navigiert sind. Durch die Bildung von Sessions [130, 131, 35] werden Navigationsmuster erkannt. Dazu werden die Sessions z. B. als Vektor repräsentiert und ähnliche Nutzer-Sessions mittels Abstandsfunktionen ermittelt. Dadurch gelingt es, analog zu den Nutzerbewertungen, ähnliche Nutzer zu identifizieren.

Kaufverhalten: Analog zum Nutzungsverhalten analysieren E-Commerce-Websites das Kaufverhalten ihrer Kunden. Ein prominentes Beispiel ist Amazon, welches u. a. eine Warenkorbanalyse durchführt [119]. Dabei werden statistische Untersuchungen durchgeführt, welche Produkte vom gleichen Nutzer gekauft worden sind.

Um auch mit großen Datenmengen effektiv Collaborative-Filtering-Algorithmen durchführen zu können, gibt es entsprechende Arbeiten zur Optimierung (u. a. [207]).

2.2.2 Hybride Recommender-Systeme

Hybride Recommender-Systeme kombinieren zwei oder mehrere der oben beschriebenen Verfahren. Dabei sollen die Schwächen der einzelnen Verfahren durch die Verwendung mehrerer Recommender ausgeglichen werden. In [25] werden dazu hybride

Recommender-Systeme u. a. nach der Art und Weise der Kombination verglichen. Dabei werden drei Methoden unterschieden:

Weighted: Jeder Recommender bewertet alle von ihm ermittelten Recommendation mit einem Gewicht (nicht ermittelte Recommendations erhalten ein minimales Gewicht von 0). Alle Gewichte einer Recommendation werden anschließend kombiniert, z. B. durch Bildung des arithmetischen Mittels, und die Recommendations mit den höchsten Gewichten werden angezeigt.

Switching: Das System wählt einen der verfügbaren Recommender aus und präsentiert dessen Recommendations.

Mixed: Die Recommendations verschiedener Recommender werden gemeinsam in einer Liste dargestellt. Im Unterschied zu Weighted werden dabei die Recommendations nicht von jedem Recommender mit einem Gewicht versehen, so dass im Vergleich zu Weighted z. B. nur von einem Recommender ermittelte Recommendations bevorzugt werden.

Für die drei Kriterien werden im Folgenden neuere, d. h. nicht in [25] erfasste, Recommender-Systeme kurz vorgestellt.

Weighted: In [170] werden Nutzerkriterien manuell Gewichte zugeordnet, die das Ranking innerhalb der Recommendation-Liste beeinflussen. [170] bestimmt Empfehlungen für Kinofilme und der Nutzer steuert durch die Angabe der Wichtigkeit seiner Präferenzen, z. B. wie wichtig ihm eine Filmlänge unter 120 Minuten ist, die relative Gewichtung der Recommender, um einen Gesamtwert (*score*) für eine Filmempfehlung zu ermitteln.

Switching: Die Verbundenheit der aktuellen Seite mit den Recommendations verschiedener Recommender wird in [134] bestimmt. Auf Grund der Hyperlink-Struktur der Website wird analysiert, welche Recommendation-Liste die stärkste Verbindung zur aktuellen Seite besitzt und angezeigt werden soll. Dem gegenüber steht in [194] ein System, welches den Nutzer explizit auffordert, die empfohlenen Seiten auf einer Skala von 1 bis 5 zu bewerten. Mit Hilfe dieses expliziten Nutzer-Feedbacks wählt das System einen der beiden integrierten Recommender aus.

Mixed: In [118] wird ein hybrides Recommender-System mit drei verschiedenen Recommendern vorgestellt. Jeder Recommender erzeugt eine Liste von Recommendations, welche anhand von Recommender-Gewichten zu einer Liste aggregiert wird. Die Gewichte bestimmen das Ranking der Recommendations und werden periodisch in einem Offline-Prozess danach aktualisiert, wie oft eine von einem Recommender bestimmte Recommendation im Laufe der Nutzer-Session angeklickt wurde. Die zu Grunde liegende Annahme ist, dass eine

im späteren Verlauf der Session aufgerufene Webseite eine gute Empfehlung gewesen wäre. Ein ähnlicher Ansatz findet sich im Yoda-System [172].

Die vorgestellten drei Kriterien machen keine Einschränkungen für die im hybriden Recommender-System verwendeten Recommender. Zusätzlich zu den beschriebenen Kombinationsarten sind in [25] noch weitere Varianten aufgeführt, z. B. eine Kombination der Vorhersagemodelle zweier Recommender (Meta-Level). Dabei beschreibt ein Vorhersagemodell z. B. die Nutzerinteressen, die dann von einem anderen Recommender als eine Eingabe verwendet werden können. Solche Kombinationsarten erheben jedoch Ansprüche an die Algorithmen der zu kombinierenden Recommender, z. B. die Verwendung solcher Vorhersagemodelle, und sind deshalb nur eingeschränkt anwendbar.

2.2.3 Evaluation

Die Evaluation von Recommendern gestaltet sich schwierig, da nicht eindeutig definiert werden kann, was *gute* Recommendations bzw. Recommender sind. Es finden sich daher verschiedene Ansätze in der Literatur, die im Folgenden kurz dargestellt werden. Eine weitere Schwierigkeit ist die schlechte Vergleichbarkeit von Evaluationsergebnissen unterschiedlicher Arbeiten, um z. B. *den* besten Recommender zu bestimmen. Der Grund liegt in der Verwendung unterschiedlicher Websites und zugehöriger Nutzer, so dass die Ergebnisse einer Untersuchung nicht auf andere Websites und Nutzer übertragen werden können.

In [67, 168] werden Recommender offline evaluiert, d. h. mit Hilfe des aufgezeichneten Webnutzungsverhalten oder Kaufdaten werden Recommendations bestimmt und überprüft, ob diese im weiteren Verlauf der Nutzer-Session erreicht werden. Dabei kommen mit Precision und Recall Maße aus dem Information Retrieval [180] zum Einsatz. Diese Art der Evaluation basiert jedoch auf der Annahme, dass korrekt vorhergesagte Seiten auch gleichzeitig gute Empfehlungen sind. Recommendations sollen aber insbesondere auch für den Nutzer interessante Seiten ermitteln, die er von allein nicht aufgesucht hätte. Die Qualität der Empfehlung bleibt also mit solchen Metriken unklar.

Andere Arbeiten [93, 101] bewerten die Qualität der Recommendations mit Hilfe von definierten Aufgaben, welche Nutzer mit Hilfe der Website lösen müssen, z. B. das Ermitteln einer bestimmten Information. Dabei werden die Nutzer in zwei Gruppen eingeteilt, wobei nur eine innerhalb der Website Recommendations angezeigt bekommt. Dabei kann z. B. die Anzahl der Klicks gemessen werden, die für die Lösung eines Problems nötig war, um so beide Gruppen miteinander vergleichen zu können. Zusätzlich können Nutzer direkt die Qualität der Recommendations verbal beurteilen, indem sie während der Benutzung ihre Gedanken bezüglich der gegebenen Recommendations, z. B. ob sie hilfreich waren, äußern. In [178] werden die Empfehlungen von Recommender-Systemen mit denen von Freunden, welche die jeweiligen

Nutzer gut kennen, verglichen. Die Ergebnisse zeigten, dass im Durchschnitt die Freunde bessere Recommendations gaben, obgleich der Vorteil automatischer Verfahren darin liegt, „neue“ bzw. „unerwartete“ Empfehlungen geben zu können. Das Hauptproblem der Evaluation von Recommender-Systemen unter Einbeziehung von Nutzergruppen liegt jedoch im großen Aufwand, was eine Anwendung für viele Nutzer nur sehr schlecht möglich macht.

Aktive Metriken messen das Verhalten der Nutzer bzgl. der gegebenen Recommendations. In [86] wird eine Architektur vorgestellt, die den Vergleich von zwei Recommendern erlaubt. Dabei werden die Recommendations beider Recommender dem Nutzer präsentiert (z. B. innerhalb einer gemischte Liste oder getrennt in zwei Listen) und er entscheidet, welche Empfehlung er als die beste erachtet. Dadurch kann eine relative Recommendation-Qualität gemessen werden.

Das REFEREE-Framework [41] erlaubt das Evaluieren verschiedener Recommender innerhalb der ResearchIndex/Citeseer-Website. Die eingesetzten Recommender orientieren sich dabei an der bibliografischen Domäne und empfehlen z. B. Publikationen, die die aktuell dargestellte Publikation zitieren, oder Publikationen der gleichen Autoren. Dabei werden u. a. mit *Click-through* und *Purchase* Maße aus dem E-Commerce-Bereich adaptiert, wobei *Click-trough* dem Folgen einer Recommendation und *Purchase* dem Download eines Dokumentes entspricht.

Speziell für Collaborative Filtering gibt es in [87] eine Übersicht zur Evaluation dieser Klasse von Recommendern.

2.3 Recommender-Klassifikationen

Die Vielzahl unterschiedlicher Recommender führt zu verschiedenen Klassifikationsansätzen.

Eine der ersten Kategorisierungen charakterisiert fünf Recommender-Systeme [160] mittels drei Dimensionen. Die erste Dimension der technischen Eigenschaften, z. B. Inhalt, Berechnung und Darstellung der Recommendation, fokussiert auf die verwendeten Algorithmen sowie deren Umsetzung. Bei domänenspezifischen Merkmalen (zweite Dimension) steht u. a. der „Objekttyp“ der Recommendations im Mittelpunkt, d. h. ob konkrete „Dinge“, z. B. Artikel, Filme oder Personen, empfohlen werden oder Webseiten unterschiedlichen Typs. Als dritte Dimension werden die Nutzer betrachtet, d. h. u. a. welche Nutzer Recommendations präsentiert bekommen (alle oder nur angemeldete) und welche Nutzer durch ihr Verhalten (z. B. Bewertungen) zur Berechnung von Recommendations beitragen.

Die Klassifikation in [169] konzentriert sich auf Recommender im E-Commerce-Bereich und verwendet spezialisierte Kriterien, z. B. welche Daten durch den Kunden bereitgestellt werden (u. a. Website-Navigation, Schlagworte, persönliche Kaufhisto-

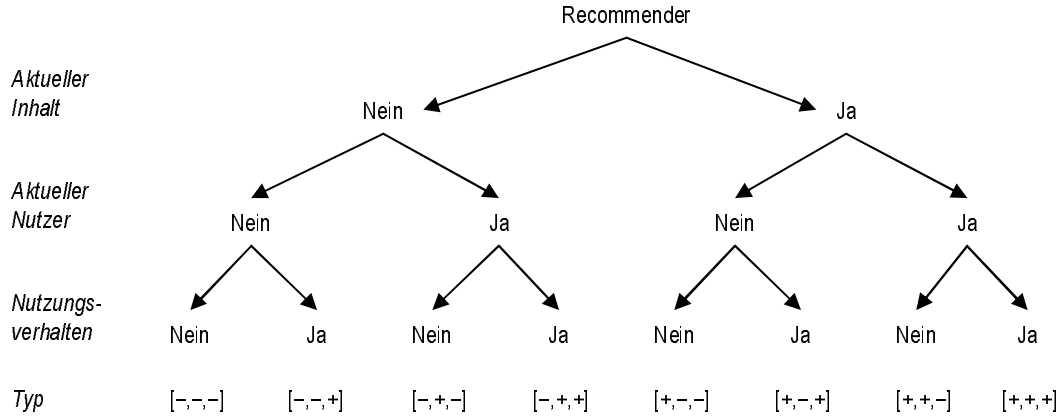


Abbildung 2.1: Vollständige 3-Ebenen-Klassifikation der Recommender

rie) oder den Grad der Personalisierung. Weiterhin wird hinsichtlich der Berechnung der Recommendations unterschieden, ob z. B. einfache statistische Verfahren oder Korrelationen zwischen Nutzern und/oder Produkten angewendet werden.

Eine gröbere Einteilung der verwendeten Techniken wird in [25] vorgenommen. Recommender werden dabei u. a. in kollaborative, inhaltsbasierte oder wissensbasierte Verfahren eingeteilt. Diese Trennung ist jedoch nicht scharf, da zum Beispiel wissensbasierte Verfahren auch die Inhalte der Website heranziehen. Demgegenüber steht eine Einteilung von Recommendern in [108], welche auf den konkreten Algorithmus abstellt, d. h. ob z. B. Clustering, Reinforcement Learning oder Assoziationsregeln verwendet werden.

Der wesentliche Nachteil dieser Klassifikationsansätze ist jedoch, dass sie nicht vollständig sind, d. h. ein neu entwickelter Recommender kann ein neues Kriterium in der Klassifikation erfordern. Zusätzlich sind einige Klassifikationsansätze auf eine Domäne spezialisiert, z. B. [169] für den E-Commerce-Bereich, was eine Einordnung anderer Recommender aus anderen Domänen erschwert. Insbesondere für eine vergleichende Evaluation verschiedener Recommender bezüglich der Qualität ihrer Recommendation ist eine vollständige Klassifikation wünschenswert. Dadurch können neue, d. h. später hinzukommende, Recommender in die bestehende Klassifikation eingeordnet werden, so dass bisherige vergleichende Evaluationsergebnisse weiter genutzt werden können.

Aus den oben genannten Gründen wurde die in Abbildung 2.1 dargestellte Klassifikation entwickelt. Es handelt sich um eine vollständige 3-Ebenen-Klassifikation der Recommender bezüglich der verwendeten Daten zur Berechnung von Recommendations. Jede Ebene unterscheidet die Verwendung eines Kriteriums, d. h. ob der Recommender die entsprechenden Informationen für seine Berechnungen verwendet oder nicht. Die Kriterien orientieren sich an den Kontextinformationen (aktueller Seiteninhalt, aktueller Nutzer) sowie an dem aufgezeichneten Webnutzungsverhalten. Viele andere Kriterien bleiben unberücksichtigt, können aber zu einer – späteren –

Verfeinerung der Klassifikation genutzt werden. Die Verwendung mehrerer, voneinander unabhängiger Kriterien, die jeweils die Menge aller Recommender partitionieren, d. h. vollständig in disjunkte Klassen einteilen, ermöglicht eine große Flexibilität bei der späteren Recommender-Evaluation. Die Kriterien können beliebig kombiniert werden und lassen dadurch multidimensionale Vergleiche der Recommender zu.

Im Folgenden werden die drei – in Abbildung 2.1 links dargestellten – zur Klassifikation verwendeten Kriterien näher vorgestellt.

Aktueller Inhalt fasst alle Informationen zur aktuell dargestellten Seite zusammen, für die Recommendations berechnet werden sollen. Neben der Seite werden auch abgeleitete Informationen betrachtet, also z. B. thematische Kategorien, in welche die Seite eingeordnet wurde.

Aktueller Nutzer beschreibt diejenigen Daten, die über den Nutzer zur Verfügung stehen, also z. B. aus welchem Land er kommt oder wie er die Website erreicht hat (z. B. mittels Bookmark oder Verwendung einer Suchmaschine). Wird ein Nutzer wiedererkannt, z. B. durch einen Login-Prozess oder über ein permanentes Cookie, kann ein Nutzerprofil verwendet werden, z. B. aus welcher Kategorie die meisten Seiten stammen, die der Nutzer bei seinen früheren Besuchen angesehen hat. Zusätzlich sind noch manuelle Nutzereinstellungen möglich, d. h. wenn der Nutzer seine Interessen explizit formuliert hat.

Nutzungsverhalten definiert das aufgezeichnete Webnutzungsverhalten aller Nutzer oder z. B. die Kaufhistorie einer E-Commerce-Website. Daraus können z. B. Informationen abgeleitet werden, welche Seiten – im Durchschnitt von allen Nutzern – am häufigsten aufgerufen worden sind oder welche Seiten/Produkte häufig gemeinsam bei einem Website-Besuch betrachtet worden sind.

Die vollständige Klassifikation nach den drei oben genannten Kriterien führt zu acht Klassen, die – wie in Abbildung 2.1 dargestellt – einen entsprechenden dreistelligen Typ-Code als Bezeichner erhalten. Dieser beschreibt durch + und –, ob die einzelnen Kriterien verwendet werden oder nicht. So verwenden z. B. Recommender des Typs (+, +, –) Informationen zur akuellen Seite sowie zum aktuellen Nutzer, aber keine Informationen des aufgezeichneten Nutzungsverhaltens. Nachfolgend werden für alle acht Klassen beispielhaft Recommender angegeben, welche größtenteils in der prototypischen Implementierung (siehe Abschnitt 3.6.2) verwendet wurden.

(–, –, –) : Recommender dieses Typs verwenden keine Informationen der drei Kriterien, so dass insbesondere keine Informationen bzgl. der aufgerufenen Seite und des aktuellen Nutzers verwendet werden. Es können z. B. manuell definierte Recommendations, z. B. gezielte Werbung für Seiten bzw. Produkte, oder die neuesten bzw. vor kurzem veränderten Seiten angezeigt werden. Eine weitere Form sind zufällig generierte Recommendations.

(–, –, +) : Hier wird ausschließlich das Nutzungs- bzw. Kaufverhalten verwendet, aus dem mittels statistischen Verfahren Recommendations bestimmt werden. Beispiele sind die am häufigsten aufgerufenen Seiten oder Produkte, die in der letzten Zeit einen besonders starken Zuwachs an Zugriffen oder Käufen hatten.

(–, +, –) : Spezifische Informationen zum aktuellen Nutzer sind Grundlage der Recommender dieser Klasse. Sind frühere Website-Besuche des Nutzer verfügbar, können als Recommendations die neu hinzugekommenen Seiten seit seinem letzten Besuch, evtl. eingeschränkt auf die von ihm favorisierten Kategorien, berechnet werden. Für den Fall, dass der Nutzer von einer Suchmaschine die Website erreicht hat, stehen in den HTTP-Referrer-Informationen meist die dort eingegebenen Schlagworte zur Verfügung [109]. Ein Suchmaschinen-Recommender kann diese Schlagworte extrahieren und die dazu passenden Seiten als Recommendations (siehe auch Abschnitt 3.6.2) bestimmen.

(–, +, +) : Innerhalb dieser Klasse von Recommendern werden die Informationen über das Webnutzungsverhalten mit denen des aktuellen Nutzers kombiniert. Durch Analyse des Nutzungsverhaltens können Nutzungsprofile (wie z. B. in [131]) gebildet werden, für welche Recommendations vorgehalten werden. Alternativ können mittels Collaborative Filtering (z. B. [128]) automatisch Gruppen ähnlicher Nutzer erstellt werden. Anschließend werden die Informationen des aktuellen Nutzers einem Profil oder einer Gruppe zugeordnet und die entsprechenden Recommendations verwendet.

(+, –, –) : Recommender dieser Klasse verwenden die aktuell dargestellte Seite als alleinige Datengrundlage. Dadurch können sowohl ähnliche Seiten, z. B. durch Bestimmung der String-Ähnlichkeit der Produktbeschreibung mittels TF·IDF, oder aber auch neue Seiten der aktuellen Kategorie als Recommendations bestimmt werden.

(+, –, +) : Die Kombination der aktuell dargestellten Seite mit dem aufgezeichneten Webnutzungsverhalten erlaubt vielfältige Möglichkeiten. Durch Anwendung von Assoziationsregeln können häufig innerhalb einer Session gemeinsam besuchte Seiten bestimmt und empfohlen werden. Ein prominentes Beispiel ist Amazon's „Kunden, die dieses Produkt gekauft haben, haben auch ... gekauft". Eine spezielle Form ist dabei der häufigste Nachfolger, d. h. welche Seite am häufigsten als nächste angeklickt wurde. Neben der aktuellen Seite selbst können auch Kategorisierungen verwendet werden, um z. B. die am häufigsten aufgerufenen Seiten der aktuellen Kategorie zu empfehlen.

(+, +, –) : In dieser Kategorie kann das Profil des aktuellen Nutzers, d. h. z. B. welche Seiten er bei seinen letzten Besuchen betrachtet hat, mit Informationen zur der aktuellen Seite abgeglichen werden. Dabei können z. B. Seiten empfohlen werden, die in der aktuellen Kategorie seit seinem letzten Besuch neu hinzugekommen sind.

(+, +, +) : Hier werden die meisten Informationen zur Berechnung von Recommendations genutzt. Ein möglicher Recommender bestimmt zunächst wie bei Typ (–, +, +) Nutzergruppen und ordnet den aktuellen Nutzer zu. Anschließend werden mit Hilfe von Assoziationsregeln – siehe (+, –, +) – entsprechende Recommendations getrennt nach Nutzergruppen, d. h. nur das Webnutzungsverhalten dieser Nutzer wird verwendet, bestimmt. Dadurch werden Seiten empfohlen, die häufig von ähnlichen Nutzern gemeinsam mit der aktuellen angezeigten Seite betrachtet werden.

2.4 Problemstellung

Die bisherigen Untersuchungen im Bereich Recommender-Systeme zeigen, dass es keinen besten Recommender gibt. Des Weiteren hängt die Qualität der Recommendations von vielen Faktoren ab, u. a. vom Nutzer und seinen Kenntnissen von der Website, aber auch von der aktuellen Seite. In diesem Zusammenhang treten folgende offene Fragestellungen auf:

- Wie lassen sich verschiedene Verfahren zur Berechnung von Recommendations effektiv innerhalb einer Website integrieren?
- Wie lässt sich die Qualität einer Recommendation oder eines Recommenders flexibel nach verschiedenen Kriterien evaluieren?
- Wie lässt sich die Qualität der präsentierten Recommendations automatisch optimieren?

Zur Beantwortung der aufgeführten Fragen werden im Folgenden die daraus resultierenden Anforderungen abgeleitet, welche in dieser Arbeit angegangen werden.

Integration verschiedener Recommender: Die Verwendung mehrerer Recommender ist für optimale Recommendations notwendig, da es nicht „den“ besten Recommender gibt. Gleichzeitig steht die Auswahl der relevanten Recommender nicht immer bereits zur Erstellungszeit der Website fest, so dass neue Recommender, z. B. mit anderen Berechnungsverfahren oder optimierten Parametern, im Laufe der Zeit hinzugefügt werden sollten. Es bedarf daher einer Architektur, welche die *einfache Erweiterbarkeit* um neue Recommender unterstützt. Um eine möglichst große Flexibilität zu erreichen, ist eine entsprechende Modularisierung notwendig, so dass ein Recommender in der Art eines „Plugins“ hinzugefügt werden kann. Dabei sollten *keine Einschränkungen* bezüglich der internen Realisierung des Recommenders gegeben sein, so dass unterschiedliche Algorithmen sowie unterschiedliche Programmiersprachen zum Einsatz kommen können. Um die Entwicklung der Recommender zu

vereinfachen, ist eine *einheitliche Datenbasis* mit allen für die Website relevanten Daten, u. a. Nutzungsverhalten und Inhalte der einzelnen Seiten, von Vorteil, auf die alle Recommender zugreifen können.

Evaluation der Recommendation-Qualität: Offenbar hängt die Qualität der ermittelten Recommendations von verschiedenen Faktoren ab, z. B. von der Website (Wie viele Produkte gibt es? Wie häufig gibt es neue Produkte?), den Nutzern und ihrem Kaufverhalten (Wie häufig wurde ein Produkt gekauft?), dem aktuellen Nutzer, dem die Recommendation präsentiert wird (Handelt es sich um einen Nutzer, der die Website zum ersten Mal besucht oder um einen wiederkehrenden Nutzer?), und der aktuellen Seite (Wie viele weitere Links gibt es auf der Seite?) ab. So ist z. B. beim Online-Händler Amazon der Recommender „Bücher des gleichen Autors“ sinnvoll für den Bereich der Belletristik, jedoch nicht unbedingt für ein Buch zum Erlernen einer Programmiersprache, dessen Autor weitere Bücher zu anderen Sprachen geschrieben hat. Es ist somit ein Ansatz nötig, der die Recommendation-Qualität erfasst und für eine flexible Auswertung bereithält. Dabei sollte eine Qualitätsbewertung jedoch *ohne zusätzlichen Nutzeraufwand* realisiert werden, da andernfalls nur für einen (geringen) Teil der Recommendations Feedback durch die Nutzer zu erwarten ist. Zusätzlich soll die Evaluation *mehrdimensional*, d. h. nach verschiedenen Kriterien, durchführbar sein, z. B. für eine bestimmte Nutzergruppe oder für einen bestimmten Teil der Website, d. h. für ausgewählte Seiten. Neben der absoluten Auswertung der Recommendation-Qualität ist insbesondere eine *vergleichende Evaluation* der verschiedenen Recommender sinnvoll. Dabei sollen die Stärken und Schwächen der einzelnen Verfahren in Abhängigkeit vom jeweiligen Kontext der Recommendations aufzeigt werden.

Optimierung der Recommendation-Qualität: Die Abhängigkeit der Recommendation-Qualität von den oben genannten Faktoren ist Beleg dafür, dass es keinen besten Recommender gibt, der immer die besten Recommendations ermittelt. Vielmehr haben die verschiedenen Recommender – je nach Kontext – unterschiedliche Stärken und Schwächen, was sich in der variierenden Qualität ihrer Recommendations niederschlägt. Um optimale Empfehlungen zu erhalten, müssen daher für einen gegebenen Kontext aus einem Pool möglicher Recommendations die besten ausgewählt werden. Analog zur Berechnung der Recommendations durch einzelne Recommender, muss die Auswahl der besten Recommendations ebenfalls *automatisch* realisiert werden, da eine manuelle Auswahl – insbesondere für große Websites mit vielen Nutzern – wahrscheinlich nur mit großem Aufwand machbar ist. Gleichzeitig muss die automatische Auswahl *adaptiv* erfolgen, um auf Veränderungen im Laufe der Zeit zu reagieren, die sich z. B. durch veränderte Nutzergruppen oder neue Webseiten ergeben.

2.5 Zusammenfassung

In diesem Kapitel wurde ein Überblick über Recommender-Systeme gegeben. Dabei wurde das breite Spektrum der Verfahren skizziert und Recommender insbesondere bzgl. der verwendeten Eingabedaten klassifiziert. Auf Basis der relevanten Literatur wurden abschließend drei offene Probleme identifiziert, welche die Grundlage für die folgenden Kapitel sind. Kapitel 3 thematisiert das Problem der flexiblen Integration mehrerer Recommender, Kapitel 4 stellt einen Ansatz zur Optimierung der Recommendation-Qualität vor und Kapitel 5 präsentiert eine Vorgehensweise zur Evaluation der Recommendations.

3 Der AWESOME-Ansatz

In diesem Kapitel werden die wesentlichen Konzepte von AWESOME (Adaptive Website Recommendations) vorgestellt. Nach der Darstellung der grundlegenden Ideen (Abschnitt 3.1) wird die Architektur von AWESOME präsentiert (Abschnitt 3.2). Dabei wird anschließend näher auf die für AWESOME relevanten Datenquellen (Abschnitt 3.3) sowie deren Verarbeitung zur Speicherung in einem Data Warehouse eingegangen (Abschnitte 3.4 und 3.5). Den Abschluss des Kapitels bildet die Illustration der Vorgehensweise, wie der AWESOME-Ansatz für beliebige Websites realisiert werden kann (Abschnitt 3.6).

3.1 Idee

Die grundsätzliche Idee von AWESOME baut auf einem *Data-Warehouse*-basierten hybriden Recommender-System auf, das unterschiedliche Recommender ausführen und die resultierenden Recommendations dem Nutzer präsentieren kann. Dabei ist AWESOME ein generischer, für beliebige Websites einsetzbarer Ansatz, der insbesondere die *Integration beliebiger Recommender* ermöglicht. Ein zentrales Element von AWESOME ist die automatische Aufzeichnung des Feedbacks bezüglich der präsentierten Recommendations. Um den Nutzern keine zusätzliche Arbeit, z. B. in Form expliziter Bewertung der Recommendations, zu machen, wird für jede präsentierte Recommendation vermerkt, welcher Recommender sie bei welchem Kontext ermittelt hat und ob sie durch den Nutzer akzeptiert, d. h. angeklickt, wurde oder nicht. Dadurch ist eine *Evaluation* der Recommendation-Qualität möglich, die

u. a. die Akzeptanzraten, d. h. wie viel Prozent der präsentierten Recommendations angeklickt wurden, verschiedener Recommender miteinander vergleichen kann. Neben der Evaluation liegt der Schwerpunkt bei AWESOME auf einer *automatischen Selbstoptimierung* der Recommendation-Qualität. Grundlage ist das aufgezeichnete Recommendation-Feedback, das zur Bestimmung neuer Recommendations ausgenutzt wird. Zu diesem Zweck wird das Feedback aggregiert, so dass für relevante, d. h. häufig auftretende, Kontextmuster (Kontexte mit unspezifizierten Kontextattributen) entschieden werden kann, welcher Recommender in der Vergangenheit besonders erfolgreich gewesen ist. Der vielversprechendste Recommender wird ausgewählt und seine für den aktuellen Kontext ermittelten Recommendations angezeigt.

Die wesentlichen – im Text kursiv hervorgehobenen – Konzepte des AWESOME-Ansatzes werden nachfolgend detaillierter vorgestellt:

Data-Warehouse: Der AWESOME-Ansatz basiert auf einem Data Warehouse als zentrale Datenquelle für das Webnutzungsverhalten, das Recommendation-Feedback sowie für Website-spezifische Informationen, z. B. thematische Kategorisierungen der einzelnen Seiten. AWESOME zeichnet dabei automatisch Feedback sowie Nutzungsverhalten auf und importiert es regelmäßig in das Data Warehouse. Dadurch stehen stets alle relevanten Informationen im Warehouse zur Verfügung, was die Basis für die Recommendation-Berechnung durch Recommender sowie für die Evaluation und die automatische Selbstoptimierung der Recommendation-Qualität ist (siehe folgende Punkte).

Integration beliebiger Recommender: Ein wichtiger Aspekt des AWESOME-Ansatzes ist die Einsetzbarkeit für beliebige Websites sowie insbesondere die Verwendung beliebiger Recommender. Durch das bereits angesprochene Data Warehouse können alle Recommender auf eine zentrale Datenbasis zurückgreifen, die alle verfügbaren Informationen zur Website als auch zum Nutzungsverhalten bereitstellt. Zusätzlich wird durch den modularen Aufbau einer Recommender-Bibliothek die Einbindung neuer Recommender durch die AWESOME-Architektur unterstützt. AWESOME ist also *kein* Ansatz für einen besonders guten Recommender, sondern vielmehr ein „Meta-Ansatz", der durch geschickte Kombination (d. h. Selektion) beliebiger zur Verfügung stehender Recommender eine verbesserte Recommendation-Qualität erreicht.

Evaluation: AWESOME stellt mit Hilfe des Data Warehouses eine Plattform zur Evaluation der Recommendation-Qualität und des Webnutzungsverhaltens zur Verfügung. Dadurch kann die Qualität der Recommender oder Recommender-Gruppen, z. B. alle Recommender einer Kategorie bzgl. der in Abschnitt 2.3 vorgestellten Klassifikation, analysiert werden. Dabei lassen sich unterschiedliche Maße verwenden, z. B. ob eine Recommendation angeklickt wurde, die empfohlene Seite betrachtet wurde oder ein empfohlenes Produkt gekauft wurde. Die Evaluation kann dabei unterschiedliche Kriterien einbeziehen, z. B.

Nutzergruppen (neue Nutzer im Vergleich zu wiederkehrenden Nutzern) oder die Kategorisierungen der einzelnen Seiten. Dadurch ermöglicht AWESOME eine flexible und zielgerichtete Analyse der Recommendation-Qualität der eingesetzten Recommender, was u. a. auch für ein manuelles Tuning der Recommender genutzt werden kann.

Automatische Selbstoptimierung: Ein weiterer Schwerpunkt von AWESOME besteht darin, mit Hilfe der bei der Evaluation der Recommendation-Qualität gewonnenen Informationen automatisch eine Optimierung der Recommendations zu realisieren. AWESOME ermöglicht dabei durch sogenannte Selektionsstrategien die Auswahl des vielversprechendsten Recommenders für einen gegebenen Kontext. Selektionsstrategien verwenden sogenannte Recommender-Selektionsregeln, die das Wissen, welcher Recommender bei welchen Kontextmustern besonders vielversprechend ist, abbilden. Der Fokus der Optimierung der Recommendation-Qualität liegt somit auf der Generierung solcher Selektionsregeln. AWESOME ist in der Lage, das aufgezeichnete Wissen (Recommendation-Feedback) automatisch, d. h. ohne manuellen Aufwand, in Selektionsregeln zu transformieren, was eine automatische Selbstoptimierung (Closed-Loop-Optimierung) ermöglicht: Das aufgezeichnete Recommendation-Feedback führt zur Auswahl eines Recommenders, dessen Recommendations dem Nutzer präsentiert werden, so dass erneut Recommendation-Feedback aufgezeichnet werden kann. Dadurch adaptiert AWESOME die Nutzerinteressen und ermöglicht insbesondere auch eine regelmäßige Anpassung bei Veränderungen der Website, der Recommender oder der Nutzerinteressen im Laufe der Zeit.

3.1.1 Vergleich mit anderen Recommender-Systemen

Das Ziel des AWESOME-Ansatzes ist es, die optimale Qualität der Recommendations zu erreichen, d. h. die präsentierten Empfehlungen möglichst optimal den Nutzerbedürfnissen anzupassen. Damit verfolgt AWESOME das gleiche Ziel wie die meisten anderen Forschungsarbeiten im Bereich der Recommender-Systeme. Um die Besonderheiten des AWESOME-Ansatzes zu verdeutlichen, wird er mit den bisherigen Arbeiten anhand dreier Kriterien verglichen.

Systemarchitektur: AWESOME ist ein Vertreter eines hybriden Recommender-Systems nach dem Switching-Prinzip (siehe Abschnitt 2.2.2), d. h. es wird ein Recommender ausgewählt und dessen Recommendations werden angezeigt. Im Unterschied zu den meisten bisherigen Arbeiten stellt der AWESOME-Ansatz ein generisches hybrides Recommender-System dar, das für beliebige Websites und beliebige Recommender ausgelegt ist. Die meisten Arbeiten konzentrieren sich auf ein System, dass für eine spezielle Website und/oder eine Menge fest

definierter Recommender konzipiert und realisiert wurde. Einzig das REFEREE-Framework fokussiert – ähnlich wie AWESOME – auf die einfache Integration beliebiger Recommender [41]. Allerdings bezieht sich REFEREE stark auf die CiteSeer-Website, so dass die Recommender in der Domäne bibliografischer Daten, d. h. Empfehlung interessanter wissenschaftlicher Publikationen, angesiedelt sind.

Evaluation: Der AWESOME-Ansatz beinhaltet ein allgemeines Vorgehen für die Evaluation von Recommendations bzw. Recommendern. Im Unterschied zu vielen Arbeiten [93, 101, 86] entsteht kein zusätzlicher Nutzeraufwand, bei dem Nutzer die Qualität der Recommendations absolut (z. B. auf einer Punkteskala) oder vergleichend („Recommendation A ist besser als Recommendation B“) bewerten müssen. AWESOME registriert dabei sowohl positives als auch negatives Feedback, d. h. ob eine Recommendation angeklickt wurde oder nicht. Der zweite entscheidende Punkt ist die mit dem Data Warehouse gewonnene Flexibilität der Evaluation. Das Design des Warehouses wird u. a. durch die für die Website zur Verfügung stehenden Daten als auch durch die Website-spezifischen Evaluationsmetriken charakterisiert. Dadurch kann die Evaluation flexibel nach verschiedenen Kriterien, z. B. Nutzergruppen oder Seitentyp, sowie verschiedenen Metriken, z. B. durchschnittliche Akzeptanzrate der Recommendations, durchgeführt werden. Es wird zwar eine beispielhafte Evaluation in Kapitel 5 für eine ausgewählte Website vorgestellt, die allerdings lediglich der Illustration der Vorgehensweise einer Data-Warehouse-basierten Evaluation für eine gegebene Website dient. Die konkreten Evaluationsergebnisse der Beispiel-Website sind dabei von sekundärem Interesse.

Optimierung: Bisherige Arbeiten zur Optimierung der Recommendation-Qualität fokussieren meist auf die Optimierung einzelner Algorithmen, z. B. durch eine Offline-Evaluation des Webnutzungsverhaltens, so dass die Recommendations möglichst gut die zukünftige Navigation der Nutzer hervorsagen [67, 168]. Der AWESOME-Ansatz unterscheidet sich davon in zwei wesentlichen Punkten. Erstens verwendet AWESOME aktive Metriken zur Evaluation, d. h. ob die Recommendation angeklickt (oder gekauft) wurde oder nicht. Zweitens optimiert AWESOME nicht einzelne Algorithmen sondern die Auswahl der Recommender. Die Recommender-Selektion kann dabei vollständig automatisch realisiert werden, was im Gegensatz zu den bisherigen Arbeiten [170, 194] keinen zusätzlichen Nutzeraufwand erzeugt. Eine zu AWESOME ähnliche Optimierung findet sich in [118] und [172]. Die vorgestellten Arbeiten verwenden allerdings eine begrenzte und vorher definierte Menge von Recommendern und sind daher nicht so flexibel wie der AWESOME-Ansatz, der für eine beliebige Menge von Recommendern eine Optimierung durchführen kann.

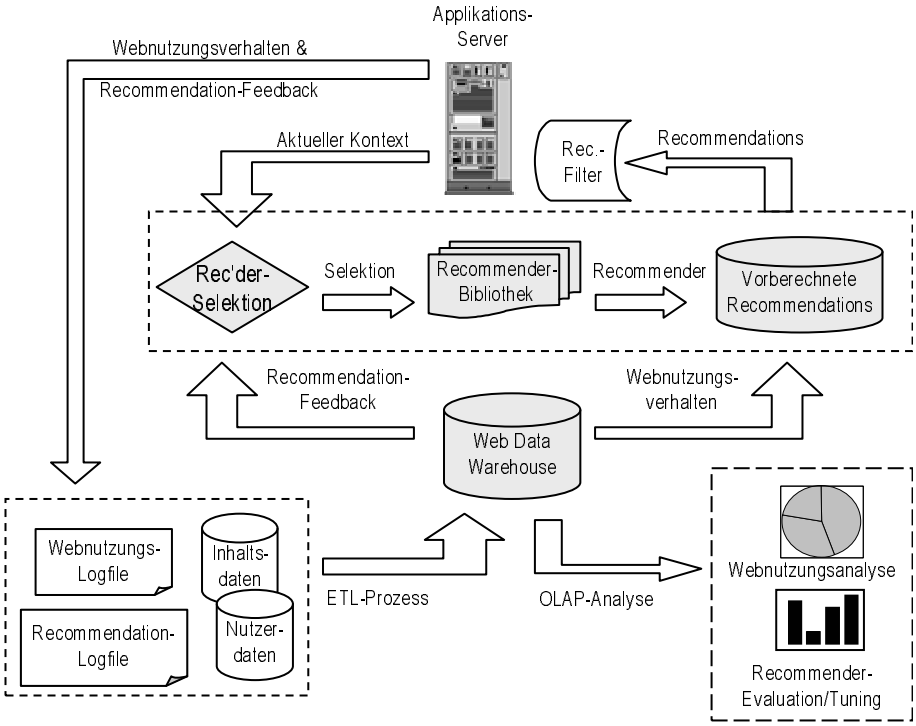


Abbildung 3.1: AWESOME-Architektur

3.2 Architektur

Die Gesamtarchitektur von AWESOME ist in Abbildung 3.1 dargestellt. Sie setzt auf einer bestehenden Website (bzw. deren Web-Applikations-Server) auf. AWESOME ermöglicht die dynamische Bestimmung und Darstellung einer Liste von Recommendations für jeden Webseitenzugriff, so dass die Webseitenempfehlungen dynamisch innerhalb der angeforderten Seite dargestellt werden können.

Die AWESOME-Architektur verfügt über die folgenden drei zentralen Bestandteile, welche in späteren Abschnitten näher dargestellt werden.

Web Data Warehouse: Das Warehouse ist die zentrale Datenplattform von AWESOME. Es speichert sowohl das Webnutzungsverhalten als auch das Feedback bzgl. der präsentierten Recommendations. Diese Informationen werden mit weiteren – je nach Website zur Verfügung stehenden – Daten kombiniert, z.B. einer (manuell erstellten) Kategorisierung der Webseiten. Die so integrierten Daten sind Grundlage für alle Recommender zur Bestimmung von Recommendations. Weiterhin dient das Data Warehouse der Recommender-Selektion, um für einen gegebenen Kontext den vielversprechendsten Recommender auszuwählen. Durch die automatische Aktualisierung des Warehouses wird dabei eine automatische Selbstoptimierung der Recommender-Selektion ermöglicht. Schließlich kann das Warehouse auch zur OLAP-Analyse (Online

Analytical Processing; zur näheren Definition siehe Abschnitt 5.1) genutzt werden, um sowohl das Webnutzungsverhalten als auch die Effektivität der Recommender bzw. der Selektionsstrategien systematisch zu evaluieren. Der Aufbau des Data Warehouses wird in Abschnitt 3.5 dargestellt.

Recommender-Bibliothek: Um die Integration beliebiger Recommender zu unterstützen, besitzt AWESOME eine erweiterbare Bibliothek von Recommendern, welche mittels einer einheitlichen Schnittstelle uniform verwendet werden können. Es werden dabei keine Einschränkung bzgl. der Funktionalität der Recommender gemacht, d. h. z. B. welche Berechnungsverfahren zum Einsatz kommen. In Kapitel 3.6.2 wird exemplarisch die Recommender-Bibliothek für die prototypische Implementation näher vorgestellt. Zusätzlich können alle Recommender nach verschiedenen Kriterien charakterisiert werden – z. B. nach der in Abschnitt 2.3 vorgestellten Klassifikation – was u. a. für die Evaluation der Recommender verwendet werden kann (siehe Kapitel 5). Die Recommender-Beschreibung wird dazu ebenfalls im Warehouse abgelegt (siehe Abschnitt 3.5) und kann somit bei einer späteren Evaluation mit einbezogen werden.

Recommender-Selektion: Kernpunkt der dynamischen Bestimmung von Recommendations ist die adaptive Auswahl von Recommendern. Das Selektionsmodul ist in der Lage, auf Grund des aufgezeichneten Recommendation-Feedbacks diejenigen Recommender auszuwählen, welche bei ähnlichen Kontexten (z. B. bei der gleichen angeforderten Seite) besonders gute Akzeptanzraten ihrer ermittelten Recommendations hatten. Zur Definition der Recommender-Auswahl werden Selektionsregeln gebildet, welche auf verschiedene Art und Weise (manuell sowie automatisch) erstellt werden können. In Kapitel 4 werden sowohl deren Aufbau als auch verschiedene Verfahren zu ihrer Erstellung vorgestellt.

Die Funktionsweise von AWESOME gliedert sich in einen Online- und einen Offline-Workflow. Ersterer wird bei jedem Webseitenzugriff ausgeführt, um die relevanten Recommendations zu bestimmen. Der Offline-Workflow wird periodisch ausgeführt (z. B. einmal am Tag) und dient der Datenaktualisierung.

Online-Workflow: Bei jedem Webseitenzugriff sind – neben der angeforderten Seiten – weitere Informationen verfügbar, z. B. die Uhrzeit des Zugriffs oder nutzerspezifische Informationen wie die IP-Adresse. Zusätzlich wird erfasst, ob die angeforderte Seite eine zuvor gegebene Recommendation ist, d. h. ob der Nutzer eine ihm präsentierte Empfehlung akzeptiert hat. Diese Informationen werden zum Kontext zusammengefasst und gespeichert (Webnutzungs-Logfile). Für den aktuellen Kontext liefert anschließend das Recommender-Selektionsmodul einen (oder mehrere) Recommender auf Grund des im Data Warehouse gespeicherten Feedbacks. Für den ausgewählten Recommender

werden anhand des Kontexts die aktuellen Recommendations bestimmt. Aus Gründen der Performanz kann dabei auf eine Datenbank mit vorberechneten Recommendations zurückgegriffen werden. Die so erhaltenen Empfehlungen werden an den Applikations-Server weitergeleitet, der diese in die angeforderte Webseite integriert. Dabei werden mit Hilfe eines Filters unsinnige bzw. unnötige Recommendations (z. B. Startseite oder aktuelle Seite) ausgeschlossen. Abschließend werden die innerhalb der angeforderten Seite dargestellten Recommendations aufgezeichnet (Recommendation-Logfile). Dadurch ist sichergestellt, dass sowohl alle akzeptierten als auch alle nicht akzeptierten Empfehlungen erfasst werden. Diese Informationen sind für die Bestimmung des Feedbacks und der daraus abgeleiteten Recommender-Selektion notwendig.

Offline-Workflow: Innerhalb des Offline-Workflows wird zunächst das Web Data Warehouse aktualisiert. Die generierten Logfiles werden innerhalb eines ETL-Prozesses (Extract, Transform, Load) in das Warehouse integriert. Dabei werden (sofern vorhanden) weitere Datenquellen hinzugezogen, die z. B. Informationen über den Inhalt der Website oder über deren Nutzer (z. B. abgespeicherte Nutzerprofile) beinhalten. Die wichtigsten innerhalb des ETL-Prozesses durchgeführten Datentransformationen (z. B. Crawler- und Session-Erkennung) werden im Abschnitt 3.4 näher dargestellt. Nach der Aktualisierung der Fakten- und Dimensionstabellen erfolgt eine Neuberechnung der Recommendations und der Selektionsregeln, welche – ebenfalls aus Performanzgründen – in eigenen Datenbanken abgelegt werden können. Dadurch ist es möglich, sowohl die Recommender-Selektion als auch die Bestimmung einzelner Recommendations im Rahmen des Online-Prozesses performant durchzuführen.

3.3 Datenquellen

Wie in Abbildung 3.1 dargestellt, unterscheidet der AWESOME-Ansatz zwischen zwei Arten von Datenquellen. Auf der einen Seite stehen Informationen zur Webnutzung sowie den präsentierten Recommendations, die als *Website-unabhängig* bezeichnet werden, da sie durch AWESOME generiert werden und somit in einer einheitlichen Struktur unabhängig von der konkreten Website zur Verfügung stehen. Im Verlauf dieses Abschnitts werden Struktur (Abschnitt 3.3.1 bzw. 3.3.2) sowie die Verarbeitung (Abschnitt 3.4) dieser Daten näher vorgestellt.

Auf der anderen Seite gibt es *Website-abhängige* Inhalts- und Nutzerdaten, d. h. Informationen zu den einzelnen Seiten sowie den Nutzern der Website. Struktur und Verfügbarkeit der Information zum Inhalt sowie den Nutzern unterscheiden sich zwischen verschiedenen Websites. Eine mögliche Form von Inhaltsdaten einer Website sind Kategorisierungen der einzelnen Seiten, d. h. Einordnungen der Seiten in hierarchisch angeordnete Kategorien. Abbildung 3.2 zeigt ein Beispiel für eine mögliche Seitenkategorisierung der in der prototypischen Implementation verwendeten

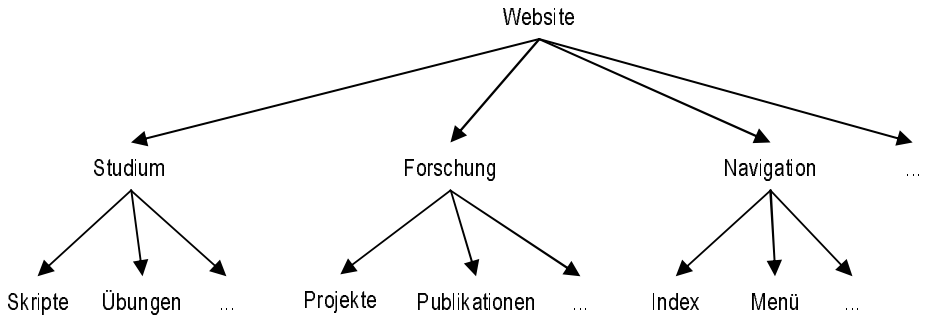


Abbildung 3.2: Beispiel für eine Kategorisierung von Webseiten

Website[17]. Für eine Website kann es selbstverständlich mehrere Arten der Kategorisierung geben. So kann sich z. B. eine Kategorisierung an der evtl. vorhandenen Menüstruktur einer Website orientieren, während eine andere Kategorisierung ausschließlich inhaltliche bzw. thematische Aspekte betrachtet. Es ist leicht einzusehen, dass eine inhaltliche Kategorisierung der Seiten für eine andere Website vollkommen anders aussieht. Die zweite Form von Daten bilden Informationen über die Nutzer der Website, welche insbesondere für Kunden von E-Commerce-Websites verfügbar sind. Deren Aufbau hängt im Wesentlichen davon ab, welche Informationen Kunden bei ihrer Registrierung bzw. vor dem Kauf eines Produkts angeben (müssen). Zusätzlich können Nutzerangaben auch mit weiteren Datenquellen verknüpft werden, um z. B. mittels sozio-ökonomischen Daten anhand der Kundenadresse das durchschnittliche Haushaltseinkommen abzuschätzen. Auf der anderen Seite sind für Websites ohne Nutzerregistrierung keinerlei nutzerspezifische Informationen verfügbar (außer „technischen“ Informationen, z. B. der IP-Adresse).

Im Folgenden stehen innerhalb dieses Abschnitts Aufbau und Verarbeitung der Website-unabhängigen Daten, d. h. des Webnutzungsverhaltens sowie der präsentierten Recommendations, im Mittelpunkt. (Die Berücksichtigung der Website-abhängigen Datenquellen erfolgt durch Hinweise an den entsprechenden Stellen, z. B. bei der Anpassung des Data-Warehouse-Schemas (Abschnitt 3.5)). Die dafür relevanten Informationen lassen sich jeweils in einer tabellarischen Form darstellen und werden über die Zeit aufgezeichnet. Daher wird der Begriff *Logfile* für die beiden Datenquellen verwendet. Die konkrete Implementierung setzt aber nicht zwingend ein (physisches) Logfile voraus, sondern kann auch mittels einer Datenbank oder einer anderen Dateistruktur (z. B. XML) realisiert werden.

3.3.1 Webnutzungs-Logfile

Die Webnutzung charakterisiert die Zugriffe der Nutzer auf die einzelnen Seiten der Websites. Üblicherweise generieren Webserver zu diesem Zweck bereits automatisch

[17] http://dbs.uni-leipzig.de

Pageview ID	eindeutige Kennzeichnung des Zugriffs
Page	angeforderte Seite (URL)
Date, Time	Datum und Uhrzeit des Zugriffs
Client	IP-Adresse des Rechners, Agent-Name des Browsers
Referrer	URL, von der aus der Zugriff erfolgte
Session ID	eindeutige Kennzeichnung der Session
User ID	eindeutige Kennzeichnung des Nutzers

Tabelle 3.1: Aufbau des Webnutzungs-Logfiles

standardisierte Logfiles, z. B. im Common Logfile Format[18] (CLF), durch Aufzeichnen der HTTP-Requests, welche Browser an die Websites schicken, um die Daten (HTML-Seiten, Bilder, Stylesheets, usw.) für den Nutzer anzufordern. Die aufgezeichneten Daten zeigen die Art und Weise, wie die Nutzer die Website „benutzen“ und bilden damit eine wichtige Informationsquelle, die u. a. zur Analyse des Webnutzungsverhaltens (siehe Abschnitt 5.2) verwendet werden kann.[19]

Tabelle 3.1 zeigt den Aufbau des Webnutzungs-Logfiles, d. h. die für jeden Pageview durch AWESOME aufgezeichneten Informationen. Neben der angeforderten Seite, Informationen zum Client (IP-Adresse, Agent-Name des verwendeten Browsers), Referrer sowie Datum und Uhrzeit des Zugriffs – die jeweils auch im CLF definiert sind – wird jedem Pageview eine (automatisch generierte) eindeutige ID hinzugefügt. Optional kann noch eine Session- und User-ID gespeichert werden, falls der Pageview einer Session und/oder einem Nutzer zugeordnet werden kann. Sämtliche IDs dienen ausschließlich der eindeutigen Identifikation, so dass z. B. die Informationen über einen Nutzer, z. B. identifiziert durch seinen Login-Namen, mit anderen Nutzerdaten kombiniert werden können.

3.3.2 Recommendation-Logfile

Die im Webnutzungs-Logfile aufgezeichneten Informationen repräsentieren lediglich die von Nutzern aufgerufenen Seiten. Es fehlen dabei sämtliche Informationen zu den Recommendations, d. h. welche Recommendations bei welchem Pageview dargestellt wurden und ob sie evtl. angeklickt wurden. Daher zeichnet AWESOME zusätzlich alle präsentierten Recommendations in einem Recommendation-Logfile auf.

Die für jede präsentierte Recommendation gespeicherten Informationen sind in Tabelle 3.2 zusammengefasst. Neben der durch die Recommendation empfohlenen Seite wird der Pageview (identifiziert durch die ID) vermerkt, bei welchem die Recommen-

[18] http://www.w3.org/Daemon/User/Config/Logging.html

[19] Auch für Suchmaschinen kann die Aufzeichnung des Nutzungsverhaltens nach dem Absetzen entsprechender Anfragen zur Verbesserung der Suchmaschine genutzt werden [99].

Pageview ID (FK)	ID des Pageviews, bei dem die Recommendation angezeigt wurde
Recommendation	empfohlener Inhalt (d. h. Seite, Produkt, ...)
Accepted	Wurde die Recommendation akzeptiert, d. h. angeklickt?
Recommender	Recommender, der die Recommendation ermittelt hat
Strategy	Selektionsstrategie, welche den Recommender bestimmt hat
Layout	Information zur Darstellung der Recommendation, z. B. Position der Recommendation innerhalb einer Liste von Recommendations

Tabelle 3.2: Aufbau des Recommendation-Logfiles

dation präsentiert wurde. Dadurch entsteht ein Bezug (Fremdschlüsselbeziehung) zwischen dem Recommendation- und dem Webnutzungs-Logfile, so dass für jede aufgezeichnete Recommendation die im Webnutzungs-Logfile enthaltenen Kontextinformationen zur Verfügung stehen. Da für jede präsentierte Recommendation zusätzlich erfasst wird, ob sie nach ihrer Präsentation auch angeklickt wurde oder nicht, ist die Aufzeichnung eines genauen Recommendation-Feedbacks möglich. Mit dessen Hilfe ist es z. B. bei einer späteren Evaluation möglich, die durchschnittliche Akzeptanzrate, d. h. wie viel Prozent der präsentierten Recommendations auch angeklickt wurden, festzustellen. Für jede präsentierte Recommendation wird auch der Recommender gespeichert, der die Recommendation berechnet hat. Diese Information ist entscheidend für die automatische Selbstoptimierung von AWESOME, die den vielversprechendsten Recommender dynamisch bestimmt. Dadurch kann z. B. die Akzeptanzrate für jeden Recommender einzeln bestimmt werden, was einen Vergleich der Qualität der einzelnen Recommender ermöglicht. Wie in Abschnitt 3.1 bereits kurz angedeutet, verwendet AWESOME zur Recommender-Selektion sogenannte Selektionsstrategien. Für eine spätere vergleichende Evaluation der verschiedenen Strategien, wird daher der (eindeutige) Strategiename vermerkt. Zusätzlich können noch Layout-Informationen zur Recommendation gespeichert werden, z. B. an welcher Position die Recommendation in einer Liste mehrere Recommendations dargestellt wurde. Dadurch kann später z. B. der Einfluss der Präsentation der Recommendations auf die Akzeptanzrate analysiert werden.

3.4 ETL-Prozess

Der ETL-Prozess (Extract, Transform, Load) [30] ist ein wichtiger Bestandteil von Data-Warehouse-Projekten, innerhalb dessen die zur Verfügung stehenden Daten aus den Datenquellen extrahiert, entsprechend den Anforderungen transformiert und in das Warehouse importiert werden. Dieser Prozess wird dabei i. Allg. (periodisch)

wiederholt, um regelmäßig einen aktuellen Datenbestand im Warehouse bereitzustellen. Entscheidend bei einem ETL-Prozess ist dabei ein ausführliches Data Cleaning, wobei Datenfehler behoben werden und Informationen aus verschiedenen Quellen kombiniert werden.

Die Datenqualität des Data Warehouses ist für AWESOME von entscheidender Bedeutung, da das Data Warehouse Grundlage für verschiedene Bereiche der AWESOME-Architektur ist (siehe Abbildung 3.1). So verfälschen z. B. Datenfehler in den Webnutzungsdaten bzw. im Recommendation-Feedback eine entsprechende Evaluation. Gleichzeitig kann das aufgezeichnete Webnutzungsverhalten von einigen Recommendern zur Berechnung von Recommendations genutzt werden, z. B. zur Empfehlung der in der letzten Woche am häufigsten aufgerufenen Seite. Unsaubere Webnutzungsdaten haben daher einen Einfluss auf die Recommendations, was wiederum deren Qualität negativ beeinflussen kann. Letztendlich erfordert auch die durch AWESOME ermöglichte automatische Selbstoptimierung der Recommendations ein ausführliches Data Cleaning des Warehouses. Insbesondere durch den Closed-Loop-Ansatz der Optimierung können sich Datenfehler verstärken, da sie – wie oben beschrieben – die Berechnung und damit die Qualität der Recommendations beeinträchtigen und damit – bei automatischer Recommender-Selektion – auch die adaptive Auswahl der Recommender beeinflussen. Fehlerhaftes Recommendation-Feedback, das z. B. das „Anklicken“ von Recommendations durch Suchmaschinen-Crawler als Akzeptieren und daher positives Feedback wertet, verstärkt den negativen Einfluss auf die Optimierung.

Für das Web Data Warehouse steht die Verarbeitung der Logfiles im Vordergrund. Der AWESOME-Ansatz setzt zur Erzielung einer optimalen Datenqualität auf eine Kombination der bisher in der Literatur vorgeschlagenen Verfahren (siehe [40] für eine Übersicht) zur Aufbereitung der Webdaten. Dabei stehen insbesondere die Crawler-Erkennung, d. h. die Erkennung aller nicht von menschlichen Benutzern erzeugten Website-Zugriffe, und die Session-Erkennung, d. h. die Zusammenfassung von Pageviews des gleichen Website-Besuchs, im Vordergrund.

3.4.1 Crawler-Erkennung

Crawler-Erkennung beschäftigt sich mit der Identifikation von Webzugriffen, die nicht von menschlichen Benutzern im Rahmen eines Website-Besuchs entstehen. Es lassen sich u. a. folgende Arten von automatischen Zugriffen unterscheiden:

Suchmaschinen-Crawler erfassen die Inhalte ganzer Websites automatisch und speichern sie auf Servern der entsprechenden Suchmaschine [145]. Dort werden die Inhalte indiziert, damit sie bei entsprechenden Suchanfragen gefunden werden. Um stets aktuelle Inhalte vorzuhalten, wird das Crawling regelmäßig wiederholt.

Website-Downloader speichern eine komplette Website oder einen Teil davon auf den Rechner eines Benutzers, damit dieser die Inhalte offline zur Verfügung hat. So können z. B. bei universitären Websites die kompletten Vorlesungsunterlagen (Skripte, Übungsaufgaben) eines Semesters gespeichert werden.

Referrer-Spam bezeichnet den wiederholten Zugriff auf eine Seite (z. B. die Startseite der Website) mit einem manipulierten Referrer, der auf eine zu bewerbende Website verweist. Der Hintergrund ist, dass automatische Analyseprogramme für den Webzugriff (z. B. Webalizer[20]) Logfiles auswerten und u. a. eine Liste der häufigsten Referrer generieren. Werden die Ergebnisse der Analyse selbst auf der Website veröffentlicht, enthalten sie dadurch Links auf die entsprechenden URLs der Referrer, d. h. die beworbenen Seiten. Da solche Links ein Kriterium für das Ranking von Websites bei Suchmaschinen sind [23], wird Referrer-Spam genutzt, um eine URL möglichst gut bei Suchmaschinen zu platzieren.

Die Entfernung solcher Zugriffe ist sowohl wichtig für die Evaluation des Webnutzungsverhaltens als auch für die Berechnung von Recommendations. Bleiben bei der Evaluation des Webnutzungsverhaltens viele automatische Zugriffe unerkannt, kann z. B. die Analyse, aus welchen Ländern die Website-Nutzer stammen, durch Suchmaschinen-Crawler verfälscht werden. Des Weiteren werden die Recommendations der Recommender, die das Nutzungsverhalten auswerten und z. B. die am häufigsten aufgerufene Seite empfehlen, durch automatische Seitenaufrufe bei Referrer-Spam verfälscht, was zu einer Qualitätsminderung der Recommendations führen kann.

Grundproblem der Crawler-Erkennung ist das Fehlen eines hundertprozentigen sicheren Verfahrens zur Unterscheidung zwischen Crawler-Zugriffen und menschlichen Zugriffen. Daher werden Heuristiken angewendet, die typische Eigenschaften der Crawler ausnutzen. Innerhalb von AWESOME wird dabei folgende Kombination von Heuristiken angewendet:

IP-Adressen: Es wird manuell eine Liste von IP-Adressen (bzw. IP-Adressbereichen) gepflegt, die als Crawler bekannt sind. Darunter fallen u. a. die Server der großen Suchmaschinen, z. B. `crawl1.googlebot.com` für die Suchmaschine Google.

Agent-Namen: Website-Downloader identifizieren sich meist über einen entsprechenden Eintrag im Agent-Feld des HTTP-Protokolls. Mit veröffentlichten Listen der Agent-Namen solcher Downloader lassen sich deren Zugriffe herausfiltern.

Bot-Trap: Da das Verhalten der Crawler im Wesentlichen bekannt ist, kann die Website durch „Fallen“ manipuliert werden. Ein Beispiel für eine solche Falle

[20] http://www.mrunix.net/webalizer/

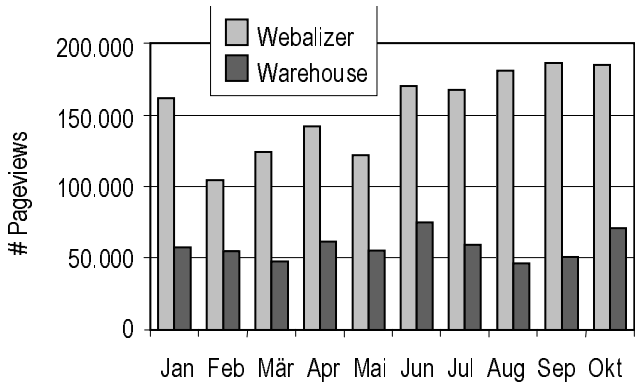


Abbildung 3.3: Anzahl der monatlichen Pageviews (2004)

wäre ein für Menschen „unsichtbarer“ Link (z. B. weiße Schrift auf weißem Hintergrund), der auf eine definierte Seite S verweist und in jeden Pageview integriert wird. Da Crawler den Quellcode der Seiten auswerten, werten sie den für Menschen „unsichtbaren“ Link als normalen Link. Folgt ein Crawler allen auf einer Seite aufgeführten Links, können Sessions mit einem Zugriff auf diese Seite S als automatischer Zugriff identifiziert werden.

Nutzungskennzahlen: Das Data Warehouse bietet die Möglichkeit, mit Anfragen einfache statistische Kennzahlen für das Nutzungsverhalten zu bestimmen. Beispiele sind die Gesamtzahl der aufgerufenen Seiten einer Session, die durchschnittliche Zeit zwischen zwei Seitenaufrufen oder die Anzahl der Anfragen mit gleichem Referrer. Übersteigen bzw. unterschreiten diese Kennzahlen (manuell) definierte Grenzwerte, z. B. mehr als 100 Seitenaufrufe pro Session oder weniger als eine Sekunde zwischen zwei Anfragen, können die Sessions Crawlern zugeordnet werden (siehe auch [185]).

Der Einfluss der Crawler-Erkennung wird anhand des freien Analyseprogramms Webalizer illustriert, welches in der verwendeten Version keine Crawler-Erkennung durchführt. Abbildung 3.3 vergleicht die Anzahl der von Webalizer protokollierten Zugriffe mit denen von AWESOME nach der Durchführung der implementierten Crawler-Erkennung für die Beispiel-Website (siehe Abschnitt 3.6). Die Grafik zeigt, dass ca. 70% aller Zugriffe als nicht von menschlichen Benutzern stammend identifiziert wurden. Dies verdeutlicht den hohen Stellenwert der Crawler-Erkennung für die Datenqualität. Fehlende oder ungenügende Crawler-Erkennung führt zu massiven Verfälschungen in der Weiterverarbeitung des aufgezeichneten Webnutzungsverhaltens.

P-ID	S-ID	Client	Date Time	Page *Referrer*	Session
1	–	cw12.H1.srv.t-online.de	27.01.2004 01:15:02	/DBS1/ch4-1.html *www.google.com/search?q=sql+statements*	A
2	–	pD9E1621B.dip.t-dialin.net	27.01.2004 01:15:10	/ADS1/ch3-5.html *www.google.com/search?q=linked+list*	B
3	1	cop7001.cpmc.columbia.edu	27.01.2004 01:15:12	/People/rahm.html *research.microsoft.com/ philbe/*	C
4	–	cw12.H1.srv.t-online.de	27.01.2004 01:15:20	/DBS1/ch4-2.html */DBS1/ch4-1.html*	A
5	1	cop7001.cpmc.columbia.edu	27.01.2004 01:15:21	/index.html */People/rahm.html*	C
6	–	cw08.H1.srv.t-online.de	27.01.2004 01:15:23	/DBS1/ch4-3.html */DBS1/ch4-2.html*	A
7	–	pD9E1621B.dip.t-dialin.net	27.01.2004 01:15:43	/ADS1/ch3-4.html */ADS1/ch3-5.html*	B

Tabelle 3.3: Auszug aus dem Web-Logfile mit drei Sessions (ohne User-ID)

3.4.2 Session-Erkennung

Bei der Session-Erkennung werden die Pageviews eines Nutzers innerhalb eines Website-Besuchs gruppiert. Session-Erkennung ist sowohl nötig, um das Webnutzungsverhalten zu analysieren, z. B. welche Seiten innerhalb eines Website-Besuchs gemeinsam betrachtet wurden, als auch um die Effektivität der Recommendations zu ermitteln, d. h. ob empfohlene Seiten innerhalb einer Session aufgerufen wurden.

Bezüglich der Terminologie in [181] verwendet AWESOME sowohl eine proaktive als auch eine reaktive Strategie zur Session-Erkennung.

Proaktive Ansätze versuchen, bereits während des Aufrufs einer Seite den Pageview einer Session zuzuordnen. Eine Möglichkeit dabei sind temporäre Cookies, um auf dem Client-Rechner eine automatisch generierte Session-ID zu speichern.

Reaktive Verfahren analysieren das Webnutzungs-Logfile und wenden – analog zur Crawler-Erkennung – Heuristiken an, um Pageviews der gleichen Session zu erkennen [40]. AWESOME unterstützt dabei eine Kombination verschiedener Heuristiken. Als ein erstes Verfahren werden der Host-Name und der Agent-Name als temporärer Identifier verwendet. Um mit variierenden IP-Adressen umzugehen, die u. a. bei der Benutzung von Proxy-Servern auftreten, werden lediglich die letzten (vier) Teile des Host-Namens verwendet. Alle Zugriffe dieses Identifiers innerhalb eines definierten Zeitfensters, z. B. von 30 Minuten, werden als eine Session behandelt. Zusätzlich wird noch die Referrer-Information ausgenutzt, um gegebenenfalls mehrere Sessions mit dem gleichen Identifier zu trennen.

AWESOME unterstützt die Kombination proaktiver und reaktiver Strategien. Dabei wird zunächst versucht, eine proaktive Strategie durchzuführen, was im Beispiel von Tabelle 3.3 für Session C gelingt. Für Pageviews, bei denen das nicht möglich ist (z. B. weil keine Cookies zugelassen wurden), wird innerhalb einer reaktiven Strategie

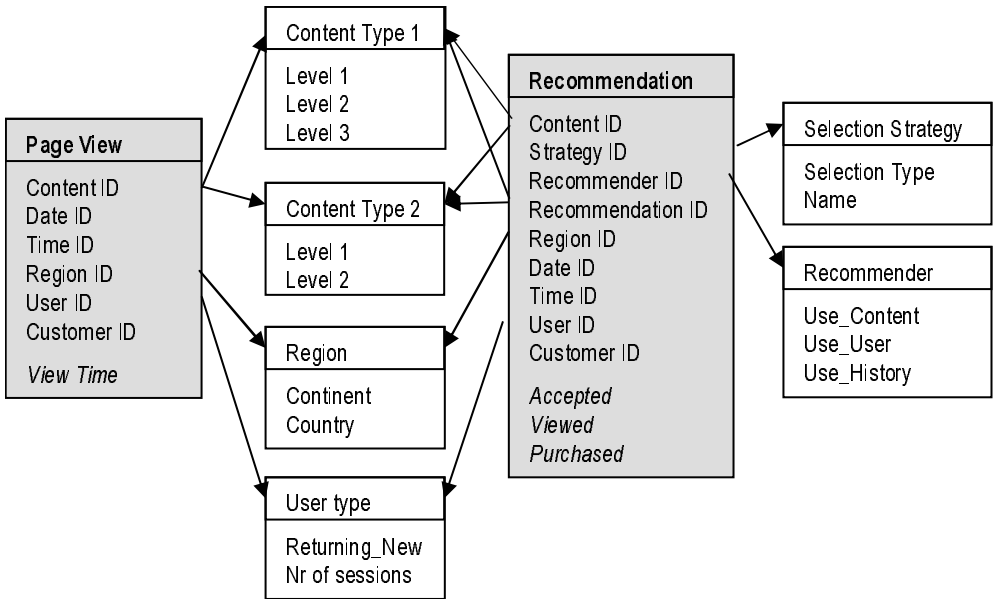


Abbildung 3.4: Vereinfachte Darstellung des Data-Warehouse-Galaxien-Schemas

eine Kombination von Heuristiken durchgeführt. Dadurch lassen sich im Beispiel bei den verbleibenden Pageviews die Sessions A und B differenzieren.

3.5 Data-Warehouse-Design

Das Data Warehouse ist eine relationale Datenbank mit einem Galaxien-Schema, welches Fakten- und Dimensionstabellen enthält [30]. Eine vereinfachte Darstellung zeigt Abbildung 3.4, welche die beiden Faktentabellen Pageview und Recommendation sowie ausgewählte Dimensionstabellen illustriert. Das konkrete Design des Data Warehouses hängt von der verwendeten Website und den zur Verfügung stehenden Datenquellen ab. Im Folgenden werden daher die typischen, d. h. Website-unabhängigen, Teile des Schemas vorgestellt.

Ein Eintrag der Faktentabelle *Pageview* repräsentiert einen Pageview. Soll in der späteren Analyse des Webnutzungsverhaltens auch die Zeitdauer der Pageviews betrachtet werden, so kann als Maß für die Faktentabelle die *View Time* verwendet werden. Sie errechnet sich als die Zeitdifferenz zwischen dem Request der Seite und der in der Session nachfolgenden Seite.[21]

Jeder Pageview wird mit Hilfe der Dimensionstabellen charakterisiert. Die konkrete Ausgestaltung der Dimensionen hängt von der verwendeten Website sowie den verfügbaren Datenquellen ab. Sind z. B. – wie in Abschnitt 3.3 beschrieben – Kategorisierungen für die einzelnen Seiten verfügbar, lässt sich das – wie in Abbildung 3.4

[21] Offenbar kann für den letzten Pageview einer Session keine View Time bestimmt werden.

dargestellt – z. B. durch entsprechende (im Beispiel: zwei) *Content-Type*-Dimensionstabellen realisieren. Für die Beschreibung der Nutzer lassen sich z. B. regionale Informationen verwenden (*Region*), die z. B. aus der IP-Adresse des Nutzers gewonnen werden können. Wird eine Wiedererkennung der Nutzer durchgeführt, können die Informationen durch eine Dimensionstabelle Nutzertyp (*User Type*) erfasst werden.[22] Dabei kann zwischen neuen und wiederkehrenden Nutzern unterschieden und insbesondere die Anzahl der früheren Website-Besuche gespeichert werden. Dadurch lässt sich z. B. später das Webnutzungsverhalten von neuen und wiederkehrenden Nutzern miteinander vergleichen.

Jede präsentierte Recommendation wird als Eintrag in der Faktentabelle *Recommendation* erfasst. Als Maß dient hierbei die Information, ob die Empfehlungen akzeptiert (d. h. angeklickt) wurde (*Accepted*) oder zumindest im Verlauf der Session aufgerufen wurde (*Viewed*).[23] Durch diese Maße wird sowohl positives als auch negatives Recommendation-Feedback repräsentiert. Analog zur Pageview-Faktentabelle wird sowohl die empfohlene Seite als auch die Seite, auf der die Empfehlung präsentiert wurde, durch die Dimensionen charakterisiert. Zusätzlich definiert die Dimension *Recommender* das Verfahren, welches die Empfehlung berechnete. Recommender-Klassifikationen, wie z. B. die in Abschnitt 2.3 vorgestellte, können dabei zur weiteren Beschreibung der Dimension dienen. Wie in Abschnitt 3.2 erläutert, unterstützt AWESOME die dynamische Selektion von Recommendern durch sogenannte Selektionsstrategien. Um die Recommender-Selektion später evaluieren zu können, wird die angewendete *Strategie* ebenfalls in einer Dimensionstabelle ablegt.

3.6 Vorgehensweise bei der Implementation

AWESOME ist ein generischer Ansatz für ein hybrides Recommender-System, dessen Recommendations evaluiert und optimiert werden können. Daher gibt es nicht *die* Implementation, sondern AWESOME wird für jede bestehende Website realisiert. Im diesem Abschnitt werden daher die grundlegenden Teile vorgestellt, welche – spezifisch für die jeweilige Website – implementiert werden müssen. Parallel dazu wird eine Beispielimplementation vorgestellt, die als Website die Homepage[24] der Abteilung Datenbanken verwendet.[25]

Die Implementation von AWESOME umfasst im Wesentlichen drei Bereiche:

[22] Ist bei kommerziellen Websites der Nutzer auch als Kunde erfasst, können die ihm zugeordneten Daten über weitere Dimensionstabellen zugeordnet werden.

[23] Bei kommerziellen Websites kann zusätzlich erfasst werden, ob das empfohlene Produkt gekauft wurde oder nicht.

[24] http://dbs.uni-leipzig.de

[25] Auf Grund einer Neugestaltung der Website, u. a. durch Umstellung auf ein Content-Management-System, sind die Recommendations in der aktuellen Website der Abteilung nicht mehr verfügbar.

Data Warehouse: Neben dem Design des Data Warehouses, das u. a. durch die Website-abhängigen Datenquellen definiert wird (siehe Abschnitt 3.5), steht bei der Implementation der ETL-Prozess im Vordergrund. Dabei müssen die in den verschiedenen Datenquellen (u. a. Logfiles) verfügbaren Informationen in einem automatischen – und periodisch ausführbaren – Prozess in das Warehouse eingebracht werden.

Recommender-Bibliothek: Die Realisierung der verschiedenen Recommender kennzeichnet den zweiten Bereich der AWESOME-Implementation. Dabei müssen die unterschiedlichen Verfahren mit einer fest definierten Schnittstelle implementiert werden, so dass ein einheitlicher Zugriff auf die Recommender möglich ist. Dabei soll die Schnittstelle aus Performanzgründen sowohl die Offline-Vorberechnung der Recommendations als auch die Online-Abfrage der Recommendations für einen definierten Kontext ermöglichen.

Website: Der letzte Bereich umfasst die Erweiterung der bestehenden Website um die von AWESOME bereitgestellten Funktionalitäten. Dazu muss eine Layout-Änderung erfolgen, so dass die Recommendations in einem Bereich des Bildschirms angezeigt werden. Dabei muss im Applikations-Server die Bestimmung der aktuellen Recommendations bei jedem Seitenaufruf angestoßen werden. Gleichzeitig muss die Website um die Funktionalität zur Erstellung der Logfiles, d. h. Webnutzungs- sowie Recommendation-Logfile, erweitert werden.

In den nachfolgenden Abschnitten werden die drei beschriebenen Bereiche für die im Rahmen der vorliegenden Arbeit durchgeführte prototypische Implementation skizziert.

3.6.1 Data Warehouse

Das Data Warehouse wurde mit Hilfe des Microsoft SQL Servers 2000 realisiert. Neben der Definition der Fakten- und Dimensionstabellen lag der Schwerpunkt auf der Implementierung des Datenimports. Dazu wurden Data Transformation Services (DTS) verwendet, welche die Realisierung eines Workflows in Form von Packages unterstützen.

Das DTS-Package beginnt mit dem Download der aktuellen Logfiles vom Webserver, welche zunächst „eins zu eins“ in temporäre Tabellen übertragen werden. Anschließend erfolgt eine Vielzahl von Transformationen, die meist mittels SQL-Statements realisiert wurden. Dabei wurde mit Transact-SQL auf eine SQL-Erweiterung zurückgegriffen, die neben dem SQL-Standard auch Elemente prozeduraler Programmiersprachen (z. B. IF-THEN-ELSE-Bedingungen und WHILE-Schleifen) bietet. Dies ermöglicht auch die Umsetzung komplexer Datentransformationen wie z. B. der Session-Erkennung. Abschließend werden die gesäuberten Daten in den Fakten- und Dimensionstabellen gespeichert.

Das gesamte Package wird automatisch mit Hilfe eines Schedulers einmal täglich ausgeführt und erlaubt daher eine vollautomatische Aktualisierung des Data Warehouses.

3.6.2 Recommender-Bibliothek

Eine Übersicht über die implementierten Recommender zeigt Tabelle 3.4. Die Berechnung der Recommendations wurde ebenfalls durch DTS-Packages realisiert. Da bei der verwendeten Beispiel-Website keine weiteren Informationen zu Nutzern und Inhalten (außer zwei Kategorisierungen sowie der textuellen Beschreibung als HTML-Dokument) vorliegen, treten bei den Recommendern hauptsächlich Verfahren auf Basis des Webnutzungsverhaltens auf. Einfache statistische Berechnungsverfahren (z. B. Anzahl der Pageviews) wurden mit entsprechenden SQL-Anweisungen realisiert. Für komplexere Berechnungen, z. B. die Bestimmung von Assoziationsregeln für häufige innerhalb einer Session gemeinsam auftretende Seiten oder die Berechnung der Ähnlichkeit zweier Seiten mittels TF·IDF, wurde auf externe Programme zurückgegriffen, die innerhalb des DTS-Packages aufgerufen wurden. Wie in Tabelle 3.4 abzulesen ist, unterscheiden sich viele Recommender nur in einzelnen Parametern, z. B. dem betrachteten Zeitraum des Webnutzungsverhaltens. Dadurch wird später ein Recommender-Tuning ermöglicht (siehe Abschnitt 5.3.2), indem z. B. die Qualität der Recommendations der Top-Recommender für verschiedene Zeitfenster miteinander verglichen werden.

Besonders erwähnenswert ist der Suchmaschinen-Recommender (SER). Bei Nutzern, die von einer Suchmaschine die Website erreichen (z. B. von Google), enthält der Referrer die URL der Suchergebnisseite, innerhalb derer ein Suchergebnis angeklickt wurde. Diese URL enthält i. Allg. auch die Schlagworte, welche der Nutzer zuvor bei der Suchmaschine eingegeben hat. Der SER-Recommender extrahiert diese Schlagworte aus dem Referrer und verwendet sie als Suchanfrage für die in die Website integrierte Suchmaschine htDig[26]. Die Ergebnisse dieser Suche werden als Recommendations dem Nutzer präsentiert (siehe auch [109]).

Für die Recommender-Bibliothek wurde ein modulares Konzept verwendet, welches sowohl die periodische Vorberechnung der Recommendations als auch die Abfrage von Recommendations unterstützt. Abbildung 3.5 verdeutlicht diesen Ansatz, der insbesondere ein einfaches Hinzufügen neuer Verfahren ermöglicht. Für jeden Recommender werden im Warehouse Informationen zu seiner Ausführung gespeichert. Dies betrifft sowohl die Vorberechnung der Recommendations als auch die Bestimmung der aktuellen Empfehlungen.

Die Vorberechnung der Empfehlungen wird – wie bereits oben erwähnt – durch DTS Packages realisiert. Innerhalb des Packages können nun Anfragen an das Data

[26] http://www.htdig.org

Typ	Name	Verfahren	Variationen	#
(–,–,–)	New	neue bzw. kürzlich aktualisierte Seiten	–	1
(–,–,+)	Top	Seiten mit den meisten Pageviews	Zeit: 1, 7, 28 Tage	3
(–,–,+)	TopInc	Seiten mit stärkstem Anstieg in der Anzahl der Pageviews	Zeit: 1, 7, 28 Tage	3
(–,+,–)	SER	Verwenden der Suchbegriffe im Referrer (falls Nutzer von Suchmaschine kommt) für Suche nach passenden Seiten	–	1
(–,+,+)	PersInt	wie Top; eingeschränkt auf Seitenkategorien, auf die beim letzten Besuch die meisten Pageviews entfielen	Verfahren: Top, TopInc; Zeit: 1, 7, 28 Tage; Kategorien: 2 Kategorisierungen mit je zwei Ebenen	24
(+,–,–)	Sim	Ähnlichkeit zur aktuellen Seite mit TF·IDF	–	1
(+,–,+)	Asso	Assoziationsregeln, d. h. welche Seiten treten häufig mit aktueller Seite in gleicher Session auf	Seite: Direct (nur als direkter Nachfolger), Successor (als Nachfolger in Session), Session (in gleicher Session); Zeit: 1, 7, 28 Tage	9
(+,–,+)	TopCat	wie Top; eingeschränkt auf Seitenkategorien der aktuellen Seite	Verfahren: Top, TopInc; Zeit: 1, 7, 28 Tage; Kategorien: 2 Kategorisierungen mit je zwei Ebenen	24
(+,+,–)	NewCat	wie New; eingeschränkt auf Seitenkategorien, auf die beim letzten Besuch die meisten Pageviews entfielen	Kategorien: 2 Kategorisierungen mit je zwei Ebenen	3
(+,+,+)	CF	Bestimmung ähnlicher Nutzer, d. h. mit gleichen Seitenkategorien beim letzten Besuch wie aktueller Benutzer, und Anwenden von Asso eingeschränkt auf das Nutzungsverhalten der ähnlichen Nutzer	Kategorien: 2 Kategorisierungen mit je zwei Ebenen; Seite (wie bei Asso): Direct, Successor, Session; Zeit: 1, 7, 28 Tage	36

Tabelle 3.4: Liste der implementierten Recommender

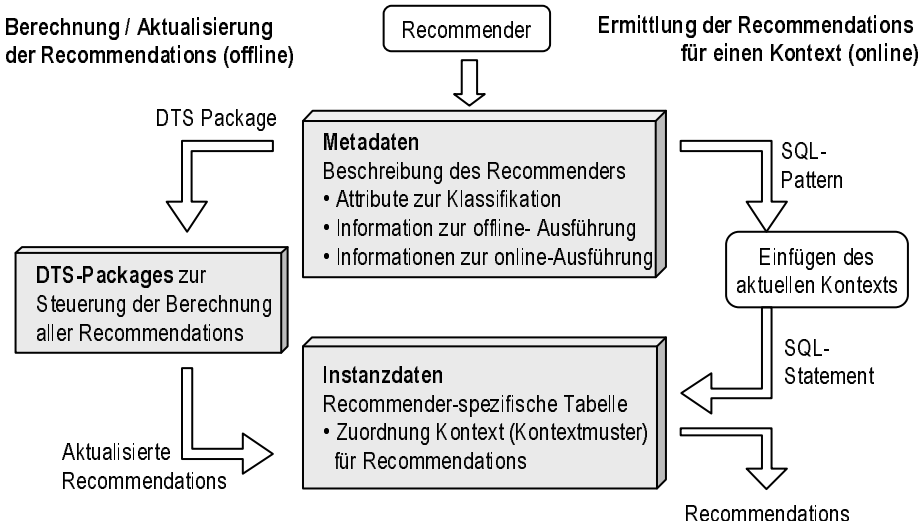


Abbildung 3.5: Schematische Darstellung der modularen Implementierung der Recommender-Bibliothek

Warehouse gestellt werden, z. B. mit der MS-SQL-Abfragesprache MDX. Des Weiteren ist auch die Verwendung externer Programme möglich. So können z. B. Java-Programme ausgeführt werden, innerhalb derer auf Funktionalitäten frei verfügbarer Bibliotheken (z. B. der WEKA-Bibliothek für Data-Mining-Algorithmen [205]) zurückgegriffen werden kann.

Nach der Ausführung des Packages liegen alle Empfehlungen in einer durch den Recommender erzeugten Look-up-Tabelle vor. Aus Gründen der Performanz kann sowohl die Berechnung der Recommendations als auch das Vorhalten der Ergebnisse in Look-up-Tabellen für mehrere Recommender zusammengefasst werden. Einfache statistische Verfahren, z. B. der Top-Recommender, können so gemeinsam für verschiedene Parameter, z. B. das Zeitfenster des verwendeten Nutzungsverhaltens, ausgeführt und gespeichert werden. Nach der Berechnung aller Recommendations werden sämtliche Look-up-Tabellen auf einen eigenen Server mit einer mySQL-Datenbank kopiert. Dadurch wird die Online-Anfrage der Recommendations von der Berechnung getrennt.

In der Metabeschreibung der Recommender wird zusätzlich festgehalten, wie die Online-Bestimmung der Recommendations durchgeführt wird. Dazu wird jedem Recommender (definiert durch seine ID) ein parametrisiertes SQL-Statement zugeordnet, wobei die Parameterliste alle Kontextinformationen umfasst. Das SQL-Statement wird für den aktuellen Kontext ausgeführt. Dabei ist es nicht notwendig, dass alle Kontextattribute als Parameter auch für die Berechnung benötigt werden. Lediglich aus Gründen der Modularität werden alle Kontextattribute als Parameter übergeben. Das resultierende SQL-Statement wird nun ausgeführt und liefert eine geordnete Liste der Recommendations.

Name	Asso-Session-7
Typ	(+, –, +)
Offline	DTS_Assoziationsregeln
Online	SELECT Rec FROM Asso WHERE Type = 'Session' AND TimeWin = 7 AND Page = <Context_Page> ORDER BY Rating DESC

Asso				
Type	TimeWin	Page	Rec	Rating
Direct	7	A	B	2
Direct	7	A	C	1
Direct	28	A	B	2
Direct	28	A	D	1
Session	7	A	E	2
Session	7	A	F	1
Session	28	A	B	2
Session	28	A	G	1

Abbildung 3.6: Darstellung der Metadaten (links) sowie eines (vereinfachten) Ausschnitts aus der Look-up-Tabelle für einen Assoziationsregeln-Recommender

Durch die beschriebene Vorgehensweise wird die Funktionsweise des Recommenders (innerhalb des DTS-Packages) gekapselt. Ein neuer Recommender muss lediglich die Schnittstellen für das Aktualisieren/Vorberechnen der Recommendations und das Online-Abfragen bereitstellen.

Das vorgestellte modulare Konzept soll abschließend an einem Recommender des Typs *Asso*, der häufig gemeinsam vorkommende Seiten einer Session (Type=Session) der letzten 7 Tage (TimeWin=7) ermittelt, illustriert werden. Abbildung 3.6 (links) zeigt die Metadaten des Recommenders. Neben seinem (eindeutigen) Namen wird der Klassifikationstyp angegeben, der die Werte der Recommender-Dimensionen im Warehouse definiert und insbesondere für die spätere Evaluation mittels OLAP-Analyse verwendet wird. Für die Vorberechnung und Aktualisierung der Recommendations wird ein DTS-Package angegeben. Da die Berechnung von Recommendations für Recommender, die sich nur in einzelnen Parametern unterscheiden, z. B. dem Zeitfenster des verwendeten Webnutzungsverhaltens, praktischerweise gemeinsam erfolgen kann, kann das gleiche DTS-Package bei mehreren Recommendern angegeben werden. Der letzte Teil der Metadaten beschreibt das SQL-Pattern, mit welchem für einen gegebenen Kontext die Recommendations ermittelt werden können. Das Beispiel greift auf die (im DTS-Package erstellte und in Abbildung 3.6 (rechts) dargestellte) Tabelle `Asso` zu. Da diese Tabelle Recommendations mehrerer Recommender enthält, werden mittels Bedingungsprüfung auf `Type` und `TimeWin` nur die Recommendations des Recommenders betrachtet. Der Bedingung `Page = <Context_Page>` repräsentiert die Abhängigkeit der Recommendations von der aktuellen Seite des Nutzers (bzw. des Kontexts). Bei der Bestimmung der Recommendations wird `<Context_Page>` durch die aktuelle Seite ersetzt, so dass ein ausführbares SQL-Statement entsteht, dessen Ergebnis die gesuchten Recommendations enthält. Die Sortierung nach dem Attribut `Rating`, welches die „Güte“ der Recommendation bzgl. des jeweiligen Berechnungsmodells widerspiegelt, sichert eine definierte Reihenfolge der Recommendations, so dass die beste Recommendation zu

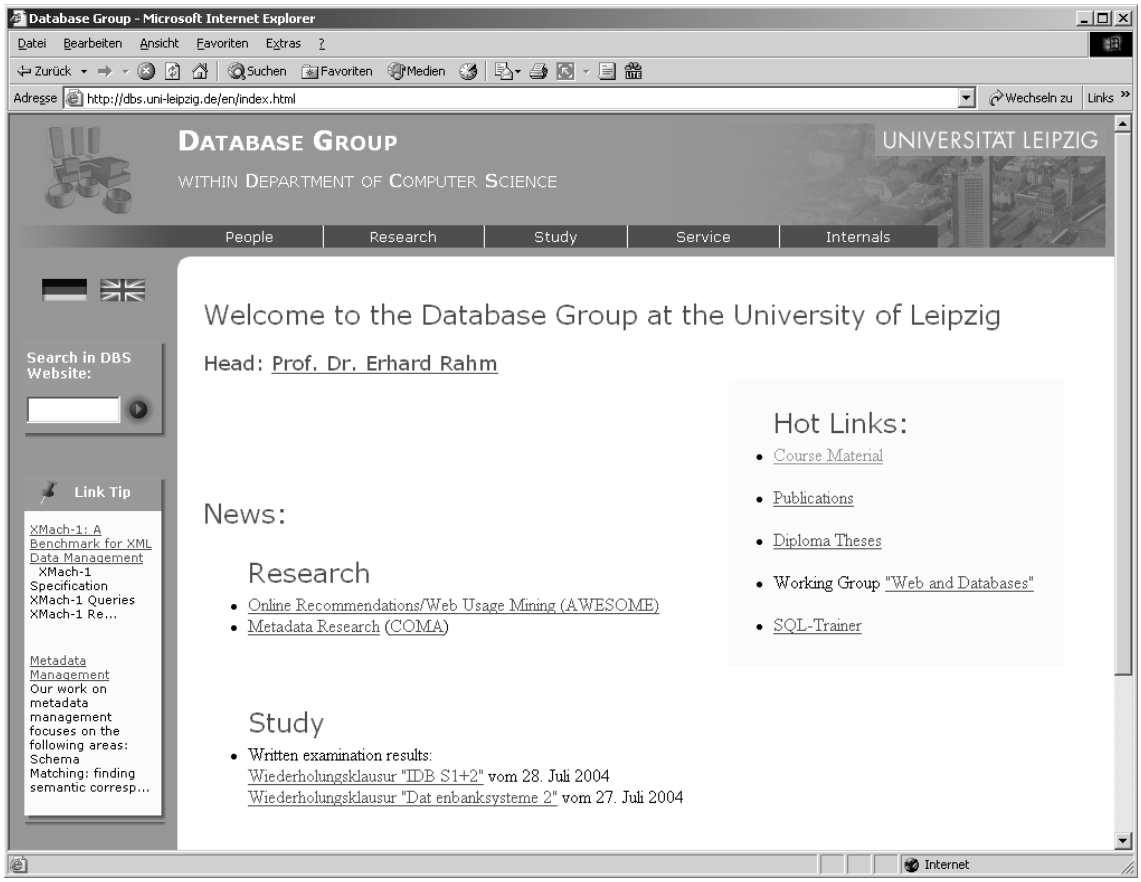


Abbildung 3.7: Screenshot der Website des AWESOME-Prototypen (August 2004)

erst erscheint, die zweitbeste als zweites usw. Im Beispiel der Assoziationsregeln kann die Güte z. B. aus der Konfidenz der Assoziationsregel abgeleitet sein; bei einem Top-Recommender kann sie sich an der Anzahl der Pageviews orientieren. Für die aktuelle Seite `A` würde der Recommender im Beispiel von Abbildung 3.6 die Recommendations `E` und `F` ermitteln.

3.6.3 Website

AWESOME lässt sich für alle Websites einsetzen, die einen Applikations-Server verwenden, mit dessen Hilfe sich das Layout der einzelnen Seiten dynamisch anpassen lässt. Für die Implementation von AWESOME wurde als Website die Homepage der Abteilung Datenbanken gewählt. Als Webserver kommt Apache zum Einsatz und die dynamische Gestaltung der Webseiten wird mit PHP realisiert. Einen Screenshot der Website zeigt Abbildung 3.7. Die dargestellten Recommendations befinden sich am linken Rand im Kasten „Link Tip“.

Die Erweiterung einer bestehenden Website um dynamische Recommendations besteht aus drei Teilen, die im Folgenden aufgeführt sind. Dabei wird zusätzlich kurz die konkrete Umsetzung bei der verwendeten Beispiel-Website skizziert.

Layout-Anpassung: Das Layout der einzelnen Seiten muss um die Darstellung der Recommendations erweitert werden. In der Beispielimplementation wurde ein entsprechender Bereich am linken Rand eingefügt (siehe Abbildung 3.7), innerhalb dessen die Empfehlungen mit Titel (inklusive Link zur Seite) und kurzer Vorschau (d. h. die ersten Zeichen des Inhalts) angezeigt werden. Zusätzlich wird auf jeder Seite ein „versteckter" Link zu einer neuen Seite `bottrap.html` eingebaut, der die Qualität der automatischen Crawler-Erkennung verbessert (siehe Abschnitt 3.4.1).

Logfile-Generierung: Der Applikations-Server muss um die Generierung von Log-Daten erweitert werden, so dass neben den einzelnen Pageviews auch alle präsentierten Recommendations – inklusive der Information, ob sie angeklickt wurden oder nicht – gespeichert werden. In der Beispielimplementation wurden mittels PHP zwei Logfiles erstellt. Das Webnutzungs-Logfile enthält die Liste aller Pageviews inklusive der im Applikationsserver verfügbaren Kontextinformationen. Für die Verbesserung der Session-Erkennung wird eine Session-ID innerhalb eines temporären Cookies auf dem Client-Rechner gespeichert. Analog wird die Nutzerwiederkennung über eine User-ID mit Hilfe eines permanenten Cookies realisiert. Akzeptiert der Nutzer keine permanenten Cookies oder hat er seine Cookies seit dem letzten Besuch gelöscht, wird in der Beispielimplementation keine Nutzererkennung durchgeführt werden. Dem Nutzer wird dann eine neue ID zugewiesen. Im Recommendation-Logfile wird für jede präsentierte Recommendation gespeichert, welcher Recommender sie generiert hat, bei welchem Pageview sie angezeigt wurde und ob sie im Verlauf der Session angeklickt wurde.

Bestimmung der Recommendations: Bei jedem Seitenaufruf muss die Bestimmung der darzustellenden Recommendations angestoßen werden. Innerhalb der Beispiel-Website wurden dazu zwei darzustellende Recommendations bestimmt. Ein an die Bestimmung der Recommendations angeschlossener Filter stellt sicher, dass weder die Startseite[27] noch die aktuell aufgerufene Seite als Recommendation dargestellt werden, da solche Empfehlungen als unsinnig angesehen werden. Zusätzlich wird innerhalb der Session sichergestellt, dass beim Wiederkehren zu einer bereits aufgerufenen Seite dieselben Recommendations erneut angezeigt werden. Dadurch soll vermieden werden, dass Nutzer durch wechselnde Inhalte auf der gleichen Seite verwirrt werden.

[27] Die Startseite konnte auf der Beispiel-Website von jeder Seite durch das Anklicken eines Logos erreicht werden.

3.7 Zusammenfassung

Mit AWESOME wurde in diesem Kapitel ein flexibler Ansatz zur Erstellung, Evaluation und Optimierung von Recommendations vorgestellt. Der AWESOME-Ansatz kann für beliebige Websites eingesetzt werden, da er auf Grund seines Data Warehouses flexibel gegenüber den verwendeten, d. h. für die Website relevanten, Daten ist und gleichzeitig durch die Recommender-Bibliothek den Einsatz beliebiger Berechnungsverfahren zur Bestimmung von Recommendations unterstützt. Die vorgestellte Architektur bildet sowohl die Grundlage für eine adaptive Auswahl der besten Recommendations (Kapitel 4) als auch für eine flexible Evaluation der Recommendation-Qualität (Kapitel 5).

4

Adaptive Recommendations

In diesem Kapitel wird ein regelbasierter Ansatz zur Bestimmung adaptiver Recommendations vorgestellt. Dazu werden sogenannte Selektionsregeln eingeführt, welche für einen Kontext den anzuwendenden Recommender definieren (Abschnitt 4.1). Neben der manuellen Erstellung solcher Regeln liegt der Schwerpunkt auf der automatischen Regelgenerierung (Abschnitt 4.2). Es werden dazu u. a. zwei Ansätze vorgestellt, die das aufgezeichnete Nutzer-Feedback automatisch in Regeln transformieren und so eine automatische Selbstoptimierung der Recommendation-Qualität ermöglichen.

4.1 Regelbasierte Recommender-Selektion

Die Bestimmung adaptiver Recommendations setzt ein Verfahren voraus, welches für einen aktuellen Kontext, d. h. dem Kontext beim Anfordern einer Seite durch einen Nutzer, die vielversprechendsten Recommendations bestimmt. Die Vorgehensweise, welche die Bestimmung adaptiver Recommendations definiert, wird im Folgenden Adaptionswissen genannt. Abbildung 4.1 illustriert dabei den Unterschied zwischen dem aufgezeichneten Recommendation-Feedback und dem Adaptionswissen. Das Feedback repräsentiert die in der *Vergangenheit* erzielten Ergebnisse der Recommendations, d. h. welche Recommendations angezeigt und angeklickt wurden. Das Adaptionswissen repräsentiert die Bestimmung von Recommendations für *zukünftige* Seitenaufrufe, bei denen Recommendations präsentiert werden sollen. Eine wesentliche Schwierigkeit besteht in der effizienten Verwendung des Recommenda-

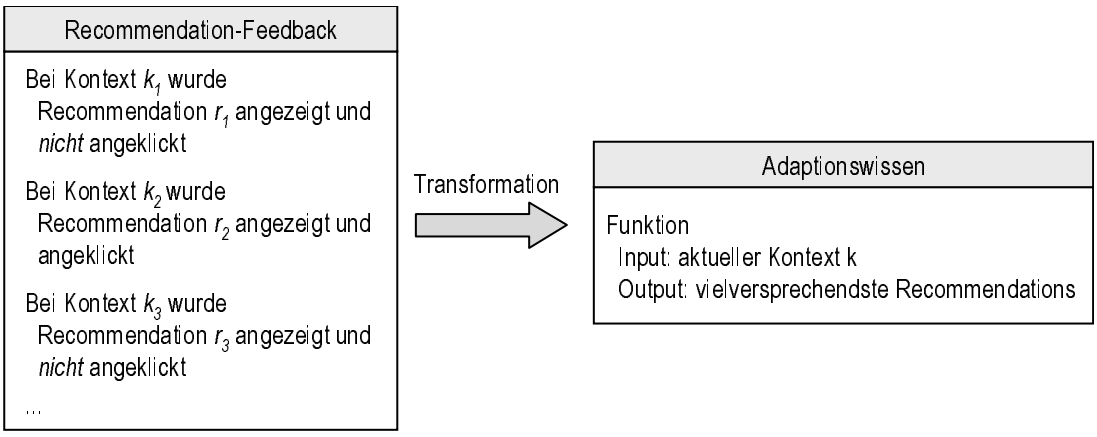


Abbildung 4.1: Schematische Darstellung des Recommendation-Feedbacks (links) und des Adaptionswissens (rechts)

tion-Feedbacks zur Erstellung von Adaptionswissen.

Unabhängig von der konkreten Website, d. h. dem konkreten „Inhalt“ des Adaptionswissens, sowie der Art und Weise der Erstellung werden folgende Anforderungen an das Adaptionswissen gestellt.

Manuelle und automatische Adaption: Die Generierung des Adaptionswissens soll sowohl manuell als auch automatisch realisierbar sein. Zum einen soll der Website-Administrator auf Basis seines Wissens über die Website und deren Nutzer die Möglichkeit haben, die Auswahl der Recommendations zu beeinflussen. Auf der anderen Seite soll durch verschiedene automatische Verfahren, welche das aufgezeichnete Feedback zur Generierung adaptiver Recommendations nutzen, die Möglichkeit der automatischen Selbstoptimierung eröffnet werden. Es ist daher eine einheitliche Beschreibung des Adaptionswissens bzgl. der Recommendations nötig, welches sowohl manuell als auch automatisch – oder durch eine Kombination von beiden – erstellt werden kann.

Kompakte Repräsentation des Adaptionswissens: Das erstellte Adaptionswissen soll möglichst in einer für den Menschen verständlichen und kompakten Darstellung repräsentierbar sein. Insbesondere im E-Commerce-Bereich sind Recommendations ein wichtiger Geschäftsfaktor, so dass ein Verständnis, wie die Recommendations ausgewählt werden, zur Akzeptanz des Einsatzes adaptiver Recommendations beitragen.

Stabiles Adaptionswissen: Insbesondere bei der manuellen Erstellung des Adaptionswissens ist eine lange „Haltbarkeit“ wünschenswert. Das bedeutet, dass der Website-Administrator nicht gezwungen sein sollte, z. B. nach dem Hinzufügen neuer Webseiten zur Website das Adaptionswissen anzupassen. Auch

wenn durch verändertes Nutzungsverhalten die einzelnen Recommender veränderte Recommendations erzeugen, sollte das Adaptionswissen zur Auswahl *guter* Recommendations unverändert, d. h. stabil, bleiben.

Schnelle Auswertung des Adaptionswissens zur Laufzeit: Da für jeden einzelnen Pageview adaptive Recommendations bestimmt werden sollen, muss das Adaptionswissen schnell zur Laufzeit vom AWESOME-System ausgewertet werden können. Daher muss das Adaptionswissen in einer entsprechenden Form repräsentierbar sein, welches eine schnelle Auswertung unterstützt.

Den geschilderten Anforderungen begegnet die AWESOME-Architektur (siehe Abschnitt 3.2) mit einer Recommender-Selektion. Die Selektion der Recommender führt zu einem zweistufigen Verfahren: Das Adaptionswissen wählt für den aktuellen Kontext den besten Recommender aus, der wiederum für den aktuellen Kontext seine Recommendations ermittelt. Da es i. Allg. wesentlich weniger Recommender als potenzielle Recommendations gibt, lässt die Recommender-Selektion eine kompakte Darstellung des Adaptionswissens zu. Gleichzeitig ist es auch bzgl. Änderungen durch Hinzufügen neuer Webseiten oder durch verändertes Webnutzungsverhalten stabil. Da die Recommender auf die genannten Veränderungen durch andere Recommendations „reagieren“ können, kann die Auswahl des Recommenders unverändert bleiben.

Zur Sicherstellung einer schnellen Auswertbarkeit des Adaptionswissens sowie der Möglichkeit der manuellen und automatischen Erstellung, wird die Recommender-Selektion durch sogenannte Selektionsregeln definiert, die folgenden Aufbau haben:

$$\text{Kontextmuster} \rightarrow \text{Recommender}\,[\text{Gewicht}]$$

Der Körper der Regel ist ein Kontextmuster, der – im Gegensatz zu einem Kontext – unspezifizierte Kontextattribute besitzen kann. Dadurch repräsentiert ein Kontextmuster die Menge aller Kontexte, die in den spezifizierten Attributen mit dem Muster übereinstimmen. Der Kopf der Selektionsregel beschreibt einen Recommender aus der Recommender-Bibliothek eindeutig durch seine ID. Aus Gründen der Übersichtlichkeit wird im Folgenden jedoch stets der Name des Recommenders angegeben. Zusätzlich erhält jede Regel noch ein Gewicht, welches durch eine reelle Zahl im Intervall [0,1] ausgedrückt wird. Das Gewicht beschreibt die Relevanz der Regel, um im Falle mehrerer für einen Kontext passender Regeln eine Auswahl treffen zu können.

Die Auswertung der Selektionsregeln, d. h. das Finden für den aktuellen Kontext passender Regeln, lässt sich durch einen einfachen Vergleich mit den Kontextattributen bewerkstelligen (siehe auch den folgenden Abschnitt 4.1.1), was z. B. effektiv durch entsprechende Datenbankanfragen realisiert werden kann. Der einfache Aufbau der Regeln unterstützt weiterhin sowohl die manuelle als auch automatische Erstellung des Adaptionswissens (siehe Abschnitt 4.2).

4.1.1 Auswertung der Selektionsregeln

Die Auswahl, welcher Recommender bei einem Pageview ausgewählt wird, erfolgt anhand einer Selektionsstrategie, die eine oder mehrere Selektionsregeln umfasst. Bei jedem Pageview werden diejenigen Regeln ausgewählt, deren Kontextmuster den aktuellen Kontext abdecken. Anschließend werden sie nach ihrem Gewicht absteigend sortiert, wobei bei gleichen Gewichten eine zufällige Reihenfolge gewählt wird. Die gefundenen Regeln werden, beginnend mit der Regel mit dem höchsten Gewicht, schrittweise abgearbeitet, d. h. der im Regelkopf angegebene Recommender wird ausgeführt. Normalerweise liefert der Recommender eine Liste von Recommendations, womit die Auswertung der Regeln abgeschlossen ist. In seltenen Fällen generieren Recommender für bestimmte Kontexte jedoch keine Recommendations, z. B. wenn der Suchmaschinen-Recommender für den übergebenen Suchbegriff keinen Seiten findet. In diesem Fall wird der Recommender der zweiten Regel angewendet usw.

Zusätzlich fügt AWESOME automatisch zu jeder Selektionsstrategie die Regel

$$\emptyset \rightarrow \text{Top-Recommender}\,[0]$$

hinzu. Diese Regel beinhaltet den allgemeinsten Kontext, d. h. alle Kontextattribute sind unspezifiziert. Daher erfasst sie alle möglichen Kontexte und ist somit immer anwendbar. Durch ihr minimales Gewicht wird sie aber lediglich in dem Fall ausgewählt, wenn es keine andere Selektionsregeln für den Kontext gibt oder alle passenden Selektionsregeln keine Recommendations für den Kontext liefern. Der Kopf der angegebenen Regel verwendet den Top-Recommender, der – zumindest sofern Webnutzungsverhalten aufgezeichnet wird – unabhängig vom Kontext stets Recommendations erzeugt. Es kann auch ein beliebiger anderer Recommender als „Backup-Recommender“ verwendet werden, solange sichergestellt ist, dass der gewählte Recommender bei jedem Pageview Recommendations generiert.

4.1.2 Abgrenzung zum einstufigen Prozess

Neben dem gewählten zweistufigen Prozess für Selektionsregeln wurde in [72, 189] ein einstufiges Verfahren vorgestellt, welches die Recommendations direkt anhand des aktuellen Kontexts ohne Auswahl der Recommender bestimmt. Die Bestimmung der Recommendations für beide Verfahren ist schematisch in Abbildung 4.2 dargestellt. Während beim einstufigen Verfahren direkt auf die Menge der vorberechneten Recommendations zugegriffen wird, ermittelt das zweistufige Verfahren zunächst einen (oder mehrere) Recommender, dessen Recommendations für den aktuellen Kontext ausgewählt werden.

Der wesentliche konzeptionelle Unterschied beider Ansätze zeigt sich bei der Transformation des aufgezeichneten Recommendation-Feedbacks in das Adaptionswissen.

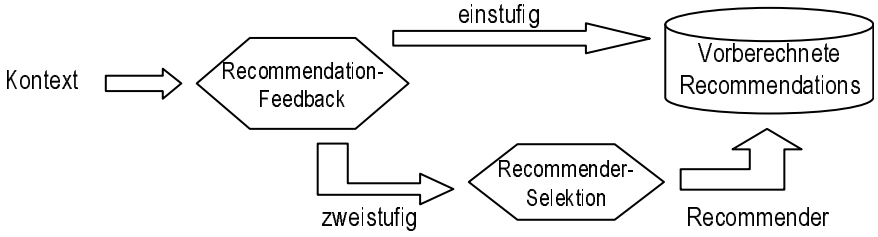


Abbildung 4.2: Schematische Darstellung des ein- und zweistufigen Verfahrens zur adaptiven Bestimmung von Recommendations

Beim zweistufigen Verfahren wird das Recommendation-Feedback, d. h. ob eine Recommendation angeklickt wurde oder nicht, dem die Recommendation erzeugenden Recommender zugeordnet. Das heißt, dass z. B. bei der manuellen Evaluation gezielt nach den besten Recommendern für ausgewählte Kontextmuster gesucht wird – unabhängig davon, welche konkreten Recommendations in den einzelnen Fällen präsentiert wurden. Das Feedback wird somit stark bzgl. der Recommender aggregiert. Der einstufige Ansatz verzichtet hingegen auf diese Aggregation und betrachtet die konkrete präsentierte und evtl. angeklickte Recommendation im aufgezeichneten Feedback. Die unterschiedliche konzeptionelle Modellierung führt zu einer Reihe unterschiedlicher Eigenschaften, die in Tabelle 4.1 kurz zusammengefasst werden. Die Vor- und Nachteile beider Ansätze werden im Folgenden beschrieben.

Websites verändern sich im Laufe der Zeit, d. h. neue Seiten werden hinzugefügt und ältere entfernt. Ein Vorteil des zweistufigen Verfahrens ist, dass diese Veränderungen keinen Einfluss auf das Adaptionswissen besitzen, solange keine Änderungen an der Recommender-Bibliothek vorgenommen werden. Insbesondere entsteht kein New-Item-Problem (Kaltstartproblem) für neu hinzugekommene Seiten, für welche keinerlei Recommendation-Feedback zur Verfügung steht. So können zwar z. B. nutzungsbasierte Recommender (z. B. der Top-Recommender) eine neue Seite nicht als Recommendation generieren, da sie (noch) nicht im aufgezeichneten Webnutzungsverhalten enthalten ist; Recommender, welche lediglich die Seite und ihren Inhalt verwenden (z. B. bei der Empfehlung ähnlicher Seiten), haben dieses Problem allerdings nicht. Da der zweistufige Prozess das Feedback den Recommendern zuordnet, unterliegt dieses nicht dem New-Item-Problem. Analog geht auch kein Adaptionswissen verloren, wenn ältere Seiten von der Website entfernt werden. Die Zuordnung des Feedbacks zu den Recommendern erzeugt somit ein deutlich kompakteres Adaptionswissen als der einstufige Prozess, was sich u. a. in der geringeren Anzahl der Selektionsregeln widerspiegelt. Daraus resultiert der Vorteil, dass es für einen Website-Administrator einfacher ist, solche Regeln manuell zu spezifizieren. Wie in der späteren Evaluation gezeigt wird (siehe Abschnitt 5.4), erreichen bereits Selektionsstrategien mit wenigen Regeln sehr gute Ergebnisse. Dies ist insbesondere für Websites interessant, die keine automatische Adaption der Recommendations wünschen, um die Kontrolle über die angezeigten Empfehlungen zu behalten. Ein wei-

Strategie	Zweistufig	Einstufig
Regeltyp / Granularität	Recommender	Recommendation
Anzahl der Regeln	wenige	viele
Regeloptimierung	offline	offline und online

Tabelle 4.1: Vergleich von ein- und zweistufiger Selektion

terer Vorteil des kompakten Adaptionswissens liegt in seiner Stabilität. Das Wissen ändert sich also nicht so stark, als würde man es für jede einzelne Recommendation aufzeichnen. Damit ist es z. B. möglich, die Adaption, d. h. die automatische Erzeugung der Selektionsregeln, in einen Offline-Prozess auszulagern und periodisch (z. B. einmal täglich) auszuführen.

Demgegenüber steht der einstufige Prozess, welcher das Adaptionswissen wesentlich feingranularer repräsentiert, in dem das Recommendation-Feedback der dargestellten Recommendation zugeordnet wird. Der wichtigste Vorteil ist die größere Möglichkeit zur Adaption. Es können sehr gute Recommendations ermittelt werden, unabhängig von der Gesamtqualität der erzeugenden Recommender. Dabei werden deutlich mehr Selektionsregeln gebildet, da für jede mögliche Recommendation mindestens eine Regel existiert. Diese große Anzahl der Regeln macht jedoch eine vollständige manuelle Regelspezifizierung nahezu unmöglich, so dass realistischerweise manuell erstellte Regeln lediglich als Ergänzung zu automatisch erstellten dienen können. Das führt insbesondere dazu, dass eine vergleichende Evaluation mit manuell erstellten Regeln nicht oder nur mit sehr großem Aufwand durchgeführt werden kann. Eine weitere Herausforderung besteht wegen der großen Zahl der Regeln darin, das Adaptionswissen für jeden Webseitenzugriff schnell auszuwerten, d. h. die passenden Recommendations zu finden. Neben der Auswertung gestaltet sich auch die Generierung der Regeln schwieriger, da es weniger Feedback pro Recommendation gibt, als für die Recommender im zweistufigen Prozess zur Verfügung steht. Insbesondere wirken sich Veränderungen der Website auf das Adaptionswissen und somit auf die Regelgenerierung stark aus, da für neue Seiten (d. h. für potenziell neue Recommendations) zunächst kein Feedback zu Verfügung steht und andererseits Feedback beim Löschen von Seiten verloren geht.

4.2 Generierung der Selektionsregeln

Selektionsregeln können sowohl manuell als auch automatisch erstellt werden. Zusätzlich ist ein hybrider Ansatz möglich, der manuell und automatisch erstellte Regeln kombiniert. Im Folgenden werden zunächst Strategien zur manuellen Regelerstellung vorgestellt. Die resultierenden Regeln sind natürlich spezifisch für die bei der prototypischen Implementation verwendeten Test-Website, weshalb der Schwer-

punkt auf die Darstellung der allgemeinen Vorgehensweise zur manuelle Regelerstellung gelegt wird. Anschließend werden zwei Verfahren präsentiert, welche Selektionsregeln automatisch und unabhängig von der verwendeten Website erstellen. Abschließend erfolgt eine kurze Darstellung der hybriden Erstellung von Selektionsregeln.

4.2.1 Manuelle Erstellung von Selektionsregeln

Für die manuelle Erstellung von Selektionsregeln gibt es keinen „besten“ Algorithmus. Sie sind vielmehr das Ergebnis eines „Modells“, welches sich an der Frage orientiert, wann, d. h. bei welchem Kontextmuster, welche Recommender sinnvolle Recommendations liefern würden. Dieses Modell kann u. a. auf der Erfahrung des Website-Administrators beruhen, d. h. z. B. auf Grund seines Wissens über die für die Website relevanten Nutzergruppen sowie deren Intentionen beim Website-Besuch. Unterstützt werden kann der Website-Administrator durch eine vorherige Evaluation des Webnutzungsverhaltens (siehe Abschnitt 5.2 für eine beispielhafte Evaluation) sowie der Effektivität der Recommender bezüglich der Akzeptanz von gegebenen Empfehlungen (Abschnitt 5.3).

Es werden im Folgenden zwei manuelle Strategien für die Beispiel-Website vorgestellt, welche später (siehe Abschnitt 5.4) auch evaluiert werden.

Strategie: Manuell 1

Die erste Strategie für die Beispiel-Website verwendet Recommender, welche die am häufigsten aufgerufenen Seiten einer Kategorie bestimmen. Dabei wird u. a. die in Abbildung 3.2 dargestellte Art der Seitenkategorisierung bei der Beispiel-Website verwendet. Alle Recommender erhalten dabei das gleiche Gewicht. Die verwendete Kategorie richtet sich dabei nach dem Nutzertyp. Für neue Nutzer wird die Kategorie der aktuellen Seite verwendet. Demgegenüber wird für wiederkehrende Nutzer auf das persönliche Profil zurückgegriffen, d. h. die Kategorie mit den meisten Seitenaufrufen früherer Website-Besuche des Nutzers verwendet. Es entstehen dabei die folgenden zwei Arten von Regeln:

{UserType=„Neu“}	→	TopCat	[1]
{UserType=„Wiederkehrend“}	→	PersInt	[1]

Dabei steht jeder der beiden Recommender stellvertretend für zwölf[28] Recommender, die sich lediglich in der Parametrisierung unterscheiden (12 = 2 Kategorisierungen ·

[28] Im Vergleich zu Tabelle 3.4 aller implementierten Recommender wurde jeweils nur das Verfahren Top und nicht TopInc verwendet.

2 Ebenen · 3 Zeitfenster). Somit ergeben sich 24 Regeln zuzüglich der automatisch hinzugefügten Regel (siehe Abschnitt 4.1.1).

Strategie: Manuell 2

Die zweite manuelle Strategie beruht auf der Erkenntnis, dass ein Großteil der Website-Besucher über eine Suchmaschine zur Website gelangen. Dadurch stehen meist in der Referrer-Information die eingegeben Suchbegriffe zur Verfügung, welche der Suchmaschinen-Recommender (SER) verwendet. Für die restlichen Nutzer basiert die Recommender-Auswahl auf der aktuellen Seite. Für Seiten, die Skript-Material beinhalten, wird der Ähnlichkeits-Recommender (Sim) ausgewählt, da diese Seiten im Vergleich zu anderen Seiten viel Inhalt besitzen, was für die Bestimmung ähnlicher Seiten von Vorteil ist. Für alle anderen Seiten wird der Assoziationsregeln-Recommender, der die häufigsten direkten Nachfolgerseiten innerhalb einer Session ermittelt (Asso-Direct), mit verschiedenen Zeitfenstern verwendet. Die Strategie umfasst daher die folgenden vier Regeln (zzgl. der automatisch angefügten Regel):

{Referrer=„Suchmaschine“}	→	SER	[1]
{PageType1=„Studium“ & PageType2=„Skript“}	→	Sim	[0.9]
∅	→	Asso-1-Direct	[0.8]
∅	→	Asso-7-Direct	[0.8]

Zufällige Recommender-Auswahl

Als Vergleichsstrategie für eine spätere Evaluation wird eine zufällige Auswahl der Recommender modelliert. Dies wird durch Regeln der Form

$$\emptyset \rightarrow Recommender_X[1]$$

erreicht, d. h. für jeden Recommender gibt es eine Regel, bei dessen Kontextmuster alle Kontextattribute unspezifiziert sind. Weiterhin haben alle Regeln ein einheitliches Gewicht von 1. Dabei wird der Umstand ausgenutzt, dass AWESOME nach dem Finden der passenden Selektionsregeln bei gleichem Gewicht eine zufällige Reihenfolge auswählt. Da alle Regeln für jeden Kontext anwendbar sind, entsteht dadurch eine zufällige Recommender-Auswahl.

4.2.2 Automatische Erstellung von Selektionsregeln

Für die automatische Erstellung von Selektionsregeln wurden zwei verschiedene Ansätze entworfen. Der anfragebasierte Ansatz macht sich die Data-Warehouse-Funktionalitäten zunutze und sucht für alle Kontextmuster, die mit einer bestimmten

```
PROCEDURE QueryBasedRules (MinSupport)
BEGIN
  Finde alle relevanten Kontextmuster M, d.h. alle Muster,
    die mindestens MinSupport-mal aufgetreten sind
  Für jedes Kontextmuster m aus M
    Suche Recommender r mit höchster Akzeptanzrate a
    Füge Regel m -> r [a] hinzu
  Lösche unnötige Regeln
END
```

Abbildung 4.3: Algorithmus zur anfragebasierten Bestimmung von Selektionsregeln

Häufigkeit aufgetreten sind, den jeweils besten Recommender. Mit Hilfe maschinellen Lernens werden die Selektionsregeln im zweiten Verfahren erzeugt. Dabei wird das gespeicherte Feedback in Trainingsdaten konvertiert, die wiederum Basis eines Lernverfahrens sind. Das Ergebnis des Lernprozesses wird anschließend in Selektionsregeln umgeschrieben.

Strategie: Anfragebasiert

Die Idee des anfragebasierten Ansatzes ist es, für einen Kontext aus der Menge aller passenden und hinreichend oft auftretenden Kontextmuster dasjenige auszuwählen, welches die höchste Akzeptanzrate für einen Recommender besitzt. Abbildung 4.3 illustriert den entsprechenden Algorithmus.

Im ersten Schritt werden nur diejenigen Muster betrachtet, die hinreichend oft innerhalb des Feedbacks aufgetreten sind. Dadurch wird vermieden, dass sehr seltene Kontextmuster in den Regeln auftreten (Overfitting). Zur Generierung aller Kontextmuster kann der *GROUP BY CUBE* Operator verwendet werden, der als SQL-Erweiterung in [74] eingeführt wurde und in der prototypischen Implementierung zur Verfügung stand. Ohne diesen Operator lassen sich alle Kontextmuster durch einen rekursiven Algorithmus bestimmen, welcher ausgehend von dem allgemeinsten Kontextmuster (alle Attribute sind unspezifiziert) durch schrittweise Verfeinerung (Spezifizierung der Attribute mit Werten) weitere (spezifischere) Kontextmuster generiert. Wird dabei ein Kontextmuster gebildet, welches den minimalen Support nicht mehr erreicht, kann dieser Teil der Rekursion abgebrochen werden, da weitere Verfeinerungen keine höheren Support-Werte besitzen.

Für jedes gefundene Kontextmuster m wird im zweiten Schritt die Akzeptanzrate pro Recommender r bestimmt. Diese ist wie folgt definiert:

$$A(m, r) = |m(r)_A|/|m(r)|$$

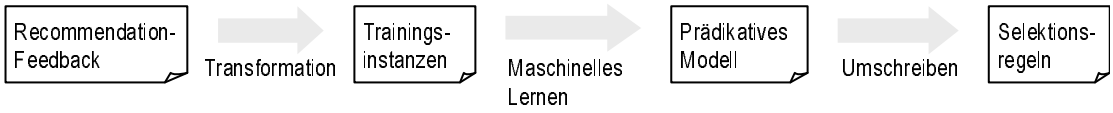


Abbildung 4.4: Allgemeine Darstellung der Regelgenerierung mittels maschinellen Lernens

Dabei bezeichne $m(r)$ die Menge der Pageviews mit dem Kontextmuster m, bei denen Recommendations des Recommenders r angezeigt wurden. $m(r)_A$ ist diejenige Teilmenge von $m(r)$, bei denen eine Recommendation angeklickt wurde. Offensichtlich ist die Akzeptanzrate eine reelle Zahl im Intervall $[0, 1]$ und kann daher als Gewicht für die Regeln $m \rightarrow r[a]$ verwendet werden.

Nach dem Bilden der Regeln ist der anfragebasierte Ansatz im Wesentlichen abgeschlossen. Da jedoch eine Auswertung der Regeln zur Laufzeit bei jedem Pageview erfolgt, ist darauf zu achten, eine möglichst geringe Zahl von Regeln zu generieren. Der soweit vorgestellte Algorithmus erzeugt jedoch auch unnötige Regeln, die im letzten Schritt entfernt werden. Als Beispiel seien die folgenden zwei Regeln gegeben:

{PageType1=„Studium“}	→	Sim	[0.5]
{PageType1=„Studium“ & PageType2=„Übung“}	→	Asso-1-Direct	[0.3]

Die zweite Regel des Beispiels kommt nie zur Anwendung, da sie komplett von der ersten überlagert wird. Diese Überlagerung besteht aus zwei Teilen: Erstens ist das Kontextmuster der Regel 1 allgemeiner als das von Regel 2, d. h. alle Kontexte die vom Muster der Regel 2 erfasst werden, werden auch vom Muster der Regel 1 erfasst. Zweitens ist das Gewicht der ersten Regel größer als das der zweiten. Auf Grund der Auswertung der Regelmenge kommt die zweite Regel nie zur Anwendung und kann daher gelöscht werden.

Diese Beobachtung wird nun im letzten Schritt des Algorithmus ausgenutzt. Gibt es für eine Regel X eine andere Regel Y mit größerem Gewicht und einem Kontextmuster, welches eine Verallgemeinerung des Kontextmusters von X ist, so wird Regel X gelöscht.

Strategie: Entscheidungsbaum

Die Recommender-Selektion von AWESOME lässt sich als Klassifikationsproblem betrachten: Für einen gegebenen Kontext soll ein Recommender aus einer definierten Menge aller Recommender ausgewählt werden, d. h. der Kontext wird einer „Recommender-Klasse“ zugeordnet (klassifiziert). Dadurch lassen sich Klassifikationsalgorithmen zur automatischen Generierung von Selektionsregeln anwenden. Das

```
PROCEDURE DecisionTreeRules (MinSupport, SkalFaktor)
BEGIN
  Finde alle auftretenden Kombinationen von Kontext und
    Recommender (k, r), die MinSupport erreichen
  Für jede Kombination (k, r)
    Bestimme Akzeptanzrate a für k(r)
    Skaliere a mit SkalFaktor
    Runde das Ergebnis auf eine ganze Zahl n
    Füge n-mal Trainingsinstanz (k, r) den Traingsdaten hinzu
  Wende Entscheidungsbaumverfahren auf alle Trainingsdaten an
  Schreibe Entscheidungsbaum in Selektionsregeln um
END
```

Abbildung 4.5: Algorithmus zur Bestimmung von Selektionsregeln mittels Entscheidungsbaum

allgemeine Vorgehen wird in Abbildung 4.4 dargestellt.

Das aufgezeichnete Feedback muss dabei zunächst in Trainingsdaten transformiert werden. Diese dienen als Eingabe für einen Klassifikationsalgorithmus, z. B. für ein Entscheidungsbaumverfahren. Der Klassifikationsalgorithmus generiert daraufhin ein prädikatives Modell, d. h. eine Zuordnungsvorschrift, die jedem Kontext einen Recommender zuweist. Struktur und Aufbau des Modells hängen vom verwendeten Algorithmus ab. So erzeugt zum Beispiel ein Entscheidungsbaumverfahren einen Entscheidungsbaum, dessen Klassifikationsvorschrift durch das Traversieren des Baumes von der Wurzel zu den Blattknoten entsteht. Abschließend wird das Modell in Selektionsregeln umgewandelt.

Im Rahmen der prototypischen Implementierung wurde mit J48 ein Entscheidungsbaumverfahren der WEKA-Bibliothek eingesetzt [205]. Die Berechnung der Selektionsregeln wurde durch den in Abbildung 4.5 aufgeführten Algorithmus realisiert. Im Unterschied zum anfragebasierten Ansatz werden hier Kombinationen zwischen Kontext und Recommender betrachtet. Der als Parameter übergebene minimale Support ist daher auch deutlich niedriger als beim anfragebasierten Ansatz, welcher zunächst nach Kontextmustern sucht. Die Berechnung der Akzeptanzrate erfolgt analog zum vorherigen Ansatz. Da diese im Bereich von [0, 1] liegt, wird sie mit einem Skalierungsfaktor (zweiter Parameter) auf einen größeren Bereich (z. B. [0, 100]) skaliert und anschließend gerundet. Damit erhält man eine ganze Zahl, die die Häufigkeit der Kombination in den Trainingsdaten angibt.

Beispiel: Für einen gegebenen Kontext k = { Referrer = „Suchmaschine“, PageType1 = „Navigation“ } gibt es 50 Pageviews, bei dem Recommendations des Recommenders r = Top-Recommender angezeigt wurden. Bei sieben Pageviews wurde dabei eine Recommendation angeklickt, was einer Akzeptanzrate von 0,14 entspricht. Die-

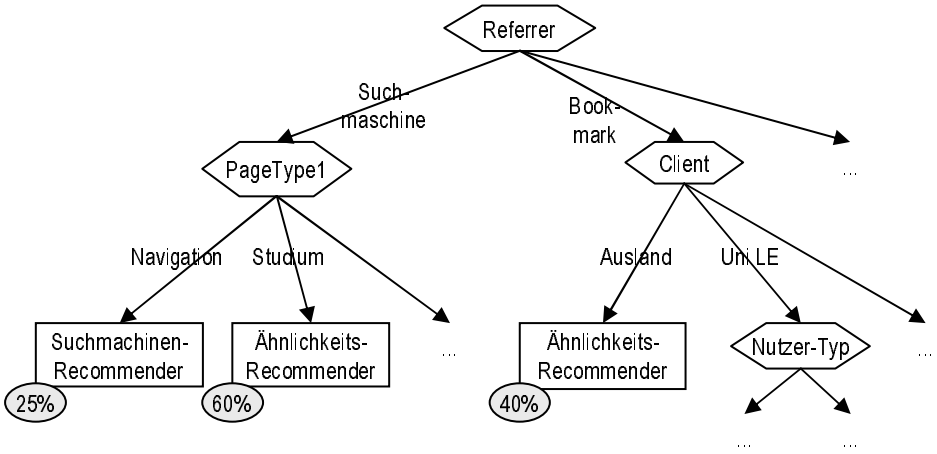


Abbildung 4.6: Ausschnitt eines Entscheidungsbaum zur Generierung von Selektionsregeln

se wird mit einem Skalierungsfaktor von 100 multipliziert und gerundet. Im Ergebnis werden anschließend 14 identische Trainingsinstanzen (k, r) den Trainingsdaten hinzugefügt.

Dieses Vorgehen stellt sicher, dass Kombinationen mit hoher Akzeptanzrate häufiger in den Trainingsdaten zu finden sind und daher verstärkt gelernt werden. Anschließend wird ein Entscheidungsbaumverfahren angewendet, welches einen Entscheidungsbaum zurückliefert. Abbildung 4.6 zeigt einen entsprechenden Ausschnitt. Die inneren Knoten des Baumes enthalten die Kontextattribute. Die ausgehenden Kanten sind mit den einzelnen Attributwerten belegt und die Blattknoten repräsentieren die Recommender. Letztere sind zusätzlich mit einer Klassifikationswahrscheinlichkeit annotiert, welche den Anteil der Trainingsinstanzen mit dem entsprechenden Kontext angibt, die dem im Blattknoten beschriebenen Recommender zugeordnet sind. Die Klassifikation ist durch den Pfad von der Wurzel zum Blattknoten definiert. Im Beispiel ordnet der Entscheidungsbaum einem Kontext k den Suchmaschinen-Recommender mit einer Wahrscheinlichkeit von 25% zu, falls k durch das Kontextmuster {Referrer = „Suchmaschine“ und PageType1 = „Navigation“} erfasst wird.

Die abschließende Umschreibung des Baumes in Selektionsregeln ist nun einfach durch das Traversieren des Baumes möglich. Für jeden Blattknoten ergibt sich eine Selektionsregel, dessen Kontextmuster durch den Pfad definiert ist. Als Gewicht der Regel wird die Klassifikationswahrscheinlichkeit gewählt. Für obigen Entscheidungsbaum ergeben sich u. a. die folgenden drei Selektionsregeln:

{Referrer=„Suchmaschine“ & PageType1=„Navigation“}	→	SER	[0.25]
{Referrer=„Suchmaschine“ & PageType1=„Studium“}	→	Sim	[0.6]
{Referrer=„Bookmark“ & Client=„Ausland“}	→	Sim	[0.4]

```
PROCEDURE HybridRules (ManRules, AutoRules)
BEGIN
  Für jede Regel m -> r [g] aus ManRules
    Füge Regel m -> r [g] hinzu
  Bestimme minimales Gewicht g_min aus ManRules
  Für jede Regel m -> r [a] aus AutoRules
    Füge Regel m -> r [g*g_min/2] hinzu
  Lösche unnötige Regeln
END
```

Abbildung 4.7: Algorithmus zur hybriden Bestimmung von Selektionsregeln aus manuellen und automatisch erzeugten Regeln

4.2.3 Hybride Erstellung von Selektionsregeln

Ziel der hybriden Erstellung von Selektionsregeln ist es, die (i. Allg. vielen) automatisch erstellten Regeln durch (i. Allg. wenige) manuelle Regeln zu verbessern. Dadurch kann z. B. das Wissen des Website-Administrators über ausgewählte Nutzergruppen mit den automatisch generierten Regeln für die verbleibenden Nutzergruppen kombiniert werden.

Die Erstellung der hybriden Selektionsstrategie ist in Abbildung 4.7 dargestellt. Dabei werden zunächst alle manuell erstellten Regeln übernommen. Zusätzlich wird das minimale Regelgewicht g_{min} aller manuell erstellten Regeln bestimmt. Da jedes Regelgewicht im Intervall $[0, 1]$ liegt, gilt: $0 \leq g_{min} \leq 1$. Anschließend werden auch alle automatisch erstellten Regeln übernommen, wobei die Regelgewichte jeweils mit $g_{min}/2$ multipliziert werden. Diese Anpassung stellt sicher, dass jede manuell erstellte Regel – sofern ihr Gewicht positiv ist – ein höheres Gewicht als alle automatisch erstellten Regeln hat. Abschließend können, wie bereits bei der anfragebasierten Regelgenerierung dargestellt, unnötige Regeln gelöscht werden. Innerhalb der prototypischen Implementation wurden für eine hybride Strategie die Regeln des Entscheidungsbaumverfahrens mit den ersten zwei Regeln der zweiten manuellen Strategie kombiniert.

4.3 Zusammenfassung

Die adaptive Auswahl von Recommendations wird durch einen zweistufigen Prozess realisiert. Dabei wird zunächst in Abhängigkeit vom aktuellen Kontext ein Recommender bestimmt, dessen Recommendations anschließend berechnet und angezeigt werden. Dieser Vorgang wird durch Selektionsregeln definiert, welche sowohl manuell als auch automatisch anhand des aufgezeichneten Feedbacks erstellt werden

können. Für die automatische Regelerstellung können u. a. Techniken des maschinellen Lernens angewendet werden. Dadurch wird eine automatische Optimierung der Recommendations ohne zusätzlichen manuellen Aufwand ermöglicht. Zusätzlich können manuelle und automatische Regelerstellung in einem hybriden Verfahren miteinander kombiniert werden.

5 Evaluation

Dieses Kapitel beinhaltet eine exemplarische Evaluation des Webnutzungsverhaltens, der Recommendation-Qualität sowie der Selektionsstrategien für die prototypische Beispiel-Website. Die konkreten Ergebnisse, d. h. z. B. welcher Recommender die beste Recommendation-Qualität erzielt, sind dabei nur von sekundärem Interesse, da diese Ergebnisse u. a. nicht auf andere Websites übertragbar sind. Das Ziel besteht vielmehr im Aufzeigen einer Vorgehensweise, wie für eine gegebene Website durch zielgerichtete Analyse die Recommendation-Qualität verbessert werden kann. Dazu wird zunächst der Aufbau der Evaluation, inklusive der Evaluationsarten sowie der verwendeten Metriken, vorgestellt (Abschnitt 5.1). Anschließend erfolgt die Evaluation des Webnutzungsverhaltens (Abschnitt 5.2) sowie eine Evaluation der Recommender (Abschnitt 5.3). Den Abschluss des Kapitels bildet eine vergleichende Evaluation der unterschiedlichen Selektionsstrategien zur Bestimmung adaptiver Recommendations (Abschnitt 5.4).

5.1 Aufbau der Evaluation

Innerhalb des Data Warehouses liegen alle Informationen zur Webnutzung sowie der präsentierten Recommendations in einer multidimensionalen Struktur eines Data Warehouses vor (siehe Kapitel 3.5). Zur interaktiven (manuellen) Auswertung dieser Daten kann daher eine sogenannte OLAP-Analyse (Online Analytic Processing [30]) durchgeführt werden. Dabei bezeichnet OLAP eine Kategorie von Applikationen bzw. Technologien für die multidimensionale Datenanalyse, welche durch

mehrere Kriterien gekennzeichnet sind [54]. Die wichtigsten sind dabei eine multidimensionale konzeptionelle Sicht auf die Daten, unbeschränkte und dimensionsübergreifende Operationen sowie eine unbegrenzte Anzahl von Dimensionen und Konsolidierungsebenen.

Ein wichtiges Einsatzgebiet von OLAP-Analysen ist Business Intelligence, d. h. „die entscheidungsorientierte Sammlung, Aufbereitung und Darstellung geschäftsrelevanter Informationen“ [171]. Dabei werden z. B. Verkaufsdaten aller Filialen eines Unternehmens in einem Data Warehouse kombiniert und analysiert, so dass daraus gewonnene Erkenntnisse, z. B. in welchen Regionen welche Produktarten besonders erfolgreich sind, in geschäftliche Maßnahmen umgesetzt werden. Analog kann mit Hilfe des AWESOME-Warehouses das Webnutzungsverhalten analysiert werden, um z. B. häufig aufgerufene Seiten zu identifizieren. Zusätzlich kann der Erfolg bzw. die Qualität der präsentierten Recommendations mittels verschiedener Metriken evaluiert werden. Dabei lassen sich für AWESOME vier Arten der Evaluation unterscheiden.

Webnutzungsverhalten: Die Analyse des Nutzungsverhaltens gibt Hinweise darauf, welche Nutzer sich wann für welche Inhalte der Website interessieren. So können u. a. „interessante“ Nutzergruppen identifiziert werden, d. h. Nutzer, die einen hinreichend großen Anteil an der gesamten Webnutzung ausmachen und durch ein Kontextmuster gut beschrieben werden können. Das können z. B. alle Nutzer sein, die mit Hilfe einer Suchmaschine die Website erreicht haben. Die resultierenden Kontextmuster können dann bei den folgenden Arten der Evaluation verwendet werden.

Kontextbasierte Recommender-Qualität: Die Qualität der Recommendations eines Recommenders hängt u. a. vom aktuellen Kontext ab. Die in der Recommender-Bibliothek zur Verfügung stehenden Recommender beruhen auf unterschiedlichen Verfahren und verwenden unterschiedliche Daten. Es wird daher analysiert, in welchen Kontexten welche Recommender besonders gut abschneiden.

Recommender-Tuning: Beim Recommender-Tuning können verschiedene Variationen eines Recommender-Verfahrens bzgl. der Qualität miteinander verglichen werden. So kann z. B. ermittelt werden, ob für einen Top-Recommender (siehe Tabelle 3.4) das Nutzungsverhalten des letzten Tages oder der letzten Woche herangezogen werden sollte.

Vergleich von Selektionsstrategien: Selektionsstrategien definieren eine kontextbasierte Auswahl von Recommendern und können – analog zu Recommendern – durch die Qualität der produzierten Recommendations evaluiert werden. Insbesondere ist dabei eine vergleichende Evaluation von Interesse, d. h. welche der vorgestellten Selektionsstrategien besonders gute Ergebnisse liefert.

Im Folgenden werden zunächst Metriken zur Evaluation vorgestellt. Wie bereits bei den Datenquellen, dem Data-Warehouse-Schema sowie bei den zur Verfügung stehenden Recommendern dargestellt, ist AWESOME ein generischer Ansatz für beliebige Websites. Daher können die vorgestellten Maße um Website- bzw. domänenspezifische Metriken ergänzt werden. Anschließend erfolgt eine kurze Darstellung der in der prototypischen Implementation verwendeten Website.

5.1.1 Evaluationsmetriken

Die *Acceptance Rate* misst die Qualität von Recommendations direkt danach, ob sie angeklickt wurden oder nicht. Sie ist definiert als Anteil der akzeptierten (d. h. angeklickten) Recommendations von der Menge aller Pageviews, welche Recommendations beinhalten. Es gilt daher:

$AcceptanceRate = \frac{|P_A|}{|P|}$

wobei P die Menge aller Pageviews bezeichnet und P_A deren Teilmenge, bei denen eine angezeigte Recommendation akzeptiert wurde. Der Wert der *Acceptance Rate* ist stets kleiner 1, da beim letzten Pageview einer Session die gegebenen Empfehlungen nie angeklickt werden können, da es sich sonst nicht um den letzten Pageview handeln würde. Des Weiteren sind hohe Werte der *Acceptance Rate* unrealistisch, da nicht davon auszugehen ist, dass Nutzer lediglich durch Anklicken der Recommendations innerhalb einer Website navigieren. Eine nicht so strenge Metrik ist daher die *Session Acceptance Rate*, welche die Qualität der Recommendations danach misst, ob mindestens eine der präsentierten Empfehlungen innerhalb einer Session akzeptiert wurde. Es gilt somit:

$SessionAcceptanceRate = \frac{|S_A|}{|S|}$

Analog zur *Acceptance Rate* bezeichnet S die Menge aller Sessions und S_A deren Teilmenge mit mindestens einer akzeptierten Recommendation.

Neben der Analyse, ob eine gegebene Recommendation angeklickt wurde, kann auch analysiert werden, ob die dabei empfohlene Seite im Laufe der Session aufgerufen wurde. Das bedeutet, dass eine Empfehlung auch dann als Erfolg gewertet wird, wenn sie zwar nicht angeklickt wurde, aber eine korrekte Vorhersage des Nutzerinteresses darstellt. Es lassen sich somit *View Rate* und *Session View Rate* wie folgt definieren:

$ViewRate = \frac{|P_V|}{|P|}$

$SessionViewRate = \frac{|S_V|}{|S|}$

Dabei bezeichne P_V die Menge der Pageviews, bei denen die präsentierte Recommendation im späteren Verlauf der Session aufgerufen wurde. S_V ist die Menge aller Sessions, die mindestens solch einen Pageview enthalten. Offenbar wird jede akzeptierte Recommendation auch aufgerufen, so dass $P_A \subseteq P_V \subset P$ gilt. Analog folgt

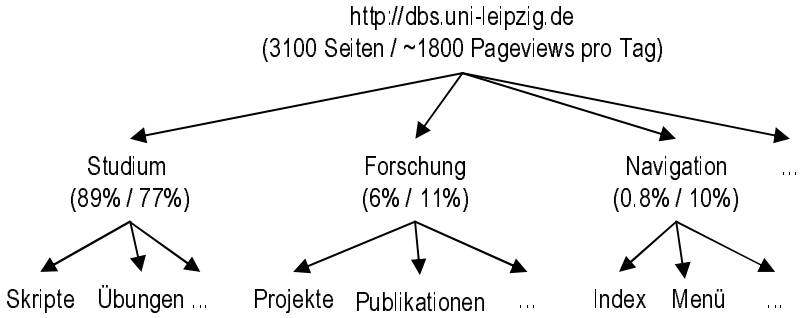


Abbildung 5.1: Inhaltliche Struktur der Evaluations-Website mit Angabe der Anzahl der Seiten und durchschnittlicher Anzahl der täglichen Pageviews (Dezember 2003 - April 2004)

$S_A \subseteq S_V \subseteq S$. Dadurch sind die entsprechenden *View*-Metriken natürlich größer als die *Acceptance*-Metriken.

Neben den vorgestellten Metriken lassen sich noch weitere, domänenspezifische Metriken definieren. Im E-Commerce-Bereich ist z. B. eine *Purchase Rate* interessant, die angibt, ob ein empfohlenes Produkt später auch gekauft wurde. Da die Anzahl der Klicks für den Umsatz des E-Commerce-Shops nicht interessant ist, ist auch eine *Session Purchase Rate* sinnvoll, die angibt, bei wieviel Prozent der Sessions mindestens eine gegebene Empfehlung zum Kauf des entsprechenden Produkts führte.

5.1.2 Evaluations-Setup

Wie bereits erwähnt, wurde die prototypische Implementierung für die Website der Abteilung Datenbanken realisiert (siehe Abbildung 3.7 auf Seite 62 für einen Screenshot) und daher mit dieser die folgende Evaluation durchgeführt. Abbildung 5.1 zeigt den groben Aufbau der Website, welche – nach der Crawler-Eliminierung und ohne Zugriffe von Rechnern der Abteilung[29] – im Zeitraum vom 1. Dezember 2003 bis 30. April 2004 ca. 1800 Pageviews pro Tag erhalten hat. Die meisten Seitenaufrufe (77%) entfallen dabei auf den Bereich Studium, der auch die meisten Seiten (89%) enthält. Die in Abbildung 5.1 dargestellte Hierarchie ist gleichzeitig eine der beiden verwendeten Inhaltshierarchien.

Wie bereits in Abschnitt 3.6.3 beschrieben, wurde die Website um einen Bereich ergänzt, der zwei Recommendations am linken Rand darstellt. Dazu wurde für jeden Pageview ein Recommender bestimmt und dessen zwei beste Recommendations angezeigt, so dass pro Tag ca. 3600 Recommendations dargestellt wurden. Dabei wurden die in der Recommender-Bibliothek implementierten Verfahren verwendet

[29] Die Zugriffe durch Mitarbeiter der Abteilung wurden entfernt, da diese u. a. zum Testen der korrekten Funktionsweise des Recommender-Systems die Website besuchten und dabei gezielt Recommendations anklickten, wodurch das Evaluationsergebnis verfälscht würde.

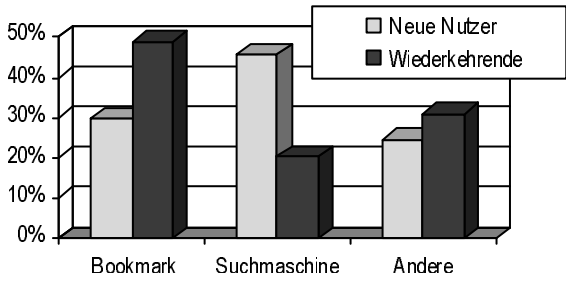


Abbildung 5.2: Relativer Vergleich der Anzahl der Pageviews für neue und wiederkehrende Nutzer bzgl. des Session-Referrers (Dezember 2003 - Januar 2004)

(siehe Tabelle 3.4).

5.2 Evaluation des Webnutzungsverhaltens

Die Analyse des Webnutzungsverhaltens dient Website-Administratoren dazu, sich einen Überblick über die Nutzung der Website durch die Besucher zu verschaffen. Es lässt sich z. B. ermitteln, woher die Besucher kommen, welche Einstiegs- und Ausstiegsseiten bei den einzelnen Sessions häufig auftreten oder welche Seiten i. Allg. am häufigsten aufgerufen werden. Solche Nutzungsanalysen können u. a. durch entsprechende Tools, z. B. Webalizer[30] oder Google Analytics[31], oder mittels OLAP-Analyse durchgeführt werden [184]. Auf Basis der Ergebnisse kann die Website optimiert werden, z. B. indem häufige Ausstiegsseiten, d. h. die letzten Seiten einer Session, umgestaltet werden, damit die Inhalte der Website deutlicher für den Besucher sichtbar sind und Nutzer zu einem längeren Verbleib auf der Website animiert werden.

Innerhalb dieser Evaluation soll die Analyse des Webnutzungsverhaltens mit Bezug auf die spätere Evaluation der Recommendation-Qualität erfolgen. Daher wird zunächst das Webnutzungsverhalten analysiert, um relevante Kontextmuster zu identifizieren, die einen hinreichenden großen Teil der Webnutzung innerhalb der Website ausmachen und deren Nutzungsverhalten sich signifikant von anderen Kontextmustern unterscheidet. Ein Beispiel dafür ist die Unterscheidung von Nutzergruppen nach neuen und wiederkehrenden Besuchern. Für die Beispiel-Website zeigt sich, dass ca. 44% aller Pageviews von wiederkehrenden Nutzern stammen. Dadurch ergeben sich mit neuen und wiederkehrenden Nutzern zwei in etwa gleich große Nutzergruppen, die bei der späteren Recommender-Evaluation berücksichtigt werden können.

Einen – wenig überraschenden – Zusammenhang zwischen Nutzertyp und Art des

[30] http://www.mrunix.net/webalizer/
[31] http://www.google.com/analytics/

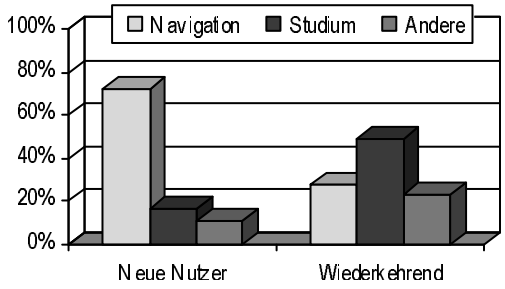


Abbildung 5.3: Wahrscheinlichkeitsverteilung für den Typ der nachfolgend aufgerufenen Seite, ausgehend von einer Seite des Typs Navigation (Dezember 2003 - Januar 2004)

Referrers zeigt Abbildung 5.2. Dabei werden die Referrer in die Klassen „Bookmark“ (kein Referrer), „Suchmaschine“ (Referrer ist URL einer bekannten Suchmaschine) und „Andere“ (sonstiger Referrer) eingeteilt. Für wiederkehrende Nutzer ist der Anteil der Referrer des Typs „Bookmark“ besonders hoch, d. h. dass wiederkehrende Nutzer die Website gezielt ansteuern, z. B. durch Eingabe der URL der Website im Browser oder Verwendung eines Bookmarks (Lesezeichen).[32] Andererseits gelangen neue Nutzer im Wesentlichen über eine Suchmaschine zur Website.

Den potenziellen Nutzen von Recommendations insbesondere für neue Nutzer zeigt Abbildung 5.3. Betrachtet wird die Wahrscheinlichkeitsverteilung bzgl. des Typs der als nächstes aufgerufenen Webseite, wenn man sich als Nutzer auf einer Navigationsseite befindet. Abbildung 5.3 zeigt u. a., dass für neue Nutzer rund 70% der Nachfolgeseiten einer Navigationsseite ebenfalls wieder eine Navigationsseite ist. Das bedeutet, dass neue Nutzer mehrere Klicks benötigen, um von Navigationsseiten zu den sie interessierenden Seiten mit entsprechendem Inhalt zu gelangen. Demgegenüber gelangen wiederkehrende Nutzer zielgerichtet auf Seiten des Typs Studium, was die Vermutung nahe legt, dass die Masse der wiederkehrenden Nutzer Studenten sind und sich offenbar mit der Website und ihrer Struktur gut auskennen.

Die geschilderte kurze Evaluation illustriert ein mögliches prinzipielles Vorgehen bei der Evaluation des Webnutzungsverhaltens sowie die Mächtigkeit der Data-Warehouse-basierten OLAP-Analyse. Zunächst wurden Nutzergruppen identifiziert, welche einen hinreichend großen Anteil am Nutzungsverhalten haben. Anschließend wurden mittels mehrdimensionaler Analyse Unterschiede im Webnutzungsverhalten aufgedeckt. Die dabei verwendeten Kontextinformationen (z. B. Nutzertyp, Referrer, Seitentyp) können anschließend zur vergleichenden Evaluation der Recommender herangezogen werden.

[32] Dass Erreichen einer Website ohne Referrer ist zusätzlich noch dadurch möglich, dass der verwendete Browser grundsätzlich keinen Referrer sendet. Da dies aber nicht der Standardeinstellung gängiger Browser entspricht, wird dieser Fall hier vernachlässigt.

5.3 Evaluation der Recommender

Das Data Warehouse erlaubt eine ausführliche Evaluation für eine Vielzahl von Kontext- bzw. Dimensionsattributen. Durch entsprechende Aggregation lassen sich beliebige Kontextmuster bilden, für welche die Qualität der Recommender oder Recommender-Klassen bestimmt werden kann. Im Evaluationszeitraum 1. Dezember 2003 - 30. April 2004 liegt die durchschnittliche *Acceptance Rate* für alle betrachteten Recommender bei 1,44%. Die durchschnittliche *View Rate* beträgt 15,89%. Für alle Sessions mit mehr als einem Seitenaufruf ergeben sich als *Session Acceptance Rate* bzw. *Session View Rate* durchschnittliche Werte von 9,03% bzw. 27,83%.

Die gemessenen Werte sind relativ gering, was im Wesentlichen an der verwendeten Website liegt. Zunächst beinhaltet jede aufgerufene Seite ein vollständiges Menü mit 78 (zum Teil in Untermenüs vorhandenen) Links. Insbesondere mit der Website erfahrene Nutzer können so direkt zu den ihnen bekannten und für sie relevanten Inhalten navigieren. Der zweite Einflussfaktor für die Akzeptanzrate der Recommendations ist die Darstellung der Recommendation selbst. Im AWESOME-Prototyp wurde bewusst auf große und auffällige Schriften verzichtet. Stattdessen werden die Empfehlungen mittels Hyperlinks dezent am linken Rand platziert. Nichtsdestoweniger liegen die gemessenen Akzeptanzraten im Bereich anderer Systeme [41].

Die absoluten Werte sind jedoch auch nicht aussagekräftig, da sie von Website zu Website differieren und – wie oben erwähnt – von verschiedenen Layout-Faktoren abhängen. Für eine Evaluation sind vielmehr relative Unterschiede zwischen Verfahren bei gleichen Bedingungen interessant. Daher wird im Folgenden zunächst eine kontextbasierte Evaluation verschiedener Recommender durchgeführt, welche z. B. die Unterschiede der einzelnen Verfahren für verschiedene Nutzer und Seiten aufzeigt. Anschließend wird das prinzipielle Vorgehen für ein Recommender-Tuning vorgestellt, bei dem für einen Recommender die optimale Parametrisierung des zu Grunde liegenden Berechnungsverfahrens, z. B. welches Zeitfenster des Webnutzungsverhaltens verwendet werden sollte, gefunden wird.

5.3.1 Kontextbasierte Recommender-Qualität

Dieser Abschnitt beinhaltet eine beispielhafte Recommender-Evaluation auf Basis der *Acceptance Rate*. Um eine übersichtliche Evaluation der Recommender in Abhängigkeit verschiedener Kontexte zu ermöglichen, wurden mehrere Recommender bzgl. ihres Typs zusammengefasst. Weiterhin werden nur fünf der in Abschnitt 2.3 vorgestellten Klassen betrachtet, da für die verbleibenden Klassen (–,–,–), (+,+,–) und (+,+,+) zu wenig Daten für eine Evaluation zur Verfügung stehen. Dabei wurden nur Recommender mit einem minimalen Support von 5% ausgewählt, d. h. bei mindestens 5% aller Pageviews wurde der Recommender angewendet. Dies ist auf Grund weniger Seitenänderungen innerhalb des Evaluationszeitraums für die Re-

Recommender		Nutzertyp		
Typ	Beschreibung (Name)	Neue Nutzer	Wieder-kehrende	Gesamt
(–,–,+)	Häufigste Pageviews (Top)	1,05%	1,19%	1,09%
(–,+,–)	Suchmaschinenschlagworte (SER)	2,83%	1,42%	2,77%
(–,+,+)	Persönliche Interessen (PersInt)	-	1,43%	1,43%
(+,–,–)	Ähnlichkeit der Seiten (Sim)	1,86%	0,76%	1,73%
(+,–,+)	Assoziationsregeln (Asso)	1,61%	1,49%	1,59%
	Gesamt	1,85%	1,29%	1,74%

Tabelle 5.1: Akzeptanzrate für ausgewählte Recommender nach Nutzertyp (Dezember 2003 - April 2004)

commender „New“ (Typ: (–,–,–)) und „NewCat“ (Typ: (+,+,–)) nicht der Fall. Der Recommender „CF“ des letzten Recommender-Typs (+,+,+) konnte zwar in vielen Fällen ausgeführt werden, lieferte aber in den meisten Fällen keine Ergebnisse. Der Grund liegt hier in den relativ wenigen Daten, die für die Berechnung von Assoziationsregeln eingeschränkt auf das Nutzungsverhalten ähnlicher Nutzer zur Verfügung stehen. Um irrelevante Recommendations zu vermeiden, wurden minimale Schwellwerte für die Berechnung der Assoziationsregeln gesetzt, die in vielen Fällen nicht erreicht wurden.

Tabelle 5.1 zeigt die ermittelten Akzeptanzraten für fünf ausgewählte Recommender-Klassen in Abhängigkeit vom Nutzertyp (neue vs. wiederkehrende Nutzer). Wie erwartet zeigen sich große Unterschiede zwischen den Recommendern. Für die verwendete Website erreichte der Suchmaschinen-Recommender (SER) die beste durchschnittliche *Acceptance Rate* (2,77%), gefolgt von den Recommendern, welche die Ähnlichkeit zwischen Seiten berechnen (Sim) sowie häufig innerhalb einer Session gemeinsam auftretende Seiten verwenden (Asso). Auf der anderen Seite erreichen einfache Verfahren, z. B. die Empfehlung der am häufigsten aufgerufenen Seiten (Top-Recommender), schlechtere Ergebnisse. Für eine genauere Analyse muss zusätzlich berücksichtigt werden, dass einzelne Recommender nur innerhalb bestimmter Kontexte anwendbar sind. Der Recommender „PersInt“ setzt z. B. einen wiederkehrenden Nutzer voraus, da sein persönliches bisheriges Navigationsverhalten verwendet wird. Andererseits benötigt die Anwendung des SER zwingend einen Nutzer, der mittels eines Suchmaschine die Website erreicht hat, was in ca. 95% der Fälle einen neuen Nutzer nach sich zieht.

Tabelle 5.1 zeigt, dass neue Nutzer häufiger Empfehlungen anklicken als wiederkehrende. Dies gilt (abgesehen von „PersInt“, der für neue Nutzer nicht angewendet werden kann) für alle Recommender außer dem Top-Recommender, der allerdings auch die schlechteste durchschnittliche *Acceptance Rate* zeigt. Offenbar kennen wiederkehrende Nutzer (z. B. Studenten) die Website und finden die für sie relevanten

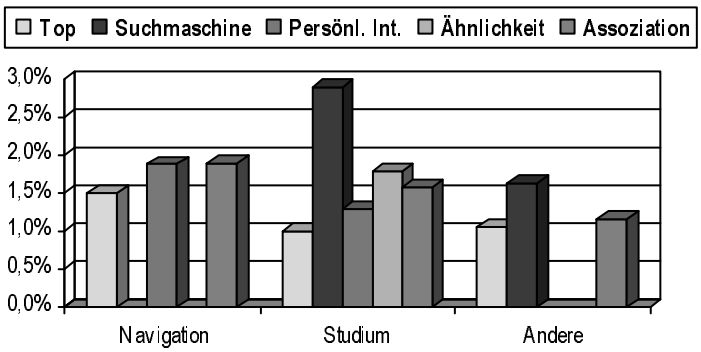


Abbildung 5.4: Akzeptanzrate für Recommender nach Seitentyp (Dezember 2003 - April 2004)

Informationen (z. B. Vorlesungsunterlagen) direkt ohne zusätzliche Hilfestellung. Am höchsten ist die Akzeptanzrate beim ersten Pageview einer Session (5,77%). Weitere innerhalb der Session besuchte Seiten erzielen lediglich eine durchschnittliche *Acceptance Rate* von 1,35%. Dabei wird der letzte Pageview nicht betrachtet, da dieser stets eine *Acceptance Rate* von 0 aufweist.

Abbildung 5.4 vergleicht die Recommender bezüglich des Typs der aktuellen Seite, wobei analog zu 5.1 nur Ergebnisse mit einem minimalen Support von 5% betrachtet werden. Während der SER-Recommender die besten Ergebnisse für *Studium* und *Andere* erzielt, erreichen die Recommender „PersInt“ sowie „Asso“ die besten Akzeptanzraten für Navigationsseiten. Ist der SER-Recommender nicht anwendbar, d. h. der Nutzer kommt nicht von einer Suchmaschine, liefern für Seiten des Typs „Studium“ der Ähnlichkeits-Recommender und auf Assoziationsregeln basierende Recommender die besten Ergebnisse.

Alle aufgeführten Beobachtungen sind natürlich spezifisch für die verwendete Website und daher nicht übertragbar auf andere Websites. Trotzdem motivieren sie die dynamische Recommender-Auswahl, da in Abhängigkeit vom Kontext die Qualität der von den verschiedenen Recommendern gegebenen Recommendations variiert. AWESOME ermöglicht durch die Verwendung des Data Warehouses eine sorgfältige OLAP-Analyse zur manuellen Bestimmung, welcher Recommender bei welchen Kontexten zur Anwendung kommen soll. Dies ist u. a. die Grundlage für die Definition manueller Selektionsregeln zur adaptiven Recommender-Selektion (siehe Abschnitt 4.2.1).

5.3.2 Recommender-Tuning

Recommender ermitteln ihre Recommendations nach einem bestimmten Berechnungsverfahren, das meist durch verschiedene Parameter beeinflusst werden kann. Als Beispiel sollen im Folgenden die Recommender „Top“ und „Asso“ betrachtet

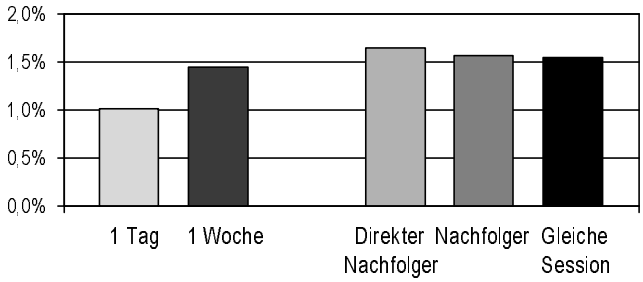


Abbildung 5.5: Durchschnittliche *Acceptance Rate* für Top-Recommender (links) und Asso-Recommender (rechts) mit unterschiedlichen Parametern (Dezember 2003 - April 2004)

werden (siehe Tabelle 3.4). Der Top-Recommender ermittelt die am häufigsten betrachteten Seiten der Website. Dabei kann der betrachtete Zeitraum der Webnutzung variiert werden, d. h. ob die am häufigsten angeklickte Seite der letzten 24 Stunden oder der letzten Woche ermittelt wird. Für einen auf Assoziationsregeln basierenden Recommender kann definiert werden, ob er alle in der Session gemeinsam auftretenden Pageviews (*Session*), nur die nachfolgenden Pageviews (*Successor*) oder nur den direkten Nachfolger (*Direct*) betrachten soll. Für ein entsprechendes Tuning wurden Recommender zufällig zur Bestimmung der Recommendations angewendet. Abbildung 5.5 zeigt die durchschnittliche *Accpetance Rate* der einzelnen Verfahren. Für den Top-Recommender ist es offenbar besser, das Nutzungsverhalten der letzten Woche statt nur der letzten Tages auszuwerten. Für die Berechnung der Assoziationsregeln ergibt die ausschließliche Verwendung der direkten Nachfolger die besten Ergebnisse, obgleich keine großen Unterschiede in der Qualität zu erkennen sind.

Die Ergebnisse sind natürlich ebenfalls spezifisch für die verwendete Website und somit keinesfalls auf andere Websites übertragbar. Das Beispiel zeigt jedoch das prinzipielle Vorgehen, wie für eine konkrete Website ein Recommender-Tuning durchgeführt werden kann. Das Vergleichen der Akzeptanzraten (oder anderer Metriken) mehrerer Recommender, die sich nur in einem oder wenigen Parametern unterscheiden, ermöglicht die Charakterisierung des Einflusses der veränderten Parameter. Auf Grund der Ergebnisse der OLAP-Analyse kann die Parameterbelegung optimiert werden.

5.4 Evaluation der Selektionsregeln

Zur Evaluation der Selektionsstrategien wurden alle Strategien parallel getestet. Jeder Session wurde zufällig eine Selektionsstrategie zugewiesen, so dass eine gleichmäßige Verteilung der Strategien gewährleistet wurde. Da die verwendete Strategie auch im Recommendation-Logfile aufgezeichnet wird (siehe Abschnitt 3.3.2), kann eine vergleichende Evaluation der Strategien durchgeführt werden. Neben den Re-

Strategie	Anzahl der Regeln	Acceptance Rate	Session Accept. Rate
Anfragebasiert	$\sim$ 2000	1,35%	10,27%
Entscheidungsbaum	$\sim$ 250	1,74%	11,13%
Hybrid	$\sim$ 250	1,71%	10,93%
Manuell 1	25	1,01%	7,56%
Manuell 2	5	1,97%	12,54%
Zufällig	105	0,96%	6,98%
Reinf. Learning	$\sim$ 60000	1,92%	11,22%
Reinf. Learning Zero	$\sim$ 60000	1,87%	10,35%

Tabelle 5.2: Vergleich aller Selektionsstrategien (Januar - Juni 2004 bzw. April - September 2004 (für Reinf. Learning))

commender-Selektionsstrategien, deren Daten vom 1. Januar bis 30. Juni 2004 aufgezeichnet wurden, sind zum Vergleich auch die einstufigen – mittels Reinforcement Learning erstellten – Recommendation-Regeln angegeben (siehe [189] für eine kurze Übersicht sowie [71, 72] für Details), deren Evaluationszeitraum vom 1. April bis 30. September 2004 lag.

5.4.1 Vergleich aller Strategien

Tabelle 5.2 zeigt eine Übersicht aller Selektionsstrategien. Die Anzahl der Regeln für die Recommender-Selektion ist relativ klein, lediglich die anfragebasierte Strategie erzeugt ca. 2000 Regeln. Im Gegensatz dazu erzeugen die einstufigen Verfahren ca. 60000 Regeln pro Strategie. Nichtsdestoweniger benötigt in der prototypischen Implementierung die Selektion des Recommenders bzw. der Recommendations anhand der Regelmenge stets unter 100ms. Die adaptive Selektion der Recommender führt damit zu keiner fühlbaren Verzögerung für die Website-Nutzer. Weiterhin ist in Tabelle 5.2 die durchschnittliche *Acceptance Rate* sowie *Session Acceptance Rate* angegeben. Alle automatischen Strategien haben deutlich bessere Akzeptanzraten als *Manuell 1* und *Zufällig*. Bei den Recommender-Strategien besitzen die *Entscheidungsbaum*-Strategie sowie *Manuell 2* die höchsten Akzeptanzraten.

Die Erstellung der zweiten manuellen Strategie setzte zunächst eine umfangreiche manuelle Webnutzungsanalyse voraus, um typische Nutzergruppen zu identifizieren (siehe Abschnitt 5.2). Anschließend wurden mit Hilfe der Recommender-Evaluation (siehe Abschnitt 5.3) entsprechende Selektionsregeln abgeleitet. Dabei wurde auch Hintergrundwissen bezüglich der Website verwendet, z. B. dass typische Nutzer einer akademischen Seite Studenten und Forscher sind. Die Tatsache, dass die automatische Generierung mittels maschinellen Lernens, d. h. Entscheidungsbaumverfahren

sowie Reinforcement Learning, ähnliche Akzeptanzraten erreicht, ist daher ein sehr gutes Ergebnis. Es zeigt die Machbarkeit einer automatischen Closed-Loop-Optimierung und den hohen Wert des aufgezeichneten Recommendation-Feedbacks. Dadurch konnte die Recommendation-Qualität maßgeblich ohne manuellen Aufwand gesteigert werden.

Die Analyse der manuellen Strategien zeigt auch die Notwendigkeit einer ausführlichen Recommender-Evaluation im Vorfeld der Definition von Selektionsregeln. Während bei der ersten Strategie lediglich auf einfaches Hintergrundwissen zurückgegriffen wurde („Wiederkehrende Nutzer interessieren sich stets für die gleichen Bereiche der Website."), wurden bei der zweiten Strategie für relevante Nutzergruppen die besten Recommender bestimmt.

Der Vergleich der beiden automatischen Recommender-Selektionsstrategien zeigt, dass die Verwendung maschinellen Lernens deutlich bessere Ergebnisse zeigt als der anfragebasierte Ansatz. Auffällig ist, dass trotz der geringen Anzahl der Regeln beim maschinellen Lernen – gegenüber dem anfragebasierten Ansatz – deutlich bessere Akzeptanzraten resultieren. Offenbar gelingt es, beim maschinellen Lernen die Kontextattribute zu gewichten und nur relevante Informationen zur Recommender-Selektion heranzuziehen. Wichtige Attribute befinden sich beim Entscheidungsbaumverfahren in den oberen Ebenen, wohingegen unwichtige weiter unten zu finden sind. Durch das Beschneiden des Baumes mit der anschließenden Transformation in die Selektionsregeln treten daher nur relevante Kontextattribute in den Regeln auf. Auf der anderen Seite behandelt der anfragebasierte Ansatz alle Attribute gleichwertig und eliminiert lediglich Attributinformationen, um einen minimalen Support der Regeln zu gewährleisten. Dadurch basiert die Auswahl der Recommender häufiger auf irrelevanten Kontextinformationen, was zu schlechteren Akzeptanzraten führt.

Der Einsatz der hybriden Strategie konnte leider nicht zur erhofften Verbesserung der Akzeptanzraten beitragen. Die erzielten Ergebnisse entsprachen in etwa denen des automatischen Entscheidungsbaumansatzes. Offenbar hatte das Entscheidungsbaumverfahren die bei der hybriden Strategie manuell hinzugefügten Regeln so oder in ähnlicher Form auch ermittelt, so dass die Hinzunahme der manuellen Regeln zu keiner Verbesserung führte.

In den durchgeführten Experimenten konnte bei den automatischen Strategien der einstufige Ansatz auf Basis des Reinforcement Learnings etwas bessere Ergebnisse als der zweistufige Ansatz erzielen. Dabei ist eine höhere *Acceptance Rate* zu verzeichnen, wobei die *Session Acceptance Rate* kaum Unterschiede aufweist. Die höhere Effektivität des einstufigen Ansatzes lag primär an der Online-Verwendung des aktuellen Feedbacks, die eine Online-Optimierung der Regeln ermöglicht. Der zweistufige Ansatz sieht dagegen ausschließlich eine Offline-Berechnung der Regeln (z. B. einmal pro Tag) vor, so dass aktuelles Feedback nicht direkt bei der Recommender-Auswahl eingeht, sondern erst bei der nächsten Neuberechnung der Regeln.

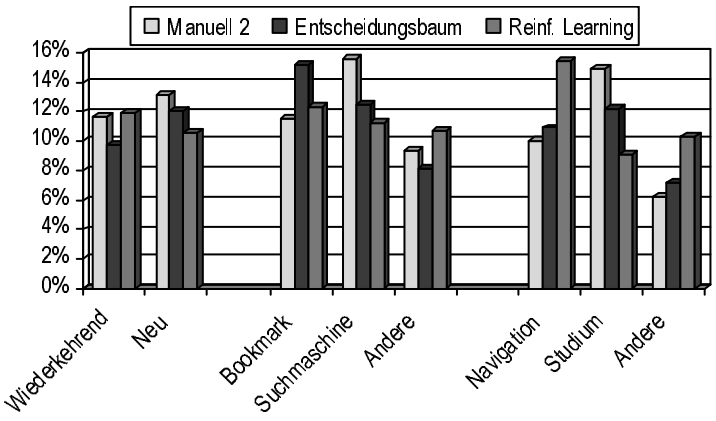


Abbildung 5.6: Vergleich der Selektionsstrategien für verschiedene Nutzergruppen

5.4.2 Kontextabhängige Evaluation der Selektionsstrategien

Die Data-Warehouse-Infrastruktur von AWESOME erlaubt eine kontextbasierte Evaluation der Selektionsstrategien (analog zur Recommender-Evaluation in Abschnitt 5.3.1). Abbildung 5.6 zeigt die *Session Acceptance Rate* der besseren manuellen Strategie, der automatischen Regelgenerierung mittels Entscheidungsbaumverfahren sowie dem Reinforcement-Learning-Verfahren. Dabei werden unterschiedliche relevante Kontextmuster betrachtet, d. h. Nutzertyp, Referrer-Information und Einstiegsseite. Bei letzterer wird eine Session danach charakterisiert, in welche Kategorie die erste aufgerufene Seite (Einstiegsseite) der Session fällt.

Die manuelle Strategie ist besonders effektiv für Nutzer, die von einer Suchmaschine kommen. In diesem Fall verwendet die manuelle Strategie stets den SER-Recommender. Dies unterstützt auch die guten Ergebnisse für neue Nutzer und Sessions, die mit einer Seite des Typs „Studium“ beginnen, da solche Sessions häufig mit der Verwendung einer Suchmaschine einhergehen. Auf der anderen Seite liefern die automatischen Strategien bessere Ergebnisse insbesondere für Nutzer, die keine Suchmaschine verwendet haben, um die Website zu erreichen, sowie für Nutzer, die mit einer Navigationsseite ihren Website-Besuch beginnen. Die Ergebnisse zeigen, dass automatisch generierte Regeln in vielen Fällen eine optimale Bestimmung der Recommender ermöglichen. Die manuelle Erstellung von Regeln ist auf der anderen Seite sehr aufwändig (z. B. mit vorgeschalteter OLAP-Analyse), kann jedoch Hintergrundwissen, z. B. über typische Nutzertypen der Website, verwenden.

5.5 Zusammenfassung

In diesem Kapitel wurde am Beispiel einer konkreten Website eine Evaluation des Webnutzungsverhaltens sowie der Recommendation-Qualität durchgeführt. Dabei

wurde insbesondere das prinzipielle Vorgehen mit Hilfe einer OLAP-Analyse illustriert, wobei verschiedene Arten der Evaluation identifiziert und vorgestellt wurden. Dabei ist es sowohl möglich, das Webnutzungsverhalten als auch die Recommender und die angewendeten Selektionsstrategien effektiv zu evaluieren. Die Ergebnisse zeigen u. a. die Abhängigkeit der Recommendation-Qualität verschiedener Recommender vom aktuellen Kontext. Gleichzeitig konnte für die Beispiel-Website gezeigt werden, dass eine automatische und adaptive Auswahl der Recommender die Recommendation-Qualität signifikant steigern kann.

Teil III

Mapping-basierte Informationsfusion

6 Datenintegration

Dieses Kapitel gibt eine Einführung in das Thema Datenintegration. Nach einer kurzen Motivation (Abschnitt 6.1), die insbesondere die Bedeutung und Schwierigkeiten der Datenintegration illustriert, werden bisherige Datenintegrationsansätze vorgestellt (Abschnitt 6.2). Anschließend wird ein Abriss relevanter Literatur zum Object Matching, einem wichtigen Teilaspekt der Datenintegration, gegeben (Abschnitt 6.3). Das Kapitel wird durch die Darstellung offener Probleme abgeschlossen (Abschnitt 6.4).

6.1 Motivation

Datenintegration[33] beschreibt das „korrekte, vollständige und effiziente Zusammenführen“ [135] von Informationen aus verschiedenen Datenquellen mit dem Ziel eines einheitlichen Zugangs zu den Daten.[34] Erst die Verknüpfung von Informationen aus mehreren verteilten, autonomen und heterogenen Datenquellen ermöglicht in vielen Fällen weitgehende Analysemöglichkeiten, z. B. im Bereich der Bioinformatik. Kernproblem der Datenintegration ist die Überwindung der Unterschiede zwischen den Datenquellen. Das beinhaltet sowohl strukturelle Unterschiede, d. h. unterschiedliche semantische Modellierungen des gleichen Sachverhalts, als auch Unterschiede in

[33] *lat.* integratio = Wiederherstellung eines Ganzen; *lat.* integrare = wiederherstellen, ergänzen, ein Ganzes ausmachen

[34] In dieser Arbeit wird mit Datenintegration der in [113] und [135] verwendete Begriff Informationsintegration gemeint.

den Daten selbst, z. B. durch Datenfehler oder unterschiedliche Repräsentationsformen [80]. Die Größe und Vielzahl der Datenquellen erfordert dabei die Anwendung automatischer Verfahren, was das Thema Datenintegration zu einem stark beachteten Forschungsgebiet der letzten Jahre macht [85]. Im Folgenden werden Bedeutung sowie Probleme und Anforderungen der Datenintegration skizziert.

6.1.1 Bedeutung

Durch den in den letzten Jahren stark gestiegenen Einfluss des Internets stehen immer mehr Daten zur Verfügung. Wie bereits in der Einführung auf Seite 13 erwähnt, gibt es mehr als 60 Millionen aktive Websites im World Wide Web gibt (Stand: Oktober 2007) – mit stark steigender Tendenz. Gleichzeitig erhöht sich die Menge der Informationen innerhalb einzelner Websites stetig. Als Beispiel sei die Website der DBLP Bibliography [2] genannt, welche monatlich um ca. 10.000 neue Publikationseinträge erweitert wird [197]. Auf der anderen Seite ist die heutige Welt von einer immer stärkeren Spezialisierung geprägt, so dass nicht jeder alle nötigen Informationen selbst generieren kann und daher auf das Wiederverwenden von Daten anderer angewiesen ist [176].

Insbesondere in der heutigen Wissensgesellschaft entsteht Wissen erst durch die Verknüpfung von Informationen aus verschiedenen Quellen. Daher ist Datenintegration eines der anhaltendsten Probleme innerhalb der Datenbankforschung und hat sowohl für die Wirtschaft als auch für die Wissenschaft eine enorme Bedeutung [79]. Während es im wirtschaftlichen Umfeld primär um finanzielle Interessen geht, z. B. schnellere Geschäftsentscheidungen durch schnelleren Zugriff auf alle relevanten Geschäftsdaten, wird die Forschung von den vielen Datenquellen im Web angetrieben, deren Integration zu einem effizienteren Umgang mit den verfügbaren Informationen führt. Exemplarisch werden daher im Folgenden je zwei Beispiele aus den Bereichen Wirtschaft und Wissenschaft näher vorgestellt, um die Bedeutung der Datenintegration darzustellen.

Datenintegration in der Wirtschaft

Am Beispiel des Bonusprogramms Payback[35] zeigt sich die wirtschaftliche Bedeutung der Datenintegration. Payback-Kunden erhalten eine nutzerbezogene Kundenkarte, welche bei jedem Einkauf bei einem der Partnerunternehmen vorgelegt wird. Die Zuordnung der getätigten Einkäufe zur kaufenden Person ermöglicht eine einfache Integration der Verkaufsdaten der beteiligten Unternehmen. Es lassen sich so Nutzerprofile erstellen, die neben persönlichen Daten (u. a. Alter, Geschlecht, Wohnort) auch sozio-ökonomische Informationen enthalten, also z. B. der durchschnittliche monatliche Kaufumsatz des Kunden. Gleichzeitig können detaillierte Analysen

[35] http://www.payback.de

durchgeführt werden, z. B. in welchen Branchen der Kunde vorrangig einkauft oder ob er eher hoch- oder niedrigpreisige Produkte bevorzugt. Aus diesen Informationen lässt sich u. a. personalisierte Werbung erstellen, die dem Kunden z. B. per Post in Form von Prospekten oder Coupons zugeschickt werden kann. Dabei kann die Analyse des nach einer Werbungsaktion erfolgten Kaufverhaltens sofort Rückschlüsse auf den Erfolg der Aktion geben.

Ein zweites Beispiel aus dem Bereich Wirtschaft umfasst die Verarbeitung von Namen und Adressdaten, welche Unternehmen bei der Erfassung der persönlichen Angaben von Personen speichern. Dabei kann die gleiche Person mehrfach in den Datenbeständen eines Unternehmens auftreten, z. B. wenn Name und Adresse unabhängig voneinander bei einem Vertragsabschluss und einer Gewinnspielteilnahme aufgenommen wurden. Durch Schreibfehler oder unterschiedliche Repräsentationen der gleichen Information, z. B. „Straße“ vs. „Str.“ oder Angabe der Telefonnummer mit und ohne Vorwahl, ist es daher leicht möglich, dass für die gleiche Person unterschiedliche Datensätze vorliegen. Die Identifikation und Zusammenführung solcher Datensätze ist von wichtiger Bedeutung für das Customer-Relationship-Management, da sonst unnötige Kosten, z. B. durch Mehrfachversand von Postsendungen oder durch unnötige Werbung für Produkte, die der Kunde bereits besitzt, entstehen. Daher ermöglichen spezialisierte Tools, z. B. Trillium[36], die automatische Erkennung unterschiedlicher Datensätze, welche die gleiche reale Person referenzieren.

Datenintegration in der Wissenschaft

Mit der Bioinformatik hat sich in den letzten Jahren ein Zweig der Informatik etabliert, der sich mit der Verarbeitung der in den Lebenswissenschaften anfallenden Daten beschäftigt. Eine treibende Kraft für die Erzeugung immer neuer Daten ist z. B. die Sequenzierung von Genomen [196]. Forschungsgruppen auf der ganzen Welt veröffentlichen ihre Ergebnisse zu den von ihnen untersuchten molekularbiologischen Objekten (Gene, Proteine, etc.) sowie deren Funktionen, Wechselwirkungen und andere Eigenschaften innerhalb von Webdatenbanken als sogenannte Annotationen. Eine aktuelle Studie [65] listet mehr als 800 solcher Datenquellen auf. Die Verknüpfung dieser Daten spielt eine wichtige Rolle bei der Interpretation und Überprüfung der Ergebnisse. Zusätzlich ermöglicht die Auswertung der in unterschiedlichen Datenquellen gespeicherten Objekte die Erstellung neuer Zusammenhänge zwischen ihnen. Solche Analyseergebnisse können wiederum Ausgangspunkt für weitere biologische Forschungen sein, um z. B. die Art oder Funktionsweise des ermittelten Zusammenhangs zu untersuchen.

Wesentlicher Bestandteil der Forschungsarbeit ist die Auseinandersetzung mit wissenschaftlichen Publikationen, die u. a. bei speziellen bibliografischen Websites, z. B. der DBLP Bibliography [2], sowie auf den unzähligen Homepages der Wissenschaftler

[36] http://www.trilliumsoft.com

zu finden sind. Allerdings gibt es keine *beste* Datenquelle, die alle verfügbaren Informationen in bester Datenqualität bereithält. Auf der Suche nach relevanter Literatur sind insbesondere häufig zitierte Überblicksartikel zu einem Themengebiet oder die Liste der zitierenden Arbeiten für eine ausgewählte Publikation X („Welche Publikationen Y zitieren X?“) von Interesse. Das Erfassen solcher Zitierungen wird u. a. von der Suchmaschine Google Scholar [3] realisiert. Zusätzlich gibt es Datenquellen, die sich auf ein spezielles Forschungsgebiet konzentrieren, z. B. der Publication Categorizer für Schema Evolution [155], und bei denen wissenschaftliche Beiträge (manuell) in verschiedene Kategorien eingeordnet sind. Erst durch die Kombination verschiedener Datenquellen – DBLP für sehr gute bibliografische Angaben, Google Scholar für Zitierungen, Publication Categorizer für thematische Kategorisierungen – stehen alle für Publikationen relevanten Informationen zur Verfügung. Durch die Integration der Datenquellen entsteht für den Nutzer ein wichtiger Mehrwert, der ihm u. a. eine effiziente und zielgerichtete Literaturrecherche ermöglicht.

6.1.2 Anforderungen und Schwierigkeiten

Die zu integrierenden Daten befinden sich in verschiedenen *verteilten* Datenquellen, wobei jede Datenquelle *autonom*, d. h. selbständig, ist. Es ergeben sich daher die zwei folgenden Probleme bei der Datenintegration [113].

Verteilung: Die Verteilung beschreibt die Tatsache, dass die zu integrierenden Datenquellen sowohl physisch als auch logisch verteilt sein können, was verschiedene Probleme nach sich zieht. Bei der physischen Verteilung muss der Ort der Datenquelle lokalisiert, die unterschiedliche Strukturierung der Daten in den einzelnen Quellen überwunden sowie die Zugriffe (über ein Netzwerk) auf die Datenquellen optimiert werden. Bei der logischen Verteilung muss u. a. das Problem der Redundanz (mehrfache Speicherung der gleichen Information) gelöst werden, was ansonsten zu Duplikaten und Widersprüchen zwischen den Datenquellen führen kann.

Autonomie: Mit der Autonomie der Datenquellen wird die Tatsache beschrieben, dass Datenquellen unabhängig über die von ihnen gespeicherten Daten verfügen können. Das betrifft u. a. die Art der Strukturierung, die Zugriffsmöglichkeiten für Dritte sowie das Recht auf Veränderung (Evolution). Bei der Erstellung eines Datenintegrationssystems können i. Allg. daher nicht die Datenquellen geändert werden, sondern das Datenintegrationssystem muss sich an die Eigenschaften, z. B. eingeschränkte Zugriffsmöglichkeiten, der Datenquellen anpassen. Als Konsequenz muss das Datenintegrationssystem bei Veränderungen der Datenquellen, z. B. bei einer Schemaänderung, ggf. angepasst werden.

Verteilung und Autonomie führen meist zum Hauptproblem der Datenintegration – der *Heterogenität* von Datenquellen, d. h. dass zwei (oder mehr) Datenquellen Unterschiede in der Repräsentation ihrer Daten aufweisen. Es lassen sich dabei im Wesentlichen die folgenden drei Arten von Unterschieden ausmachen [176] (in [113] werden noch weitere genannt).

Syntaktische Heterogenität reflektiert unterschiedliche Formate in der Darstellung von Informationen. Das Spektrum reicht hierbei von technischen Aspekten, z. B. unterschiedliche Zahlenformate oder Zeichenkodierungen, bis zu unterschiedlichen „gängigen" Repräsentationsformen, z. B. bei Datumsangaben („19. Oktober 2007", „19.10.07" oder „10/19/2007").

Strukturelle Heterogenität entsteht meist bei der unabhängigen Modellierung mehrerer Datenquellen durch verschiedene Personen. Unterschiedliche Modellierungsaspekte, z. B. die während der Modellierung intendierte primäre Verwendung der Datenquellen, führen zur Verwendung unterschiedlicher Abstraktionsebenen, Detailstufen oder Modellierungsformen (z. B. hierarchisch oder relational). Gibt es in einer Datenquelle z. B. die Produktypen Buch und DVD, in einer anderen die Typen Hardcover-Buch, Softcover-Buch, Spiele-DVD und Film-DVD, so ist das ein Beispiel für unterschiedliche Detailstufen bei der Modellierung.

Semantische Heterogenität beschreibt Probleme bei der Bedeutung bzw. Interpretation der in den Datenquellen verwendeten Begriffe. So können scheinbar gleiche Informationen unterschiedliche Bedeutungen haben, z. B. wenn der Preis eines Produkts in der einen Datenquelle die Lieferkosten enthält und in der anderen nicht. Andererseits können unterschiedliche Begriffe, z. B. „Titel" und „Name", in einigen Fällen die gleiche Bedeutung haben, z. B. bei der Beschreibung des Produktnamens.

Den genannten Schwierigkeiten bei der Datenintegration stehen die Anforderungen entgegen, dass die Integration korrekt, vollständig und effizient durchgeführt werden soll [113]. Auf Grund der Größe der Datenquellen kommen bei der Datenintegration automatische Verfahren zum Einsatz. Diese sollen auf der einen Seite flexibel einsetzbar sein, d. h. insbesondere in vielen Integrationsszenarien und für möglichst viele Datenquellen anwendbar sein. Gleichzeitig soll die Ausführung der Verfahren effizient sein, d. h. dass z. B. Nutzeranfragen unter Verwendung der integrierten Daten schnell beantwortet werden können.

Korrektheit und Vollständigkeit der integrierten Daten sind wichtige Faktoren für die Datenqualität, die wiederum ein entscheidender Aspekt von Datenintegrationssystemen ist [10]. Wie bereits geschildert, hat eine geringe Datenqualität, z. B. das mehrfache Auftreten der gleichen Person in einem Datenbestand, negative Folgen

für die weitere Verarbeitung, z. B. erhöhte Kosten durch Mehrfachversand von Postsendungen. Gleichzeitig beeinflusst die Datenqualität das Ergebnis bzw. die Aussagekraft daraus abgeleiteter Analysen, z. B. in der Bioinformatik („garbage in, garbage out“). Datenqualitätsprobleme treten sowohl innerhalb einer Quelle, z. B. durch Tippfehler oder Duplikate, als auch durch die Verbindung mit anderen Quellen, z. B. durch sich daraus ergebende Widersprüche in den Daten, auf [156, 142]. Das breite Spektrum der Probleme führt zu einer Vielzahl von Ansätzen zu ihrer Behebung (siehe u. a. [10] für eine Auswahl). Für das wichtige Problem des Object Matchings gibt Abschnitt 6.3 eine Übersicht der Verfahren. Neben automatischen Tools kann die Datenqualität auch durch entsprechende Richtlinien bei der manuellen Verarbeitung der Daten unterstützt werden. So wird z. B. innerhalb der DBLP Bibliography bei der manuellen Eingabe der Daten durch die „All or nothing“-Richtlinie sichergestellt, dass stets komplette Publikationslisten der Konferenzen und Journals vorliegen [161]. Zusätzlich gibt es semiautomatische Verfahren, die automatisch Hinweise während der Eingabe zu möglichen Fehlern präsentieren, z. B. wenn ein neu eingegebener Autor einen sehr ähnlichen Namen zu einem schon vorhandenem DBLP-Autoren hat.

6.2 Related Work: Ansätze zur Datenintegration

Datenintegration ist ein viel beachtetes Thema der Forschung, das in den letzten Jahren intensiv bearbeitet wurde [85, 209] und u. a. zu einer Vielzahl von Integrationssystemen, z. B. in der Bioinformatik [90], führte. Eine ausführliche Liste weltweiter Projekte zum Thema Datenintegration stellt [208] bereit. Die verschiedenen in der Literatur vorgestellten Ansätze zur Datenintegration lassen sich nach unterschiedlichen Kriterien einordnen bzw. differenzieren (siehe z. B. [50]). In [113] werden u. a. die folgenden drei Eigenschaften genannt:

Bottom-up- oder Top-down-Entwurf unterscheidet die Strategien beim Entwurf eines Datenintegrationssystems. Der Bottom-up-Entwurf beginnt mit einer festen Menge bestehender und bekannter Quellen, während beim Top-down-Entwurf die Datenquellen nach dem Informationsbedarf schrittweise hinzugefügt werden.

Anfrageparadigma definiert die Art der unterstützten Anfragen, z. B. ob beliebige strukturierte Anfragen (z. B. mit SQL oder XQuery) oder nur Suchanfragen mit einzelnen Suchbegriffen möglich sind.

Read-only oder Read-and-Write beschreibt, ob das Datenintegrationssystem die Daten der einzelnen Quellen manipulieren kann (Einfügen, Ändern, Löschen) oder ob ausschließlich ein Lesezugriff unterstützt wird.

Innerhalb dieses Abschnitts wird eine andere Art der Klassifikation gewählt, welche sich nach der Art der Integration von Meta- und Instanzdaten richtet. Strukturierte Datenquellen besitzen ein Schema (Metadaten), welches den Aufbau der gespeicherten Informationen (Instanzdaten) beschreibt. Beispiele für Schemata sind relationale Datenbankschemata oder XML-Schemata; die zugehörigen Instanzdaten umfassen die einzelnen Datensätze bzw. XML-Dokumente. Im Rahmen der Integration mehrerer Quellen wird dabei häufig ein Abgleich zwischen den Schemata vorgenommen, um semantisch gleiche Schemaelemente oder (Teil-) Strukturen zu identifizieren (Schema Matching, siehe [154] für eine Übersicht). Das Ergebnis ist ein Schema Mapping, das Korrespondenzen zwischen semantisch gleichen Schemaelementen (z. B. Tabellen und Attribute des relationalen Schemas) beinhaltet. Sollen mehrere Quellen integriert werden, kann ein globales Schema verwendet werden, so dass jedes Quellenschema ein Schema Mapping zum globalen Schema besitzt.

Folgende Arten der Schema- und Instanzintegration werden unterschieden:

Schemaintegration: Bei der Integration der Schemata wird in vielen Fällen ein globales Schema verwendet, so dass sich Integrationsansätze bezüglich der folgenden drei Fälle unterscheiden lassen:

- Ansätze mit einem *domänenspezifischen globalen Schema*, d. h. ein speziell für die Domäne erstelltes Schema wird verwendet, welches alle in der Domäne relevanten Schemainformationen repräsentiert
- Ansätze mit einem *generischen globalen Schema*, die Schemainformationen in einer generische Struktur ablegen
- Ansätze, die *kein globales Schema* verwenden

Integration der Instanzdaten: Es lassen sich für die Integration der Instanzdaten die folgenden drei Fälle unterscheiden:

- *Physische Integration*, d. h. die Instanzdaten der einzelnen Quellen werden materialisiert an einer zentralen Stelle (als Kopie) gespeichert
- *Virtuelle Integration*, d. h. die Instanzdaten verbleiben ausschließlich in den Datenquellen und werden erst bei Anfragen gezielt angefordert
- *Hybride Integration*, d. h. eine Mischform der beiden oben genannten Formen, bei der Teile der Daten physisch und andere Teile virtuell integriert werden

Die im Folgenden vorgestellten Datenintegrationsansätze werden in den nächsten drei Abschnitten bezüglich der Art der Schemaintegration charakterisiert. Dabei wird innerhalb jedes Abschnittes jeweils auf die Unterscheidung zwischen physischer, virtueller und hybrider Integration der Instanzdaten eingegangen.

6.2.1 Datenintegration mit Hilfe eines domänenspezifischen globalen Schemas

Die Verwendung eines domänenspezifischen globalen Schemas erfordert zunächst dessen Erstellung und anschließend eine Verknüpfung der Quellenschemata mit dem globalen Schema. Dabei lassen sich mit *top-down* und *bottom-up* im Wesentlichen zwei Vorgehensweisen unterscheiden.

Beim Top-Down-Ansatz wird das globale Schema unabhängig von den konkreten Datenquellen erstellt, sondern entsteht aus Anforderungen der späteren Nutzung bzw. Eigenschaften der betrachteten Domäne. Dieser Ansatz ist besonders dann sinnvoll, wenn zur Erstellungszeit nicht bekannt ist, welche Quellen im Laufe der Zeit integriert werden sollen. Bei der anschließenden Schemaintegration wird jedes Quellenschema als eigene Sicht auf das globale Schema aufgefasst, d. h. das Quellenschema wird durch eine Transformation des globalen Schemas, z. B. durch einen SQL-View, repräsentiert („local as view“ [78]).

Im Gegensatz dazu entsteht beim Bottom-Up-Ansatz das globale Schema als Ergebnis der Analyse aller zu integrierenden Quellenschemata. Dabei werden mittels Schema Matching korrespondierende Elemente zwischen den Quellenschemata ermittelt und das globale Schema als vollständige, redundanzfreie und konsistente Sicht auf die einzelnen Quellenschemata erzeugt. Dabei wird für jedes Quellenschema das globale Schema als Sicht bzgl. des jeweiligen Quellenschemas dargestellt („global as view“ [195]).

Der Einsatz eines Data Warehouses [92] führt zu einer materialisierten Integration der Datenquellen, d. h. dass die Instanzdaten physisch im Warehouse (als Kopie) abgelegt werden. Der Import der Daten geschieht im Rahmen eines ETL-Prozesses (Extract, Transform, Load), der typischerweise der aufwändigste Teil eines Data-Warehouse-Projektes ist. Im ersten Schritt, der Extraktion, werden die relevanten Teile aus den Datenquellen selektiert, die anschließend bei der Transformation aufbereitet werden. Es erfolgen dabei u. a. eine Anpassung der Quellenschemata an das globale Schema des Warehouses, eine Identifikation gleicher Instanzen (Object Matching, siehe Abschnitt 6.3) sowie die Erkennung und Behandlung von Datenfehlern. Im dritten Schritt, dem Laden, werden die transformierten Daten in das Data Warehouse eingebracht, wobei ggf. entsprechende Aggregationen durchgeführt werden. Ein Beispiel für einen ETL-Prozess wird in Kapitel 3.4 im Rahmen des AWESOME-Ansatzes gegeben. Weitere Beispiele für die Anwendung von Data Warehouses sind z. B. Semex [27, 51] im Bereich des Personal Information Managements oder Integrated Genomic Database (IGD) [162] für Gendaten. Die Erstellung von Data Warehouses wird durch moderne Datenbankmanagementsysteme unterstützt, wobei insbesondere die Erstellung des ETL-Prozesses durch eingebaute Datenverarbeitungskomponenten, z. B. beim Mircrosoft SQL Server 2005 [31], vereinfacht wird. Die materialisierte Integration der Daten ermöglicht eine performante Analyse und

Weiterverarbeitung der Daten. Um stets die aktuellen Daten im Warehouse bereitstellen zu können, muss dafür der Datenbestand durch den ETL-Prozess regelmäßig aktualisiert werden.

Bei der virtuellen Integration durch Query-Mediator-Architekturen [201] stellt sich das Aktualisierungsproblem nicht. Die Integration der Instanzdaten erfolgt im Gegensatz zum Data-Warehouse-Ansatz erst bei der Ausführung von Anfragen an den Mediator, so dass stets die aktuellen Daten der Datenquellen verwendet werden. Die bezüglich des globalen Schemas gestellte Anfrage wird durch den Mediator in Anfragen für die einzelnen Quellen reformuliert, wozu es je nach Art der Schemaintegration unterschiedliche Verfahren gibt [111]. Bei GAV (global as view) kann die View-Definition des globalen Schemas einfach in die Anfrage „eingesetzt“ werden, ähnlich der Verwendung einer relationalen Sicht innerhalb einer SQL-Anfrage. Die Anfragereformulierung gestaltet sich bei LAV (local as view) wesentlich komplexer, da zur Verarbeitung der Anfrage nur die View-Definitionen für die jeweiligen Quellenschemata zur Verfügung stehen. Ein Kombination von GAV und LAV bildet GLAV [61], das die Vorteile beider Ansätze (einfache Anfrageverarbeitung bei GAV, einfaches Hinzufügen neuer Quellen bei LAV) vereint. Nach der Ausführung der Anfragen bei den einzelnen Quellen werden die Anfrageergebnisse entsprechend des globalen Schemas konvertiert. Anschließend kann eine Duplikaterkennung erfolgen, um semantisch gleiche Ergebnissätze aus den verschiedenen Quellen zu identifizieren (siehe Abschnitt 6.3). Ausgewählte Beispiele für Query-Mediator-Ansätze sind SIMS/Ariadne [5], Disco [193], Garlic [28], Information Manifold [115] sowie speziell aus dem Bereich der Integration biologischer Datenquellen TAMBIS [68] und BioMediator [173]. Für weitere Projekte sei auf [208] verwiesen.

Hybride Ansätze, wie z.B. DataFoundry [42], kombinieren beide Ansätze, indem z.B. häufig verwendete Daten zusätzlich als Kopie innerhalb eines Warehouses gespeichert werden.

6.2.2 Datenintegration mit Hilfe eines generischen globalen Schemas

Der große Vorteil aller Ansätze mit domänenspezifischem globalen Schema liegt im einfachen und einheitlichen Zugriff auf die Daten. Anfragen werden gegenüber dem globalen Schema gestellt und durch das System beantwortet. Nachteilig ist jedoch der Aufwand zur Erstellung des globalen Schemas sowie für das anschließende Transformieren und ggf. Importieren der Daten. Des Weiteren kann die Hinzunahme einer neuen Quelle zu einer Anpassung der globalen Schemas führen. Insbesondere für große Schemata und beim Vorhandensein vieler Datenquellen skaliert dieser Ansatz nicht sehr gut, weshalb andere Datenintegrationsansätze ein generisches globales Schema verwenden. Dieses kann entweder inkrementell mit jeder neuen Datenquelle erweitert werden oder verwendet eine generische Datenrepräsentation, in die sich

jedes Quellenschema einfach überführen lässt.

Mit GenMapper [48] wurde eine Data-Warehouse-basierte Architektur zur Speicherung biologischer Daten (Annotationsdaten) vorgestellt. Es verwendet ein generisches Datenmodell (GAM, Generic Annotation Model), welches die Speicherung beliebiger Annotationsdaten zulässt. Die Datenquellen werden physisch in ein Data Warehouse integriert und die bestehenden Beziehungen zwischen den (biologischen) Objekten werden als Korrespondenzen gespeichert. Für die Analyse bietet GenMapper die Bildung von Annotationssichten aus dem generischen Datenmodell an.

Als Alternative zur Transformation der Quellenschemata in ein generisches Datenmodell kann das globale Schema auch dadurch generisch erzeugt werden, indem einfach die Vereinigung der Quellenschemata gebildet wird. Dieser Ansatz wird z. B. in Columba [165], Aladin [112], SRS [58] und DBGET/LinkDB [62] verfolgt. Die Integration erfolgt anschließend über Instanzkorrespondenzen zwischen den Objekten der einzelnen Quellen, welche durch ihren Primärschlüssel identifiziert werden. Dabei kann eine Pivot-Datenquelle verwendet werden (z. B. die Protein Data Bank bei Columba), zu der Korrespondenzen von allen anderen Quellen gebildet werden. Der Humboldt Merger (HumMer [20]) importiert alle Quellen mit ihrem Originalschema in eine Datenbank und erlaubt dann mittels Schema-Mappings sowie einer entwickelten SQL-Erweiterung Anfragen, die mehrere Quellen umfassen. Innerhalb der resultierenden Daten werden Duplikate erkannt und Attributwerte können nutzerdefiniert fusioniert werden, um z. B. widersprüchliche Angaben in verschiedenen Quellen aufzulösen.

DiscoveryLink [77] verfolgt ebenfalls die Idee, als globales Schema die Vereinigung der lokalen Quellenschema zu verwenden. Im Unterschied zu den obigen Ansätzen, die eine physische Integration der Instanzdaten vornehmen, stellt DiscoveryLink eine sogenannte föderierte Datenbank [120] dar. Nutzer können Anfragen unter Verwendung mehrerer Quellen stellen, welche in Teilanfragen für die einzelnen Datenquellen umgesetzt werden. Es erfolgt im Gegensatz zur Verwendung eines domänenspezifischen globalen Schemas keine vorherige Schemaintegration, sondern die Integration erfolgt „on-the-fly“ durch die vom Nutzer gestellten Anfragen, d. h. insbesondere durch die Join-Bedingungen zwischen Attributen verschiedener Quellen.

Eine andere Form der virtuellen Integration lässt sich – analog zu GAM bei GenMapper – durch ein generisches Schema bzw. ein generisches Datenmodell realisieren. So ist z. B. TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources) [33, 66] ein Query-Mediator-Ansatz auf OEM-Basis (Object Exchange Model). Dabei werden die Daten der Quellen als Objekte mit eindeutiger ID interpretiert und deren Attribute durch Name, Wert und Datentyp beschrieben. Die Zuordnung der Objekt-IDs zu den Ursprungsdaten erfolgt durch Regeln. Erhalten zwei Objekte dabei die gleiche Objekt-ID, werden sie als gleich angesehen und fusioniert [146]. Mittels einer SQL-Erweiterung können dann Anfragen bezüglich des generischen Schemas ausgeführt werden. Ein ähnliches Vorgehen realisiert Kleisli [206], das ein

objektorientiertes Typsystem mit primitiven Datentypen und verschachtelten Objekten unterstützt.

Ein hybrider Ansatz bzgl. der Instanzdatenintegration wird z. B. in einer Erweiterung von GenMapper verfolgt [106, 102]. Dabei werden auf der einen Seite Korrespondenzen zwischen Objekten in verschiedenen Datenquellen materialisiert in einem Data Warehouse gespeichert. Auf der anderen Seite werden aktuelle Annotationsdaten der gespeicherten Objekte mit Hilfe der Datenquelle SRS bei Anfragen generiert. Dadurch werden die Vorteile der materialisierten Integration (performanter Zugriff auf bereits integrierte Daten) mit den Vorteilen der virtuellen Integration (stets aktuelle Daten mittels Anfragen an die Quellen) kombiniert.

6.2.3 Datenintegration ohne Verwendung eines globalen Schemas

Die Verwendung eines globalen Schemas führt stets zu einer zentralisierten Architektur, bei der alle Datenquellen mit einer zentralen Instanz (Warehouse, Mediator) verbunden werden. Als Alternative zu den bisher vorgestellten Ansätzen wurden Verfahren entwickelt, die gänzlich ohne globales Schema auskommen, was stets eine virtuelle Integration zur Folge hat. Einsatzschwerpunkt sind Peer-Data-Management-Systeme (PDMS), welche auf dezentralen Peer-to-Peer-Umgebungen (P2P) aufbauen, in denen autonome Datenquellen mit benachbarten Datenquellen verbunden sind [16].

Statt Schema-Mappings zwischen den Quellenschemata und einem globalen Schema verwendet Piazza [83, 84] (als Teil des Revere-Projekts [81]) Schema-Mappings zwischen den Quellen selbst. Dies erlaubt eine P2P-Verknüpfung der Datenquellen, so dass insbesondere eine neue Quelle nur ein Schema-Mapping zu einer bereits verknüpften Quelle benötigt. Ein Vorteil liegt dabei in der Möglichkeit, sich diejenige Quelle auszuwählen, für welche sich die Bildung des Schema-Mappings besonders einfach gestaltet. Anfragen werden durch den Nutzer bezüglich des Schemas einer Quelle gestellt. Durch eine Konvertierung der Anfrage mittels der Schema-Mappings können auch die benachbarten Quellen (sowie deren Nachbarn usw.) bei der Anfrageverarbeitung einbezogen werden. Einen ähnlichen Ansatz stellen Edutella [138], bei dem Peers durch RDF-Metadaten miteinander verknüpft werden, und System P [164] dar. Im Gegensatz dazu verwendet PeerDB [141, 144] keine Mappings zwischen den Quellenschemata, sondern propagiert lediglich Suchanfragen (z. B. Keyword-Suche) von einem Peer zu seinen benachbarten Peers. Jeder Peer ist in der Lage, die Suchanfragen auszuführen und liefert die Ergebnisse zurück. Ähnliche Ansätze gibt es im Bereich von P2P-Suchmaschinen (u. a. Minerva [12] und [152]).

Orchestra [94] erweitert die Ideen von Piazza in verschiedene Richtungen. Zunächst werden Hub-Schemata gebildet (shared relations), zu denen eine Anzahl von Peers Schema-Mappings besitzt und die selbst über Schema-Mappings miteinander ver-

bunden sein können. Dadurch entsteht eine Peer-Infrastruktur mit Superknoten, welche verschiedene Teile des P2P-Netzes separiert. Neben der metadatenbasierten Verknüpfung verwendet Orchestra auch instanzbasierte Mappings, um Vertrauen in die Daten der Peers auszudrücken (trusted data). Instanzen, die als vertrauensvoll angesehen werden, können z. B. bei der Query-Verarbeitung verwendet oder in der Peer-Datenquelle repliziert werden, um den eigenen Datenbestand zu vergrößern. Informationen zu korrespondierenden Instanzen werden in Hyperion [8, 163] ausgenutzt. Es basiert auf Mapping-Tabellen [100], die korrespondierende Attributwerte (z. B. künstliche Primärschlüssel) in den verschiedenen Datenquellen repräsentieren. Zusätzlich gibt es Mapping-Ausdrücke, die angeben, in welchen anderen Quellen sich relevante Instanzen zur Beantwortung von Anfragen befinden. Dadurch ist es möglich, Anfragen nicht nur auf Grund der verschiedenen Schemata umzuformulieren, sondern z. B. auch die zu überprüfenden Attributwerte bei SQL-WHERE-Bedingungen.

Eine Verallgemeinerung der P2P-artigen Verknüpfung von Datenquellen stellen sogenannte Data Spaces [60, 82] dar. Ein Data Space besteht dabei aus einer Reihe heterogener Datenquellen (z. B. Datenbanken, XML-Dateien, Web Services), die durch Relationen miteinander verknüpft sind. Relationen können dabei sowohl schema- als auch instanzbasierte Mappings sein, aber auch einfache Beziehungen (z. B. „Quelle A ist View von Quelle B“) zwischen Datenquellen beschreiben. Die Grundidee der Data Spaces ist die Abkehr von der klassischen Datenintegration mit einem einheitlichen Zugriff auf alle Quellen hin zu einem heterogenen Miteinander verschiedener Datenquellen, mit denen innerhalb des Data Spaces „gearbeitet“ werden kann. Die für den Nutzer bereitgestellten Dienste richten sich dabei nach den von den Quellen angebotenen Funktionalitäten, z. B. ob sie eine Schlüsselwortsuche unterstützen oder nicht.

6.3 Related Work: Object Matching

Object Matching ist Teil des Data Cleanings [156] und ein entscheidender Schritt bei der Datenintegration. Es beschreibt die Identifikation verschiedener Datensätze, welche Abbilder der gleichen Entität der realen Welt sind. Dabei können die betrachteten Datensätze sowohl in verschiedenen Datenquellen als auch innerhalb der gleichen Datenquelle auftreten. Object Matching tritt in der Literatur u. a. unter den folgenden verschiedenen Begriffen auf: Deduplication [43], Duplicate Detection [18], Duplicate Record Elimination [21], Entity Identification [117], Entity Reconciliation [45], Entity Resolution [143], Identity Resolution [97], Merge/Purge Problem [88], Object Consolidation [34], Object Identification [10], Record Linkage [59], Reference Matching [123] oder Reference Reconciliation [52]. Übersichtsartikel zum Thema Object Matching finden sich in [75, 57, 204].

Object Matching wurde 1959 erstmals definiert und 1969 wie folgt formalisiert [140,

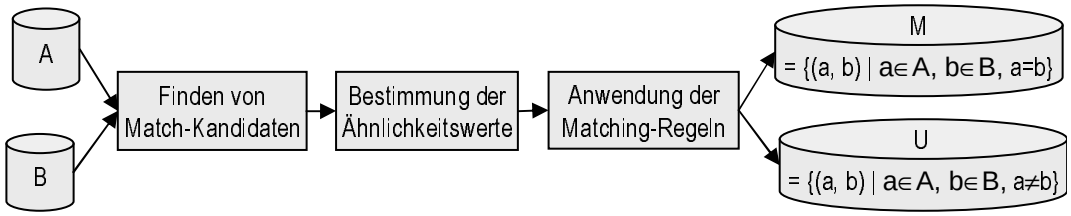


Abbildung 6.1: Allgemeines Modell des Object Matchings

59]: Gegeben ist das kartesische Produkt $P = A \times B$ zweier Objektmengen A und B. Durch Object Matching wird P in zwei disjunkte Teilmengen M und U partitioniert, wobei M (matched) alle gleichen Paare (a, b) und U (unmatched) alle nicht-gleichen Paare enthält.

Der Prozess des Object Matchings ist in Abbildung 6.1 illustriert und lässt sich wie folgt vereinfacht darstellen: Ausgehend von den beiden Datenquellen A und B werden zunächst *Match-Kandidaten* ermittelt, d. h. Paare von Datensätzen aus A und B, die potenziell als gleich angesehen werden könnten. Anschließend werden für die Match-Kandidaten *Ähnlichkeitswerte* zwischen den jeweiligen Datensätzen bestimmt. Dabei werden für zwei Datensätze i. Allg. mehrere unterschiedliche Werte durch verschiedene Verfahren sowie durch die Verwendung unterschiedlicher Attribute ermittelt. Im letzen Schritt werden mit Hilfe der Ähnlichkeitswerte sogenannte *Matching-Regeln* ausgeführt, die definieren, ob zwei Datensätze als gleich anzusehen sind oder nicht. Beim Object Matching innerhalb der gleichen Datenquelle (Duplikaterkennung) können die resultierenden Paare gleicher Datensätze optional noch zu Clustern zusammengefasst werden.

Im Folgenden wird zu jedem der drei Teilschritte des Object Matchings relevante Literatur näher vorgestellt. Abschließend erfolgt ein Überblick über Frameworks, die für das Object Matching eingesetzt werden können.

6.3.1 Finden von Match-Kandidaten

Die Identifikation von Match-Kandidaten hat das Ziel, die nachfolgende Bestimmung der Ähnlichkeitswerte zwischen Datensätzen auf ein Minimun zu reduzieren [11]. Der Fokus liegt auf einer Steigerung der Effizienz des Object Matchings ohne Reduzierung der Match-Qualität. Es sollen also möglichst viele Paare von Datensätzen, die sich als Nicht-Match herausstellen, bereits in diesem Schritt eliminiert werden, ohne dass korrekte Match-Paare fälschlicherweise entfernt werden. Auf der einen Seite wird daher eine hohe *Reduction Rate* (RR), d. h. um wie viel Prozent die Anzahl der Match-Kandidaten im Vergleich zum Worst-Case („jeder Datensatz mit jedem“) reduziert werden konnte, angestrebt. Auf der anderen Seite soll die *Pairs Completeness* (PC), d. h. welcher Anteil der korrekten Match-Ergebnisses nach der Reduktion (d. h. in den Match-Kandidaten) noch vorhanden ist, ebenfalls so hoch wie möglich

sein. Das optimale Ergebnis entsteht, wenn alle gefundenen Match-Kandidaten auch im Match-Ergebnis auftreten und gleichzeitig alle korrekten Match-Ergebnisse auch als Match-Kandidaten ermittelt werden ($PC = 1$).

Blocking-Verfahren erzeugen eine Partitionierung der Datensätze, d. h. die vollständige Zerlegung in disjunkte Bereiche (Blöcke), so dass nur noch zwischen Objekten desselben Blocks eine Ähnlichkeitsbestimmung durchgeführt werden muss [139]. Ein typischer Ansatz ist dabei die Generierung eines Blockschlüssels (block key) für jedes Objekt. Objekte mit gleichem Blockschlüssel gehören zum gleichen Block. Die Generierung des Schlüssels erfolgt mit relevanten Attributwerten jedes Datensatzes und soll möglichst einfach und schnell zu berechnen sein. Zum Beispiel kann für Publikationsdatensätze ein möglicher Blockschlüssel der Referenz-String einer Publikation sein, der sich aus Anfangsbuchstaben der Familiennamen der Autoren sowie dem Publikationsjahr zusammensetzt, z. B. [TR07] für die Publikation [192]. Dabei sollte ein Blockschlüssel möglichst unanfällig gegenüber Datenfehlern, z. B. fehlende Werte oder Tippfehler, sein, d. h. auch bei unsauberen Attributwerten soll für alle *gleichen* Datensätze *derselbe* Blockschlüssel erzeugt werden. Das Beispiel des Referenz-Strings für Publikationen ist z. B. unanfällig gegen Datenfehler im Publikationstitel oder bei Autoren, solange nicht der erste Buchstabe des Familiennamens falsch ist. Andererseits führt z. B. ein Vertauschen der Reihenfolge der Autoren zu einem anderen Schlüssel (z. B. [RT07] statt [TR07]).

Wichtigstes Kennzeichen des Blocking-Verfahrens ist, dass nur Datensätze mit dem gleichen Blockschlüssel als Match-Kandidaten ermittelt werden. Die Sorted-Neighborhood-Methode (SNM) [88] weicht diese Prämisse auf, indem auch Datensätze mit ähnlichen Schlüsseln verglichen werden. Dazu geht die SNM in drei Phasen vor. In der ersten Phase wird – analog zum Blocking-Verfahren – für jeden Datensatz ein Schlüssel generiert. Die Datensätze werden in der zweiten Phase (alphabetisch) nach ihrem Schlüssel sortiert, so dass durch eine geeignete Schlüsselwahl ähnliche Datensätze nach der Sortierung „nahe beieinander liegen“. Die dritte Phase verwendet ein Sliding Window, d. h. ein „Fenster“ fester Breite w ($2 \leq w \leq n$, wobei n die Anzahl der Datensätze bezeichnet) wird über die sortierte Liste „entlanggezogen“. Nur Objekte innerhalb des Fensters werden miteinander als Match-Kandidaten verglichen. Das bedeutet, dass ein Objekt mit den $w - 1$ vorherigen und $w - 1$ nachfolgenden Objekten verglichen wird. Als Erweiterung der SNM wurde der Mutli-Pass-Ansatz [89] vorgestellt, der eine mehrfache, unabhängige Schlüsselgenerierung unter Verwendung unterschiedlicher Attribute unterstützt. Durch die mehrfache Ausführung der SNM wird der Einfluss der Schlüsselwahl reduziert, so dass die Wahrscheinlichkeit steigt, dass auch bei fehlerhaften Daten gleiche Datensätze zumindest bei einem Schlüssel nach der Sortierung „nahe“ beieinander liegen. Gleichzeitig wurde experimentell gezeigt, dass für gute Ergebnisse beim Multi-Pass-Ansatz die Fenstergröße w pro Durchlauf der SNM deutlich kleiner gewählt werden kann als bei der Standard-SNM.

Sowohl Blocking-Verfahren als auch SNM hängen wesentlich von der Schlüsselgene-

rierung ab. Meist wird die Bestimmung des Schlüssels von einem Domänenexperten durchgeführt, der die relevanten Attribute bzw. deren Teile (z. B. Anfangsbuchstaben der Autoren bei Publikationen) erkennt und dabei typische Datenfehler in der Domäne beachtet. Mit automatischen Lernverfahren können Schlüssel auch ohne manuellen Aufwand erzeugt werden [127]. Dabei werden sowohl die verwendeten Attribute identifiziert, als auch die verwendete Methode zur Definition des zu vergleichenden Wertes, z. B. vollständiger Attributwert oder Übereinstimmung des ersten Zeichens, ermittelt. In [127] wird dazu ein inkrementelles Verfahren vorgestellt, das mittels Trainings- und Testdaten sukzessive Regeln zur Definition des Schlüssels lernt.

Ohne Schlüsselgenerierung kommen Cluster-Ansätze aus, die mit Hilfe „billiger“, d. h. schnell und einfach zu berechnender, Abstandsfunktionen überlappende Cluster, sogenannte Canopies [123], bilden. Alle Datensätze eines Clusters werden anschließend jeweils als Match-Kandidaten angesehen. Für die Abstandsfunktion werden – analog zur Schlüsselgenerierung – i. Allg. nur ausgewählte Attribute verwendet, so dass eine approximierte Ähnlichkeit berechnet wird. Bei Verwendung eines Standard-Greedy-Algorithmus für das Clustering müssen auch alle Objekte paarweise miteinander verglichen werden, so dass der Effizienzgewinn – im Vergleich zum Object Matching ohne vorherige Identifikation der Match-Kandidaten – durch die schnelle Berechnung der Ähnlichkeit entsteht. Zusätzlich lässt sich die Anzahl der Vergleiche beim Clustering weiter minimieren, indem Cluster inkrementell gebildet werden und neue Datensätze zunächst nur mit einem Repräsentanten des Clusters verglichen werden [133]. Nur in den Fällen, in denen die Ähnlichkeit nicht hinreichend groß (Hinzunahme des neuen Datensatzes zum Cluster) und nicht hinreichend klein (keine Hinzunahme zum Cluster) genug ist, werden weitere Vergleiche mit anderen Datensätzen des Clusters hinzugezogen.

6.3.2 Bestimmung der Ähnlichkeitswerte

Zur Bestimmung der Ähnlichkeitswerte können sowohl Attribute als auch Kontextinformationen, d. h. z. B. assoziierte Datensätze, verwendet werden.

Attributbasierte Ähnlichkeitswerte

Attributbasierte Verfahren verwenden ausschließlich die zur Verfügung stehenden Attributwerte der Datensätze und schließen von der Ähnlichkeit der Attributwerte auf die Ähnlichkeit der Datensätze. Dieser Ansatz wird auch als Field Matching bezeichnet [132]. Die zu Grunde liegende Annahme ist, dass gleiche (oder ähnliche) Datensätze gleiche (oder ähnliche) Attributwerte aufweisen. Für die Bestimmung der Ähnlichkeitswerte von Attributwerten wird zwischen domänenspezifischen und generischen Ähnlichkeitsfunktionen unterschieden.

Domänenspezifische Ähnlichkeitsfunktionen berechnen die Ähnlichkeit von Attributwerten mit Hilfe von speziell auf die Domäne abgestimmten Maßen. Für die Ähnlichkeit geografischer Orte ist das z. B. eine Funktion, die aus den Längen- und Breitengraden eine geografische Entfernung zwischen Orten bestimmt. Im Bereich der Ahnenforschung [153], bei der gleiche Personen in verschiedenen Quellen (z. B. Tauf-, Hochzeits- und Sterberegister der Kirchen) identifiziert werden sollen, werden spezielle Besonderheiten der Namensgebung beachtet. Das beinhaltet z. B. entsprechende Synonymtabellen für häufig gebrauchte Abkürzungen bzw. Variationen der Vornamen (z. B. Peg, Peggy, Margaret[37]) oder die Tatsache, dass Frauen (meistens) nach der Hochzeit den Familiennamen ihres Mannes annehmen, was insbesondere beim Abgleich von Tauf- und Sterberegister beachtet werden muss.

Im Bereich generischer Ähnlichkeitsfunktionen wurden hauptsächlich verschiedene Stringähnlichkeitsmaße vorgestellt (siehe u. a. [73] oder das SimMetrics-Projekt[38] für eine Bibliothek mit implementierten Algorithmen), die in [136, 39] miteinander verglichen werden. Es wird zwischen Edit- und Token-basierten Ähnlichkeitsmaßen unterschieden, um die Ähnlichkeit zweier Strings s und t zu bestimmen. Edit-basierte Maße vergleichen die Strings zeichenweise. Ein bekanntes Maß ist die Edit- bzw. Levenshtein-Distanz [114], welche die minimale Anzahl von Änderungsoperationen (Einfügen, Löschen, Ersetzen) bestimmt, um den String s in den String t zu überführen. Weitere Beispiele für Edit-basierte Ähnlichkeitsmaße sind die Jaro-Winkler-Ähnlichkeit [203], die insbesondere für (kurze) Personennamen gut geeignet ist, sowie der SOUNDEX-Algorithmus [202], welcher die phonetische Aussprache vergleicht. Token-basierte Ähnlichkeitsmaße zerlegen die Strings in einzelne Tokens. Tokens können dabei einzelne Worte (Zerlegung mittels Trennzeichen, z. B. Leerzeichen oder Komma) oder sogenannte n-Gramme, d. h. Teilstrings einer festen Länge n, sein. Ein Beispiel eines Token-basierten Ähnlichkeitsmaßes ist die Jaccard-Ähnlichkeit, welche den Anteil der übereinstimmenden Tokens an der Menge aller Tokens ermitellt. Das TF·IDF-Maß [166] gewichtet zusätzlich alle Tokens. Je häufiger ein Token in den betrachteten String vorkommt und je seltener das Token in der Gesamtheit aller bekannten Strings vorkommt, desto höher ist sein Gewicht. Die Qualität des Ähnlichkeitsmaßes hängt von verschiedenen Faktoren ab, u. a. von den konkreten Attributwerten. Ein weiterer Nachteil insbesondere komplexerer Ähnlichkeitsmaße ist deren Parametrisierung. Daher wurden in [18, 19, 129] Lernverfahren für Stringähnlichkeitsmaße vorgestellt, welche für jedes Attribut ein optimiertes Ähnlichkeitsmaß ermitteln. Als Lernverfahren kommen z. B. Expectation-Maximization (EM) und Support-Vector-Machine (SVM) zum Einsatz.

[37] http://de.wikipedia.org/wiki/Margarete
[38] http://www.dcs.shef.ac.uk/~sam/simmetrics.html

Kontextbasierte Ähnlichkeitswerte

Kontextbasierte Verfahren verwenden zur Bestimmung der Ähnlichkeitswerte auch Daten, die mit den zu vergleichenden Datensätzen in Beziehung stehen. Eine Möglichkeit ist z. B. das Anreichern der Datensätze um assoziierte Informationen. Dabei werden in [125, 126] Attribute assoziierter Objekte in anderen Datenquellen (Secondary Source) zu den ursprünglichen Datensätzen (aus der Primary Source) hinzugefügt. Ein Beispiel ist das Nachschlagen und Hinzufügen einer Telefonvorwahl anhand der Adresse, was das Vergleichen von Telefonnummern verbessert. Anschließend können die bisherigen attributbasierten Verfahren zur Bestimmung von Ähnlichkeitswerten angewendet werden.

Ein ähnlicher Ansatz wird in [110] am Beispiel der bibliografischen Datenquelle DBLP vorgestellt. Kontextattribute, d. h. Attribute korrespondierender Daten, werden hinzugezogen und mittels Assoziationsregeln die für die Objektkonsolidierung relevanten Attribute bestimmt. Des Weiteren werden auch hierarchische Daten verwendet, z. B. eine thematische Kategorisierung der Konferenzen bzw. Journals.

Ein wichtiger Spezialfall von Kontexten sind Hierarchien, die z. B. in Data-Warehouse-Dimensionen oder XML-Daten entstehen. Für Data Warehouses wurde in [7] ein Ansatz vorgestellt, der hierarchische Dimensionen zur Duplikaterkennung verwendet. Als Beispiel dient eine geografische Dimension mit den Ebenen Stadt, Region und Staat. Obwohl die Staaten USA und Großbritannien auf Grund ihrer Schreibweise im Englischen (US bzw. UK) als Duplikate in Betracht kommen, kann durch Hinzunahme der Daten in den weiteren Ebenen der Dimensionen eine Gleichheit ausgeschlossen werden, da für die korrespondierenden Regionen und Städte keine entsprechenden Duplikate im jeweils anderen Land gefunden werden.

Auf XML-Daten ist der in [198] vorgestellte Ansatz ausgerichtet, welcher XML-Elemente nicht nur auf Grund ihres Inhalts, sondern auch auf Grund der Ähnlichkeit ihrer Vater- und Kindknoten vergleicht. Dazu werden Heuristiken entwickelt, welche u. a. die direkt benachbarten Knoten oder alle Knoten innerhalb eines Maximalabstandes einbeziehen. Ein ähnlicher Ansatz findet sich in [29] und die Idee wird z. B. auch für das Schema Matching angewendet (siehe Übersichtsartikel [154]). In [175] werden mittels Constraints auch Beziehungen zu anderen Datensätzen ausgedrückt. So kann z. B. für Autoren die Liste ihrer Koautoren genutzt werden, um gleiche Personen zu erkennen.

Mittels eines graphbasierten Ansatzes werden in [52] Datensätze miteinander in Beziehung gesetzt, z. B. Publikationen und ihre Autoren. Anschließend wird ein iterativer Algorithmus angewendet, welcher die Ähnlichkeit zweier Knoten (d. h. Datensätze) auch auf Grund der Ähnlichkeit der über Kanten verbundenen Knoten (d. h. der assoziierten Datensätze) bestimmt. Dadurch wird Wissen über als Duplikate erkannte Datensätze propagiert (Reference Propagation) und durch die Zusammenfassung der Attribute aller Duplikate werden die Datensätze um Informationen angereichert

(Reference Enrichment). Ähnliche Ansätze finden sich in [17, 34, 98] sowie speziell für XML in [199].

Methoden des maschinellen Lernens werden in [147, 177, 43] angewandt. Die Modellierung der Kontextinformationen erfolgt durch Conditional Random Fields bzw. Relation Probability Models. Ausgehend von diesen Modellen werden Wahrscheinlichkeiten bzgl. der Gleichheit zwischen Datensätzen abgeleitet, welche dann als Ähnlichkeitswerte aufgefasst werden.

6.3.3 Definition der Matching-Regeln

Manuell definierte Matching-Regeln

Die einfachste Form manuell definierter Regeln sind Schwellwerte für die erhaltenen Ähnlichkeitswerte eines Attributs, welche Duplikate und Nicht-Duplikate trennen. Der Schwellwert kann bei einer Evaluation auch als Parameter für die Genauigkeit der Objektkonsolidierung dienen (siehe z. B. [110]).

Mit der Verwendung mehrerer Attribute innerhalb einer Matching-Regel können komplexere Zusammenhänge zwischen zwei Datensätzen dargestellt werden. Diese beinhalten in der Regel Expertenwissen über die zu Grunde liegende Domäne [122]. In [88] können z. B. Regeln der Art „Wenn die Familiennamen gleich sind und die Vornamen hinreichend ähnlich“ formuliert werden. Das Data-Cleaning-Framework AJAX [64] bietet dazu einen Matching-Operator, mit welchem eine Relation definiert wird, die die korrespondierenden Duplikate enthält. Dies wird mit Hilfe einer SQL-Erweiterung ähnlich einer SQL-View-Definition realisiert, so dass auch auf Daten in anderen Relationen zugegriffen werden kann.

Eine besondere Art von Matching-Regeln sind Constraints, die erfüllt sein müssen. In [49, 175] wird dazu ein Ansatz vorgestellt, der auch Attribute verwendet, die nicht in die Ähnlichkeitsberechnung eingegangen sind. Werden z. B. zwei Personen als gleich erkannt, so wird geprüft, ob das Alter der Person (gepeichert in der einen Datenquelle) mit dem Einkommen (aus der anderen Datenquelle) im Einklang steht. Entsteht durch die Objektkonsolidierung z. B. ein neunjähriger Junge mit einem Jahreseinkommen von 200.000 US-Dollar, ist dies ein Indiz für zwei verschiedene Personen. Dadurch ist es möglich, trotz Verwendung generischer Ähnlichkeitsmaße (z. B. für den Personennamen), domänenspezifisches Wissen durch die Matching-Regeln einzubringen.

Adpative Matching-Regeln

Das Active Atlas System [186] lernt anhand von Trainingsdaten Matching-Regeln der Form „Wenn Attribut X größer Schwellwert Y, dann Match“. Diese Regeln entstehen aus der Verwendung mehrerer Entscheidungsbaumverfahren (Bagging). Dadurch

werden sowohl die relevanten Attribute (bzw. die daraus ermittelten Ähnlichkeitswerte) als auch die zugehörigen Schwellwerte automatisch bestimmt. In einer Erweiterung [187] werden zusätzlich Gewichte für Attributtransformationen gelernt, d. h. ob und wie Attributwerte im Vorfeld transformiert werden müssen. Solche Transformationen umfassen z. B. die Auflösung von Abkürzungen oder die Bildung von grammatikalischen Stammformen (Stemming).

Ein interaktiver Ansatz zur Objektkonsolidierung wird in [167] verfolgt. Mittels Trainingsdaten wird eine optimale Kombination verschiedener Ähnlichkeitswerte bestimmt. Im Rahmen dieses adaptiven Prozesses kann der Nutzer Feedback durch manuelle Klassifikation weiterer Trainingsdaten geben. Dabei wählt das System selbst die zu klassifizierenden Daten aus, um einen möglichst großen Informationsgewinn durch die Nutzerinteraktion zu erzielen (Active Learning).

6.3.4 Frameworks

Neben den vorwiegend „algorithmischen“ Arbeiten zu einzelnen Teilaspekten des Object Matchings spielen Frameworks, die den gesamten Object-Matching-Prozess unterstützen, eine wichtige Rolle.

Auf Basis dieses allgemeinen Modells wurde in [137] ein generisches Object Identification Framework entwickelt, mit welchem verschiedene Verfahren zum Object Matching verglichen werden können.[39] Andere Data-Cleaning-Frameworks (TAILOR [56], Febrl [38], AJAX [63, 64]) unterstützen das Object Matching durch Bereitstellen mehrerer Verfahren für Ähnlichkeitswerte und Matching-Regeln. Insbesondere die Bestimmung der Ähnlichkeitswerte kann durch eine entsprechende Datentransformation im Vorfeld verbessert werden. Einen interaktiven Ansatz bietet dabei Potter's Wheel [159], mit dessen Hilfe der Nutzer Transformationsvorschriften definieren und überprüfen kann.

Auch in kommerziellen Datenbanksystemen, z. B. SQL-Server 2005 [31], finden sich zunehmend Funktionen zur Unterstützung des Object-Matching-Prozesses. So können z. B. spezielle Operatoren innerhalb eines Datentransformations-Workflows verwendet werden. Ein Beispiel für den SQL Server 2005 ist der Fuzzy-Join-Operator [32], der für Datensätze syntaktisch ähnliche Datensätze in einer Referenztabelle identifiziert.

6.4 Problemstellung

Zur Verdeutlichung offener Probleme werden für die bereits in Abschnitt 6.1.1 genannten Bereiche (Bioinformatik und bibliografische Daten) jeweils kurze Szenarien

[39] Es liegen jedoch derzeit (Stand: Oktober 2007) keine publizierten Evaluationsergebnisse vor.

skizziert, welche insbesondere die Anforderungen der Nutzer in Bezug auf die Datenintegration hevorheben.

Wie bereits in Abschnitt 6.1.1 beschrieben, entstanden durch die intensive Forschung mittlerweile hunderte von Webdatenquellen zu molekularen Objekten, z. B. Genen, Proteinen oder Stoffwechselwegen, die untereinander durch Referenzen stark vernetzt sind [65]. Die Erstellung eines globalen Schemas für alle oder für einen signifikanten Teil der Quellen ist auf Grund der Heterogenität der Quellen sowie deren Komplexität und dem hohen Tempo, mit dem die Quellen weiterentwickelt und somit verändert werden, fast unmöglich. Auf der anderen Seite bieten die vielen Querverweise die Möglichkeit, zu einem Objekt (z. B. einem Gen) weitere Informationen innerhalb anderer Quellen zu erhalten. Diese Querverweise sind üblicherweise von hoher Qualität, da sie von entsprechenden Experten manuell erstellt und gepflegt werden. Während das manuelle Traversieren der Links für ein oder wenige Objekte noch relativ einfach möglich, erfordert es für viele Objekte, z. B. bei einer Genexpressionsanalyse [46], einen sehr hohen Aufwand. Gleichzeitig muss der Nutzer bei der manuellen Traversierung der Links deren Semantik kennen, um die Bedeutung der Assoziationen zu verstehen und zu interpretieren. Bei der Durchführung verschiedener Analysen kommt es zusätzlich zu einer wiederholten Ausführung einzelner Analyseschritte. Es wird daher ein Integrationsansatz benötigt, in dem Workflows erstellt und ausgeführt werden können, die auf den verfügbaren Querverweisen in Form von Mappings aufsetzen und entsprechende Analysen durch Datenfusion verschiedener Quellen ermöglichen.

Ein weiterer Aspekt der dynamischen Informationsfusion ist die Kopplung lokaler Datenquellen mit Webdatenquellen. Ein mögliches Szenario aus dem Bereich bibliografischer Daten wird in Abbildung 6.2 illustriert. Die drei Webdatenquellen DBLP Bibliography [2], ACM Digital Library [1] und Google Scholar [3] stellen jeweils Informationen zu wissenschaftlichen Publikationen bereit, wobei die letzten beiden neben den bibliografischen Angaben auch die Zahl der Zitierungen für Publikationen erfassen. Nutzer 1 möchte für die in seinem Adressbuch gespeicherten Kollegen erfahren, welche neuen Publikationen sie in letzter Zeit hervorgebracht haben. Dazu muss er für jeden Namen den passenden Autoreintrag bei DBLP finden und aus der Liste der Publikationen die neuesten extrahieren. Neben diesem zeitaufwändigen Vorgehen sollten abschließend noch doppelte Einträge entfernt werden, wenn zwei oder mehr Autoren an der gleichen Publikation beteiligt waren. Nutzer 2 hat für die auf seinem Rechner gespeicherten PDF-Files eine Excel-Tabelle angelegt und dort für jede Publikation – jeweils identifiziert durch die ID bei ACM, wo er die PDF-Volltexte heruntergeladen hat – vermerkt, an welcher Institution sie entstanden ist. Er möchte nun für jede Institution wissen, wie viele Zitierungen der zugehörigen Publikationen im Durchschnitt bei Google Scholar verzeichnet sind. Dazu muss er für jede Institution anhand der IDs der Publikationen bei ACM Titel, Autoren und Publikationsjahr extrahieren, mit denen er anschließend bei Google Scholar sucht. Dabei muss er die Zitierungszahlen für eine Publikation i. Allg. aufsummieren, da

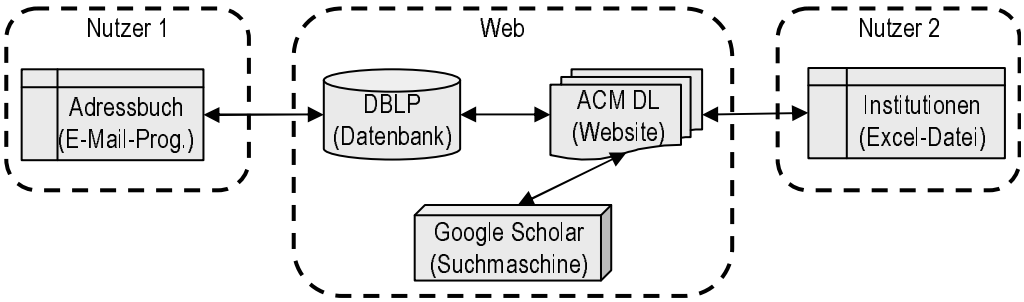


Abbildung 6.2: Szenario zur Kopplung lokaler und Webdatenquellen innerhalb der bibliografischen Domäne

Google Scholar meist mehrere Einträge zu einer Publikation liefert.

Eine materialisierte Integration der Daten verbietet sich aus verschiedenen Gründen. Zunächst wollen Nutzer i. Allg. nicht ihre persönliche Daten in einem zentralen Repository, z. B. in einem Data Warehouse, ablegen. Des Weiteren können Suchmaschinen und große Websites nicht – oder nur mit sehr großem Aufwand – vollständig heruntergeladen werden. Zusätzlich ist auch – wie in der Bio-Informatik – eine Erstellung eines globalen Schemas auf Grund der Vielzahl der lokalen Datenquellen aussichtslos. Das beschriebene Szenario verlangt daher nach einer flexiblen Möglichkeit, Datenquellen durch instanzbasierte Mappings (z. B. bei Nutzer 2 mittels gleicher ID) zu koppeln, um anschließend nutzerspezifische Workflows ausführen zu können. Dabei muss auch das Identifizieren gleicher Objekte in verschiedenen Datenquellen (z. B. Suche nach Autoren anhand des Namens bei Nutzer 1) unterstützt werden, was zu einer Erstellung neuer Mappings führt.

Die beschriebenen Szenarien führen u. a. zu folgenden drei Problemen, die innerhalb der vorliegenden Arbeit angegangen werden:

- Wie können Datenquellen ohne Verwendung von Schemata einfach, flexibel und schnell miteinander verknüpft werden?
- Wie können in einer P2P-artigen Umgebung verteilter Datenquellen zielgerichtet Prozesse zur Informationsfusion definiert werden?
- Wie kann unter Verwendung instanzbasierter Korrespondenzen Object Matching realisiert werden?

Architektur zur dynamischen Datenfusion: Die Hauptanforderung einer flexiblen Datenintegrationsarchitektur muss in der schnellen Verknüpfung von Datenquellen liegen, so dass insbesondere neue Datenquellen *einfach* hinzugefügt werden können. Es sollen dabei keine großen Anforderungen an die Datenquellen gestellt werden, insbesondere im Hinblick auf ihre Heterogenität. So sollen u. a. Dateien, Suchmaschinen und Datenbanken in uniformer Art

und Weise miteinander verknüpft werden können. Bei der Verknüpfung der Datenquellen soll auf *instanzbasierte Mappings* zurückgegriffen werden, die (wie z. B. in der beschriebenen Biodomäne) zum Teil in hoher Qualität vorliegen. Zusätzlich müssen aber auch neue Mappings dynamisch generiert werden können, z. B. unter Verwendung von Suchanfragen. Analog zu den Datenquellen soll eine breite Vielfalt von Realisierungsformen für Mappings unterstützt werden. Neben einer materialisierten Tabellenrepräsentation (Menge von Korrespondenzen) müssen auch ausführbare Mappings unterstützt werden, welche die Korrespondenzen erst zur Ausführungszeit bestimmen. So können z. B. für Mappings, deren Berechnung eine Suchmaschine involviert, nicht alle mögliche Korrespondenzen vorberechnet werden, sondern eben nur zur Anfragezeit für eine bestimmte Menge von Objekten. Die geschilderte Bandbreite der Datenquellen und Mappings erfordert daher eine *uniforme Repräsentation* der Datenquellen sowie ihrer Verknüpfungen. Dabei muss zur Unterstützung einer nutzergesteuerten Datenintegration (siehe folgenden Punkt) die Semantik der Datenquellen und Mappings erfasst werden.

Definition von Datenintegrationsaufgaben: Typischerweise eröffnen Datenintegrationsansätze eine Anfragemöglichkeit bzgl. eines Schemas (globales Schema oder Schema einer Datenquelle). Dabei wird die Anfrage mit Hilfe von Schema-Mappings in Anfragen für andere Quellen transformiert, ausgeführt und die resultierenden Daten zurücktransformiert, vereinigt und an den Nutzer gesendet. Dieses Verfahren ist bei einer schemalosen Kopplung von Datenquellen nicht möglich. Andererseits sind einfache navigierende Ansätze, die z. B. das Traversieren von Mapping-Korrespondenzen durch Hyperlinks ermöglichen, ungeeignet für datengetriebene Auswertungen für eine Menge von Objekten. Es muss daher eine *kompakte* Beschreibungsmöglichkeit für Nutzer geschaffen werden, die weitreichende Möglichkeiten zur Definition von Datenintegrationsaufgaben eröffnet. Dabei sollen Nutzer sowohl gezielt mit *spezifischen* Quellen und Mappings und ihren Besonderheiten als auch auf einer *abstrakteren* Ebene arbeiten können, die lediglich Wissen über die Domäne ohne Kenntnis der Quellen voraussetzt. Nutzer sollen so ohne großen Nutzeraufwand komplexe Datentransformationsprozesse definieren und ausführen können, die dabei vielseitig in verschiedenen Domänen, d. h. insbesondere für verschiedene Quellen, einsetzbar sein sollen. Daher muss eine *generische Verarbeitung* der relevanten Daten als Grundlage dienen.

Object Matching: Object Matching bezeichnet den Prozess der Bestimmung gleicher Entitäten in (meist verschiedenen) Datenquellen. Ein statisches, aufwändiges Verfahren zum Object Matching ist für die dynamische Umgebung ungeeignet, da z. B. auch für neue Datenquellen ad-hoc ein Matching möglich sein muss. Zusätzlich muss der Match-Prozess auch *flexibel* gegenüber den Anforderungen der das Match-Ergebnis nutzenden Applikation gestaltet werden können. Ein nicht zeitkritischer Prozess, der höchsten Wert auf Datenqualität legt,

benötigt ein anderes Match-Verfahren als eine Online-Anwendung, bei der das Ergebnis sehr schnell vorliegen muss und gegebenenfalls kleinere Datenfehler in Kauf genommen werden können. Für das Object Matching sollen *attributbasierte* Verfahren zum Einsatz kommen. Weiterhin finden sich relevante (zu vergleichende) Informationen zu den Objekten nicht nur in den jeweiligen Attributen, sondern auch in Korrespondenzen zu *assoziierten Objekten*, welche daher in den Match-Prozess eingehen müssen. Des Weiteren kann sich der Vergleich von Objektinstanzen, insbesondere ihrer Attributwerte, zwischen heterogenen Datenquellen schwierig gestalten. Daher ist es wichtig, in den Mapping-Korrespondenzen vorliegende Informationen auszunutzen, um z.B. durch eine entsprechende *Mapping-Kombination* bereits ermittelte Instanz-Mappings mit Korrespondenzen zwischen gleichen Objekte wiederzuverwenden. Schließlich muss die Heterogenität der Datenquellen auch beim Matching berücksichtigt werden, da z.B. für Datenquellen, die nur über Suchanfragen zugänglich sind, zunächst vor dem Match-Prozess mögliche „Match-Partner“ erst mittels Suchanfragen bestimmt werden müssen.

6.5 Zusammenfassung

In diesem Kapitel wurde eine Einführung in das Thema Datenintegration gegeben. Dabei wurde der Schwerpunkt auf Datenintegrationsansätze sowie auf den wichtigen Aspekt Object Matching gelegt. Ausgehend von der vorgestellten Literatur wurden offene Probleme für eine dynamische und flexible Datenintegration identifiziert, die Ausgangspunkt für die weiteren Kapitel sind.

7 Der iFuice-Ansatz

Im Mittelpunkt dieses Kapitels steht die Präsentation des iFuice-Ansatzes (Information Fusion utilizing Instance Correspondendes and Peer Mappings). Dazu werden zunächst die wesentlichen Ideen skizziert und der iFuice-Ansatz mittels verschiedener Kriterien von bisherigen Integrationsansätzen abgegrenzt (Abschnitt 7.1). Anschließend erfolgt die Darstellung der grundlegenden Datenstrukturen, d. h. wie Datenquellen repräsentiert werden und untereinander durch Mappings verknüpft werden (Abschnitt 7.2). In Abschnitt 7.3 wird das Workflow-basierte Vorgehen zur Datenintegration erläutert, ehe in Abschnitt 7.4 eine Übersicht aller Operatoren gegeben wird. Das Kapitel schließt mit der Vorstellung der Mediator-basierten iFuice-Architektur ab (Abschnitt 7.5).

7.1 Idee

Mit iFuice wird ein neuartiger Datenintegrationsansatz vorgestellt. Er basiert auf der Idee einer Peer-to-Peer-artigen Verknüpfung von Datenquellen durch *Instanz-Mappings*, d. h. durch Korrespondenzen zwischen Objektinstanzen. Solche Korrespondenzen sind u. a. in der Form von Hyperlinks bereits im Web verfügbar. Datenquellen und Mappings werden durch ein *Domänen-* sowie ein *Source-Mapping-Modell* beschrieben, die eine einfache semantische Repräsentation der betrachteten Domänen ermöglichen. iFuice ermöglicht eine *Workflow-basierte Nutzung*, welche auf einer Reihe von Operatoren und generischen Datenstrukturen aufsetzt. Dadurch können Nutzer mittels einer skriptbasierten Workflow-Definition Informationen aus

verschiedenen Datenquellen kombinieren und integrieren.

Die im Text kursiv hervorgehobenen wesentlichen Aspekte von iFuice werden im Folgenden näher vorgestellt.

Instanzbasierte Mappings: Die grundlegende Datenstruktur von iFuice sind instanzbasierte Mappings, die als Menge von Korrespondenzen zwischen je zwei Objektinstanzen repräsentiert werden. Jede Objektinstanz muss lediglich eine innerhalb der Datenquelle eindeutige ID besitzen und kann zusätzlich eine beliebige Menge weiterer Attribute aufweisen. Mappings repräsentieren damit Beziehungen zwischen Objekten und lassen sich dadurch als flexible Datenstruktur einsetzen und verarbeiten. Durch Mappings lassen sich sowohl semantische Beziehungen innerhalb einer Datenquelle ausdrücken, z. B. dass ein Autor A eine Publikation P geschrieben hat, als auch gleiche Objekte in verschiedenen Datenquellen referenzieren, z. B. die gleiche (reale) Publikation in zwei Datenquellen. Die einfache Struktur von Mappings erlaubt zusätzlich vielfältige Verarbeitungsmöglichkeiten. So kann z. B. die Vereinigung zweier Mappings gebildet werden oder durch Komposition („Hintereinanderschaltung“) zweier Mappings ein neues Mapping abgeleitet werden.

Domänen- und Source-Mapping-Modell: Das Domänenmodell ermöglicht eine einfache Modellierung einer Domäne. Es beschränkt sich auf die Definition von relevanten Objekttypen sowie auf eine semantische Beschreibung der Beziehungen zwischen Objekten durch Mapping-Typen. Dadurch ist es auf der einen Seite einfach zu realisieren, insbesondere im Vergleich zur Erstellung eines globalen Schemas. Auf der anderen Seite ermöglicht es aber dennoch eine auf eine Domäne fokussierte semantische Unterstützung der Datenintegration. Das Source-Mapping-Modell baut auf dem Domänenmodell auf und ordnet die zur Verfügung stehenden Datenquellen und Mappings den im Domänenmodell definierten Objekt- bzw. Mapping-Typen zu.

Nutzerdefinierte Workflows: iFuice ermöglicht Nutzern die Definition von Workflows zur Lösung einer konkreten Datenintegrationsaufgabe. Workflows werden durch Skripte beschrieben, die wiederum innerhalb einer prozeduralen Programmierung Operatoraufrufe enthalten. Die von iFuice unterstützten Operatoren arbeiten auf generischen Datenstrukturen und sind dadurch unabhängig von den konkreten Datenquellen oder der Domäne. Die flexible, skriptbasierte Definition von Workflows ermöglicht weiterhin die Erstellung und Ausführung unterschiedlicher Arten von Workflows. Nutzer können z. B. bestehende Mappings zur Datenintegration verwenden und alle verfügbaren Informationen zu gleichen Objekten fusionieren. Dabei erlauben die Operatoren sowohl ein quellenspezifisches Arbeiten, d. h. mit genauer Angabe, welche Mappings und Datenquellen zu verwenden sind, als auch eine transparente Ausführung, bei der z. B. nur die Zieldatenquelle definiert wird und iFuice die benötigten Mappings

selbst auswählt. Zusätzlich können durch Workflows auch neue Mappings bestimmt werden, die dann wiederum innerhalb anderer Workflows verwendet werden.

7.1.1 Vergleich mit anderen Datenintegrationsansätzen

Der Datenintegrationsansatz iFuice wird im Folgenden mit Data-Warehouse-, Query-Mediator- und schemabasierten Peer-Data-Management-Systemen(PDMS) anhand von sieben verschiedenen Kriterien verglichen. Dabei werden insbesondere die Unterschiede und Alleinstellungsmerkmale von iFuice gegenüber bestehenden Integrationsansätzen charakterisiert und die sich daraus ergebenen Vorteile beschrieben. Eine tabellarische Kurzübersicht des Vergleichs findet sich in Tabelle 7.1.

Globale Modellierung der Domäne: Zur Integration von Daten verwenden Data-Warehouse- sowie Query-Mediator-Ansätze ein globales Schema. Es repräsentiert eine einheitliche Modellierung der betrachteten Domäne und beschreibt die relevanten Entitäten inklusive ihrer Attribute sowie der Beziehungen zwischen den Entitäten. PDMS-Ansätze kommen ohne eine globales Modell aus und verwenden ausschließlich die Schemata der einzelnen Quellen.

Der iFuice-Ansatz verwendet ein Domänenmodell zur Beschreibung der betrachteten Domäne. Ähnlich einem globalen Schema enthält es Objekttypen sowie Beziehungen zwischen ihnen, die durch semantische Mapping-Typen charakterisiert werden. Das Domänenmodell erfordert jedoch im Unterschied zu einem globalen Schema keine Definition der Attribute für jeden Objekttyp. Es handelt sich daher im Vergleich zum globalen Schema um eine abstraktere Modellierung, welche dadurch einfacher zu realisieren ist. Gleichzeitig unterstützt das Domänenmodell die einfache Erweiterung um neue Datenquellen, da sich neue Quellen entweder dem bestehenden Domänenmodell zuordnen lassen oder das Domänenmodell um neue Objekttypen erweitert wird.

Verknüpfung der Datenquellen: Alle betrachteten Integrationsansätze außer iFuice sind schemabasiert, d. h. die Verbindung zwischen den Datenquellen erfolgt über Schema-Mappings. Data-Warehouse- und Query-Mediator-Ansätze verwenden für jede Datenquelle je ein Schema-Mapping zwischen dem globalen Schema und dem Schema der Datenquelle. Bei PDMS-Ansätzen erfolgt die Verknüpfung der Datenquellen durch Schema-Mappings zwischen den Datenquellen selbst. Für jede Peer-Datenquelle muss es mindestens ein Schema-Mapping zu einer anderen Peer-Datenquelle geben. Eine Quelle kann zusätzlich beliebig viele weitere Mappings zu anderen Quellen besitzen.

Im Unterschied zu den genannten Integrationsansätzen basiert iFuice auf instanzbasierten Mappings. Zwei Datenquellen werden durch Korrespondenzen zwischen je einer Instanz der beiden Datenquellen miteinander verbunden. Die

Merkmal	Data Warehouse	Query-Mediator	P2PDM	iFuice
Globale Modellierung der Domäne	ja, globales Schema	ja, globales Schema	nein	ja, Domänenmodell
Verknüpfung der Datenquellen	schemabasiert, Mappings zwischen globalem Schema und Quellenschema	schemabasiert, Mappings zwischen globalem Schema und Quellenschema	schemabasiert, Mappings zwischen Quellenschemata	instanzbasiert, Mappings zwischen Instanzen der Quellen
Art der Nutzerinteraktion	Query bzgl. globalem Schema	Query bzgl. globalem Schema	Query bzgl. Quellenschema	Workflow
Object Matching	offline (während des ETL-Prozesses), sehr gute Datenqualität	online (während der Query-Verarbeitung durch Mediator), mittlere Datenqualität	online (während der Query-Verarbeitung durch Peer-Datenquelle), mittlere Datenqualität	online (während der Workflow-Ausführung) und offline (bei Mapping-Definition), mittlere bis sehr gute Datenqualität
Autonomie der Datenquellen	gering	mittel	mittel	hoch
Aktualität der Daten	mittel, periodische Aktualisierung nötig	hoch, Zugriff auf aktuelle Daten	hoch, Zugriff auf aktuelle Daten	hoch, Zugriff auf aktuelle Daten
Skalierbarkeit bzgl. Anzahl der Quellen	gering	gering	mittel	mittel bis hoch

Tabelle 7.1: Vergleich bestehender Integrationsansätze mit iFuice

Verwendung von Instanz-Mappings führt im Vergleich zu Schema-Mappings zu einer größeren Flexibilität. Erstens lassen sich dadurch auch semi- und unstrukturierte Datenquellen verknüpfen, was bei schemabasierten Mappings durch das notwendige Vorhandensein eines Schemas nicht möglich ist. Zweitens lässt sich ein Instanz-Mapping einfach durch eine Menge von Korrespondenzen, dargestellt als Paare von IDs der beteiligten Objektinstanzen, repräsentieren. Solche Korrespondenzen finden sich z. B. in der Form von Hyperlinks zwischen Webseiten und können somit einfach innerhalb von iFuice (wieder-) verwendet werden. Drittens lässt sich eine „Gleichheitsrelation“, d. h. welche Objektinstanzen in verschiedenen Datenquellen das gleiche Objekt (der realen Welt) darstellen, durch instanzbasierte Mappings spezifischer als durch Schema-Mappings repräsentieren. Letztere beinhalten lediglich korrespondierende Attribute, während Same-Mappings aber auch zwischen Objektinstanzen mit vollkommen unterschiedlichen (d. h. nicht korrespondierenden) Attributen gebildet werden können. Andererseits müssen korrespondierende Attribute mit gleichen Werten nicht zwangsläufig zu gleichen Instanzen führen. Viertens lassen sich durch instanzbasierte Mappings auch assoziierte Informationen modellieren. Es können dadurch auch andere Beziehungstypen durch Instanz-Mappings modelliert werden, z. B. das ein Autor A (einer Datenquelle) die Publikation P (gespeichert in einer anderen Datenquelle) geschrieben hat.

Art der Nutzerinteraktion: Die schemabasierten Ansätze zielen auf eine anfrageorientierte Verwendung durch den Nutzer ab. Data-Warehouse- und Query-Mediator-Ansätze ermöglichen dem Nutzer das Senden einer Anfrage bezüglich des globalen Schemas. Bei PDMS-Ansätzen stellt der Nutzer die Anfrage an (s)ein Peer-Schema. In allen Fällen erhält der Nutzer ein Anfrageergebnis, dessen Instanzdaten aus einer oder mehreren Datenquellen stammen.

Da iFuice sowohl auf ein globales Schema als auch auf Schema-Mappings verzichtet, liegt der Fokus nicht auf der Beantwortung von Nutzeranfragen sondern auf der Ausführung nutzerdefinierter Workflows. Solche Workflows können selbst Anfragen an Datenquellen beinhalten, sofern diese eine Anfrageunterstützung anbieten. Ein weiterer wichtiger Aspekt der Workflows liegt in der Mapping-Verarbeitung, d. h. in der zielgerichteten Verwendung der zur Verfügung stehenden Mappings zur Lösung einer Integrationsaufgabe. Der Nutzer beschreibt durch Skripte einen Workflow zur Verarbeitung von Objektinstanzen und Mappings mit Hilfe von Operatoren, dessen Ergebnisse in Variablen gespeichert werden und wieder verwendet werden können. Er wird dadurch in die Lage versetzt, komplexe Datenintegrationsaufgaben zu formulieren, die Anfragen an Datenquellen und entsprechende Weiterverarbeitungen, z. B. zur Sicherung der Datenqualität, enthalten.

Object Matching: Data-Warehouse-Ansätze realisieren das Object Matching, d. h. das Erkennen gleicher Objektinstanzen, innerhalb des ETL-Prozesses und

damit vor der Ausführung von Anfragen durch den Nutzer (offline). Dabei spezifiziert der ETL-Designer den Algorithmus für das Object Matching, wobei z. B. auf die vom verwendeten Datenbanksystem angebotenen Funktionalitäten (z. B. Fuzzy Match bei Microsoft SQL Server 2005 [31]) oder auf externe Algorithmen (z. B. [56]) zurückgegriffen werden kann. Dadurch wird bei Data-Warehouse-Ansätzen i. Allg. eine sehr gute Datenqualität beim Object Matching erzielt. Bei Query-Mediator- und PDMS-Ansätzen ist der ausgeführte Object-Matching-Algorithmus Teil der Funktionalität des Mediators bzw. des Peers. Die Durchführung des Object Matchings erfolgt online, d. h. erst zur Ausführungszeit der Nutzeranfrage. Dabei werden i. Allg. Ähnlichkeiten der Werte korrespondierender Attribute bestimmt und mittels einer definierten Heuristik entschieden, ob aus den Ähnlichkeitswerten auf gleiche Objektinstanzen geschlossen werden kann oder nicht. Die Qualität solcher generischer Verfahren erreicht dabei i. Allg. nicht die im Data Warehouse erzielte Match-Qualität.

Die Repräsentation von Object-Matching-Ergebnissen erfolgt in iFuice jeweils durch ein Instanz-Mapping, d. h. durch Korrespondenzen zwischen als gleich erkannten Objektinstanzen (sogenannte Same-Mappings). Dadurch ist es möglich, den Prozess des Object Matchings sowohl durch die Verwendung bestehender Mappings (offline) als auch durch die Berechnung neuer Mappings innerhalb eines Workflows (online) zu realisieren. Beide Ansätze sind auch kombinierbar, so dass z. B. innerhalb eines Match-Workflows bestehende Same-Mappings zur Berechnung eines (neuen) Same-Mappings verwendet werden oder das Ergebnis eines Match-Workflows später in anderen Workflows genutzt wird. Die Qualität der resultierenden Same-Mappings variiert dabei in Abhängigkeit von der Art ihrer Erzeugung. Einfache Match-Workflows, die z. B. Attributwerte vergleichen, entsprechen der Qualität von Query-Mediator- bzw. PDMS-Ansätzen. Analog zu ETL-Workflows lassen sich aber auch ausgefeilte sowie domänenspezifische Match-Workflows erstellen und ausführen, die zu einer sehr guten Match-Qualität führen. Zusätzlich erlauben Instanz-Mappings die einfache Erstellung und Wiederverwendung manuell überprüfter Match-Ergebnisse, deren Qualität i. Allg. ebenfalls sehr gut ist.

Autonomie der Datenquellen: Bei der Autonomie wird im Folgenden die Freiheit der Datenquellen bzgl. Repräsentation und Zugriffsmöglichkeit der von ihnen verwalteten Daten betrachtet. Ziel ist eine möglichst große Autonomie der Datenquellen, so dass Änderungen in den Datenquellen möglichst keine Anpassung am Integrationssystem erfordern. Bei Data-Warehouse-Ansätzen liegt eine geringe Autonomie der Datenquellen vor, da sämtliche Daten in das Warehouse (regelmäßig) importiert werden. Damit muss die Datenquelle einen Zugriff auf alle Daten in einem definierten Schema ermöglichen. Eine (interne) Schemaänderung kann evtl. durch entsprechende Views kompensiert werden, so dass das nach außen hin sichtbare (externe) Schema unverändert

bleibt. Größere Schemaänderungen, die nicht mehr durch Sichten gekapselt werden können, erfordern eine entsprechende Anpassung des ETL-Prozesses (u. a. Anpassung des Schema-Mappings). Query-Mediator- und PDMS-Ansätze ermöglichen eine höhere Autonomie der Datenquellen, da nicht der Zugriff auf den kompletten Datenbestand nötig ist, sondern der Zugriff durch Anfragen realisiert wird. Analog zum Data-Warehouse können Schemaänderungen Anpassungen an den betreffenden Schema-Mappings erfordern.

Die generische Datenmodellierung von iFuice erhöht die Autonomie der Quellen im Vergleich zu den anderen Integrationsansätzen. Außer dem ID-Attribut können neue Attribute einfach hinzugefügt oder bestehende entfernt werden. Weiterhin sind Instanz-Mappings „stabiler“ bzgl. Änderungen der internen Datenrepräsentation als Schema-Mappings. So haben z. B. Schemaänderungen keinen Einfluss auf die Mappings, solange die ID-Werte beibehalten werden. Analog zu Query-Mediator- und PDMS-Ansätzen wird auf die Datenquellen durch Anfragen zugegriffen.

Aktualität der Daten: Bei Data-Warehouse-Ansätzen werden die Daten der Quellen innerhalb des ETL-Prozesses in das Warehouse importiert und regelmäßig aktualisiert. Damit werden Anfragen an das Warehouse mit dem Datenbestand der Quellen bei der letzten Aktualisierung verarbeitet. Es ist also nicht sichergestellt, dass bei Anfragen stets mit den aktuellen Daten der Quellen gearbeitet wird.

iFuice greift, wie auch Query-Mediator- und PDMS-Ansätze, direkt auf die Quellen durch Anfragen zu und stellt damit die Verwendung stets aktueller Daten sicher.

Skalierbarkeit bzgl. Anzahl der Quellen: Data-Warehouse- und Query-Mediator-Ansätze verwenden ein globales Schema, so dass die Hinzunahme einer neuen Datenquelle durch ein Schema-Mapping zwischen globalem Schema und Quellenschema realisiert wird. Die Erstellung eines Schema-Mappings gestaltet sich problematisch, wenn z. B. nicht alle Informationen (z. B. Attribute) der Quelle im globalen Schema reflektiert sind oder umgekehrt. Eine etwaige Veränderung des globalen Schemas, um eine neue Quelle anzubinden, kann allerdings eine Anpassung der Schema-Mappings zu den bisherigen Quellen und damit einen großen Aufwand nach sich ziehen. Daher skalieren sowohl Data-Warehouse- als auch Query-Mediator-Ansätze nicht gut bzgl. der Quellenanzahl. PDMS-Ansätze vereinfachen die Verknüpfung einer neuen Quelle, indem nur ein Schema-Mapping zu einer beliebigen anderen Quelle etabliert werden muss. Die Möglichkeit der freien Auswahl der „Partnerquelle“ vereinfacht i. Allg. die Erstellung des Schema Mappings, da eine neue Quelle mit derjenigen Quelle verbunden werden kann, zu der das Mapping am einfachsten realisiert werden kann.

Im Gegensatz zu den drei betrachteten Integrationsansätzen verwendet iFuice

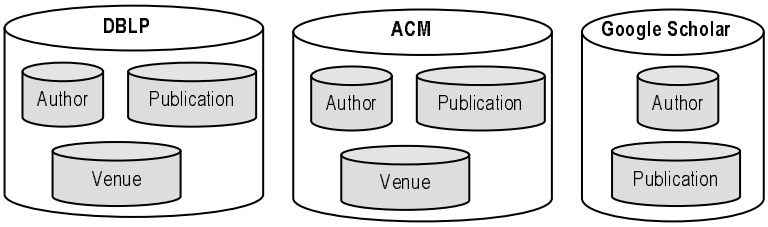


Abbildung 7.1: Beispiel für drei PDS mit mehreren LDS

instanzbasierte Mappings. Neben der daraus resultierenden Möglichkeit, auch semi- und unstrukturierte Datenquellen einzubinden, bieten Instanz-Mappings auch Vorteile bei der Integration strukturierter Datenquellen. Sie ermöglichen die Verknüpfung der Instanzen mit Hilfe von IDs zwischen zwei Datenquellen, auch wenn sich die Quellenschemata stark unterscheiden. Selbst wenn kein Schema-Mapping zwischen den Quellen etabliert werden kann, weil es z. B. keine korrespondierenden Attribute gibt, können die Instanzen durch Korrespondenzen miteinander verbunden werden. Dadurch skaliert der iFuice-Ansatz gut mit der Anzahl der Quellen. Dieser Vorteil erfordert allerdings deutlich größere Mappings. Während Schema-Mappings korrespondierende Schemaelemente repräsentieren und damit nur von der Größe der beiden Schemata (u. a. Anzahl der Attribute) abhängen, sind Instanz-Mappings von der – i. Allg. deutlich größeren – Anzahl der Instanzen in den Quellen abhängig. Gleichzeitig besteht die Notwendigkeit des Vorhandenseins entsprechender Instanzen in den Datenquellen, die durch ein Mapping verknüpft werden können.

7.2 Datenquellen und Mappings

7.2.1 Datenquellen

Datenquellen enthalten i. Allg. unterschiedliche Arten von Informationen. So beinhaltet z. B. die DBLP Bibliography sowohl Informationen über Publikationen als auch über Autoren und Venues[40]. Um dieser Unterscheidung Rechnung zu tragen, werden Datenquellen als *Physische Datenquellen* (phyiscal data source, PDS) bezeichnet. Dies können Webdatenquellen, Datenbanken, Dateien oder andere Informationsquellen sein. Abbildung 7.1 zeigt ein Beispiel für die drei PDS DBLP, ACM und Google Scholar (GS). Die einzelnen Informationseinheiten einer PDS werden als *Objektinstanzen* bezeichnet. Eine Objektinstanz bezieht sich dabei meist auf ein Objekt der realen Welt. Jede Objektinstanz besitzt genau einen Objekttyp, z. B. Pu-

[40] In dieser Arbeit wird mit dem Begriff Venue das Publikationsorgan für wissenschaftliche Papiere gemeint. Der Begriff umfasst die Tagungsbände von Konferenzen und Workshops sowie die Ausgaben von Journals.

Publication@DBLP	
Key	journals/vldb/RahmB01
Title	A survey of approaches to automatic schema matching
Start	334
End	350

Publication@ACM	
Id	P-767154
Name	A survey of approaches to automatic schema matching
Abstract	Schema matching is a basic problem ...

Abbildung 7.2: Beispiele für Objektinstanzen

blikation oder Autor. Die Menge aller Objektinstanzen des gleichen Typs innerhalb einer PDS wird als *Logische Datenquelle* (logical data source, LDS) bezeichnet. Eine LDS wird daher durch Angabe des Objekttyps und der PDS eindeutig definiert. Im Beispiel von Abbildung 7.1 gibt es daher z. B. für die physische Datenquelle DBLP die drei logischen Datenquellen „Publication@DBLP“, „Venue@DBLP“ und „Author@DBLP“.

Die Repräsentation der Objektinstanzen erfolgt durch eine einfache Attribut-Wert-Darstellung. Jede Objektinstanz besteht aus einer Menge von Attributen, die jeweils durch Attributnamen und Attributwert definiert sind. Für jede LDS wird zusätzlich ein ID-Attribut definiert, welches alle Objektinstanzen der LDS eindeutig identifiziert. Das ID-Attribut muss stets einen Wert besitzen, wohingegen alle weiteren Attribute auch unbestimmt sein können. Abbildung 7.2 zeigt je eine Objektinstanz aus den LDS „Publication@DBLP“ und „Publication@ACM“. Das ID-Attribut ist dabei jeweils unterstrichen.

Die verwendete Modellierung ist sehr einfach und stellt – bis auf die Existenz einer ID – kaum Anforderungen an existierende Datenquellen. Es lassen sich verschiedenartige Datenquellen einheitlich modellieren, d. h. sowohl unstrukturierte (z. B. Textdokumente) als auch semistrukturierte (z. B. XML-Dateien, Websites) und strukturierte Datenquellen (z. B. Datenbanken). Weiterhin erlaubt die einfache Modellierung eine leichte Hinzunahme einer neuen Datenquelle und kapselt deren interne Struktur. Eine Strukturveränderung von Datenquellen zieht somit nicht zwangsläufig eine geänderte Modellierung der PDS/LDS nach sich, was die Autonomie der Datenquellen erhöht.

Jede LDS muss lediglich einen Zugriff auf die Objektinstanzen gewähren. Dies wird über eine *GetInstance-Funktion* realisiert, welche für eine gegebene ID die assoziierte Instanz inklusive aller Attribute zurückliefert. Dies ermöglicht das Akquirieren der aktuell verfügbaren Attribute, was insbesondere bei sich regelmäßig ändernden Datenquellen erforderlich ist. Die Implementation kann dabei sehr einfach sein (z. B. ein Nachschlagen in einer Datenbank), aber auch komplexere Verfahren wie z. B. das Extrahieren von Informationen einer Webseite umfassen.

Eine weitere und optionale Form des Datenzugriffs auf eine LDS ist ein sogenanntes *Query-Mapping*. Wie im nachfolgenden Abschnitt erläutert wird, ergibt sich damit die Möglichkeit, Objektinstanzen einer LDS mit Hilfe einer Anfrage zu ermitteln. Query-Mappings sind eine spezielle Form von Mappings, die im nächsten Abschnitt näher vorgestellt werden.

7.2.2 Mappings

Mappings repräsentieren Beziehungen zwischen Objektinstanzen innerhalb der gleichen oder zwischen zwei verschiedenen Datenquellen. Sie ermöglichen eine einheitliche und von der Datenspeicherung unabhängige Repräsentation der Verknüpfung von Instanzen.

Mapping: Ein Mapping m zwischen zwei logischen Datenquellen LDS_A und LDS_B besteht aus einer Menge von Korrespondenzen $\{(a, b, c)|a \in LDS_A \wedge b \in LDS_B \wedge c \in [0, 1]\}$. Dabei bezeichne c die Konfidenz der Beziehung zwischen dem Domain-Objekt a und dem Range-Objekt b.[41] Der Typ des Mappings wird mit $LDS_A \times LDS_B$ notiert. LDS_A und LDS_B werden als Domain- bzw. Range-LDS bezeichnet.

Im Folgenden werden Mappings nach drei Kriterien charakterisiert.

Query- vs. ID-Mappings: Um eine einheitliche Mapping-Verarbeitung zu unterstützen, werden auch Anfrageergebnisse an Datenquellen als Mappings dargestellt. Bei Query-Mappings werden daher Korrespondenzen zwischen einer Anfrage und den resultierenden Objektinstanzen gebildet. Demgegenüber beinhalten ID-Mappings Korrespondenzen zwischen Objektinstanzen.

Same- vs. Assoziations-Mappings: Bei dieser Unterscheidung steht die Semantik im Vordergrund, welche durch eine Mapping-Korrespondenz ausgedrückt wird. Same-Mappings repräsentieren eine „Gleichheitsbeziehung“ zwischen den korrespondierenden Objekten, wohingegen Assoziations-Mappings eine andere semantische Relation darstellen.

Traversierbare vs. Match-Mappings: Das letzte Kriterium zielt auf die Bestimmung der Korrespondenzen ab, d. h. welche Informationen zur Berechnung der Mapping-Korrespondenzen verfügbar sein müssen. Traversierbare Mappings zeichnen sich dadurch aus, dass zur Bestimmung der Korrespondenzen lediglich die Domain-Objekte verwendet werden. Auf der anderen Seite verlangen Match-Mappings sowohl die Angabe aller Domain- als auch aller Range-Objekte zur Ermittlung der Korrespondenzen.

[41] Ist innerhalb dieser Arbeit bei einer Korrespondenz zwischen zwei Objekten a und b die Konfidenz uninteressant, wird die Korrespondenz auch kurz durch (a, b) notiert.

Query@DBLP	Author@DBLP	Conf.
name like "%rahm%"	Erhard Rahm	1
name like "%rahm%"	Gerhard Rahmsdorf	1

Abbildung 7.3: Beispiel für ein Query-Mapping

Query- vs. ID-Mappings

Um eine einheitliche Mapping-basierte Verarbeitung zu gewährleisten, wird auch die Möglichkeit von Anfragen an Datenquellen mittels Mappings realisiert. Dazu wird für jede PDS eine zusätzliche LDS mit dem Objekttyp „Query“ definiert und jede Anfrage wird als Instanz dieser LDS interpretiert. Die Objektinstanz besitzt genau ein Attribut (Anfragestring), das gleichzeitig als ID-Attribut fungiert. Dadurch lässt sich die Query-Funktionaliät einer Datenquelle als *Query-Mapping* wie folgt repräsentieren.

Query-Mapping: Ein Query-Mapping ist ein Mapping des Typs $LDS_A \times LDS_B$ bei dem LDS_A und LDS_B zur selben PDS gehören. Der Objekttyp von LDS_A ist „Query“; der von LDS_B ungleich „Query“.

Demgegenüber stehen *ID-Mappings*, die wie folgt definiert sind.

ID-Mapping: Ein ID-Mapping ist ein Mapping des Typs $LDS_A \times LDS_B$ bei dem weder LDS_A noch LDS_B den Objekttyp „Query“ besitzen.

Abbildung 7.3 zeigt ein Query-Mapping für eine Anfrage bzgl. DBLP-Autoren. Das Ergebnis der Anfrage enthält zwei Objektinstanzen, so dass zwei Korrespondenzen zwischen der Anfrage und der jeweiligen Objektinstanz (identifiziert durch das ID-Attribut) gebildet werden. In Abbildung 7.4 sind zwei Beispiele für ID-Mappings durch eine tabellarische Form (Mapping-Tabelle) illustriert. Jede Zeile steht für eine Korrespondenz, wobei ebenfalls zur Definition der jeweiligen Objektinstanzen die Angabe der ID genügt.

Same- vs. Assoziations-Mappings

Durch ID-Mappings werden Objektinstanzen miteinander in Verbindung gesetzt. Um die Semantik dieser Verbindung näher zu charakterisieren, wird zwischen Same- und Assoziations-Mappings unterschieden.

Same-Mappings verbinden semantisch gleiche Objektinstanzen, d. h. Instanzen, die sich auf das gleiche Objekt der realen Welt beziehen. Sie verbinden zwei LDS (meist aus unterschiedlichen PDS) des gleichen Objekttyps. Ein Beispiel zeigt Abbildung 7.4 (oben), das eine Korrespondenz zwischen dem gleichen Publikationsobjekt bei DBLP und ACM besitzt.

Publication@DBLP	Publication@ACM	Conf.
journals/vldb/RahmB01	P-767154	1

Publication@DBLP	Author@DBLP	Conf.
journals/vldb/RahmB01	Erhard Rahm	1
journals/vldb/RahmB01	Philip A. Bernstein	1

Abbildung 7.4: Beispiel für ein Same-Mapping (oben) und ein Assoziations-Mapping (unten)

Same-Mapping: Ein Same-Mapping ist ein ID-Mapping des Typs $LDS_A \times LDS_B$, wobei LDS_A und LDS_B den gleichen Objekttyp besitzen und zwei Objektinstanzen $a \in LDS_A$ und $b \in LDS_B$ in einer Korrespondenz (a, b, c) auftreten, wenn sich a und b auf das gleiche Objekt der realen Welt beziehen. Die Konfidenz c wird dabei auch als *Ähnlichkeitswert* bezeichnet, wobei $c = 1$ die maximale Ähnlichkeit (Gleichheit) und $c = 0$ die maximale Unähnlichkeit (Ungleichheit) repräsentieren.

Gilt für eine Same-Mapping zusätzlich $LDS_A = LDS_B$, d. h. dass die Korrespondenzen zwischen Objektinstanzen der gleichen logischen Datenquelle gebildet werden, spricht man von einem *Self-Mapping*.

Nicht-Same-Mappings werden als *Assoziations-Mappings* bezeichnet. Sie repräsentieren meist Beziehungen zwischen Objektinstanzen innerhalb der gleichen PDS. Die Semantik der repräsentierten Beziehung wird durch einen Mapping-Typ charakterisiert, der für jedes Mapping im Domänen- bzw. Source-Mapping-Modell angegeben wird (siehe Abschnitt 7.2.3). Dadurch wird u. a. eine Kennzeichnung verschiedener Mappings des gleichen semantischen Typs ermöglicht. Im Beispiel von Abbildung 7.4 (unten) wird die Beziehung „Autor von Publikation“ für eine Publikationsinstanz bei DBLP illustriert. Da die Publikation zwei Autoren besitzt, ergeben sich zwei Korrespondenzen.

Assoziations-Mapping: Ein Assoziations-Mapping ist ein ID-Mapping, das kein Same-Mapping ist.

Im Beispiel der (in Abschnitt 7.2.3 näher erläuterten) Abbildung 7.6 gibt es bei der Datenquelle ACM das Assoziations-Mapping ACM.PubVenue, welches zu einer Publikation das zugehörige Venue ermittelt. Zusätzlich gibt es das umgekehrte Mapping ACM.VenuePub, welches zu einem Venue alle bei ACM gelisteten Publikationen liefert. Das Mapping DBLP.CoAuthor ist ein Beispiel für ein Assoziations-Mapping innerhalb der gleichen LDS, welches für einen Autoren alle seine Koautoren bestimmt.

Traversierbare vs. Match-Mappings

Neben der Semantik der ID-Mappings spielt bei der Datenintegration die Berechnung der Korrespondenzen eine wichtige Rolle. An je einem Beispiel werden dazu die zwei möglichen Fälle für die Bestimmung der Korrespondenzen illustriert. Liegt ein Mapping – wie in Abbildung 7.4 dargestellt – z. B. in Form einer Mapping-Tabelle vor, können für eine Menge von Domain-Objekten die relevanten Korrespondenzen direkt, z. B. durch eine entsprechende Datenbankabfrage, bestimmt werden. Die zugehörigen Range-Objekte werden durch die Mapping-Ausführung sozusagen „on-the-fly“ ermittelt. Auf der anderen Seite kann z. B. ein Same-Mapping auch durch einen Attributvergleich von Objektinstanzen ermittelt werden. Dazu müssen aber alle zu vergleichenden Objektinstanzen definiert werden, d. h. bei der Mapping-Ausführung müssen sowohl alle Domain-Objekte als auch alle Range-Objekte übergeben werden. Es ergeben sich daher die folgenden zwei Arten von ID-Mappings.

Traversierbares Mapping: Ein ID-Mapping m heißt traversierbar, wenn für ein $a \in LDS_A$ die Berechnung aller Korrespondenzen (a, b_i, c_i) ohne Angabe der Range-Objekte b_i erfolgt.

Match-Mapping: Ein ID-Mapping m heißt Match-Mapping, wenn zur Bestimmung der Korrespondenzen sowohl alle Domain- als auch alle Range-Objekte angegeben werden müssen.

Ein traversierbares Mapping besitzt damit eine Richtung, d. h. für eine Menge von Domain-Objekten werden die zugehörigen Range-Objekte ermittelt. Die Richtung des Mappings ist entscheidend, da jedem Mapping bei der Ausführung eine Implementierung zu Grunde liegt (siehe Abschnitt 7.2.3). Analog können Match-Mappings als ungerichtet angesehen werden. Da stets alle Domain- und Range-Objekte angegeben werden müssen, werden für die Domain-Objekte Korrespondenzen zu Range-Objekten bestimmt und umgekehrt für alle Range-Objekte die zugehörigen Domain-Objekte.

7.2.3 Domänen- und Source-Mapping-Modell

Das Domänenmodell definiert die domänenspezifischen Objekttypen und semantischen Mapping-Typen. Dadurch kategorisiert es Datenquellen und Beziehungen zwischen Objektinstanzen und unterstützt eine fokussierte Datenintegration. Abbildung 7.5 zeigt ein Beispiel aus der bibliografischen Domäne. Es besitzt drei Objekttypen (Author, Publication und Venue), die untereinander durch Mapping-Typen verknüpft sind. So können z. B. Instanzen des Objekttyps Author mit Hilfe eines Mappings vom Typ AuthPub entsprechende Publikationen zugeordnet sein. Mappings können dabei auch Instanzen des gleichen Typs miteinander in Verbindung

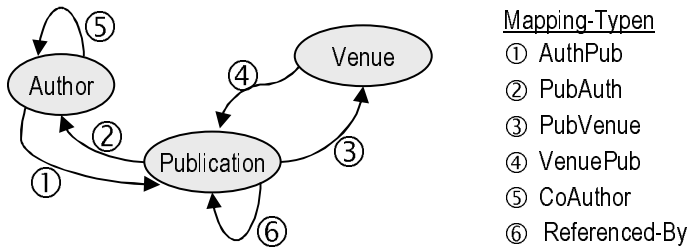


Abbildung 7.5: Beispiel eines Domänenmodell aus der bibliografischen Domäne

setzen, z. B. die Koautoren zu einem Autor. Das Domänenmodell enthält u. a. keine Attribute, was seine Erstellung – insbesondere im Vergleich zu einem globalen Schema – deutlich vereinfacht. Zusätzlich kann in vielen Fällen von einer relativ geringen Zahl von Objekt- und Mapping-Typen innerhalb einer Domäne ausgegangen werden, so dass eine überschaubare Modellierung einer Domäne ermöglicht wird.

Das Domänenmodell enthält lediglich die Objekt- und Mapping-Typen. Alle Datenquellen und Mappings werden in einem zugehörigen Source-Mapping-Modell zusammengefasst. Abbildung 7.6 zeigt ein Beispiel, welches auf dem Domänenmodell von Abbildung 7.5 aufbaut.

Dabei wird die folgende Notation verwendet: PDS werden durch ein weißes Datenbanksymbol gekennzeichnet und durch ihren Namen definiert. Jede PDS enthält eine oder mehrere LDS, die durch ein grau hinterlegtes Datenbanksymbol sowie ihren Objekttyp gekennzeichnet ist. Query-Mappings werden nicht verzeichnet, d. h. insbesondere die zusätzlichen logischen Datenquellen des Objekttyps „Query“ werden nicht extra dargestellt. Stattdessen werden alle LDS, für die ein Query-Mapping existiert, durch Unterstreichung des Objekttyps gekennzeichnet. Sie bilden sozusagen den „Einstiegspunkt“ in das Modell bei der Datenintegration. ID-Mappings werden durch Verbindungslinien zwischen den beteiligten LDS dargestellt. Dabei werden Same-Mappings gestrichelt und Assoziations-Mappings mit einer durchgehenden Linie dargestellt. Für Assoziations-Mappings wird zusätzlich der semantische Mapping-Typ durch einen Namen an der Verbindungslinie angegeben. (Dies erfolgt aus Platzgründen in Abbildung 7.6 durch eine Nummer, die auf der rechten Seite der Abbildung definiert wird.) Traversierbare Mappings haben zusätzlich an der Range-LDS einen Pfeil; Match-Mappings werden durch Linien ohne Pfeil beschrieben. Durch die Pfeile werden somit die Richtungen charakterisiert, in denen Mappings bei deren Ausführung traversiert werden können.

Das in Abbildung 7.6 gegebene Beispiel besteht aus den – bereits in Abschnitt 7.2.1 eingeführten – drei PDS DBLP, ACM und GS. Als Objekttypen sind Author, Publication und Venue definiert und es ergeben sich acht logische Datenquellen. Mittels Query-Mappings können Objektinstanzen für alle LDS der Datenquelle DBLP sowie für alle LDS des Objekttyps Publication durch Anfragen ermittelt werden. Durch Assoziations-Mappings, welche im Beispiel alle traversierbar sind, werden Be-

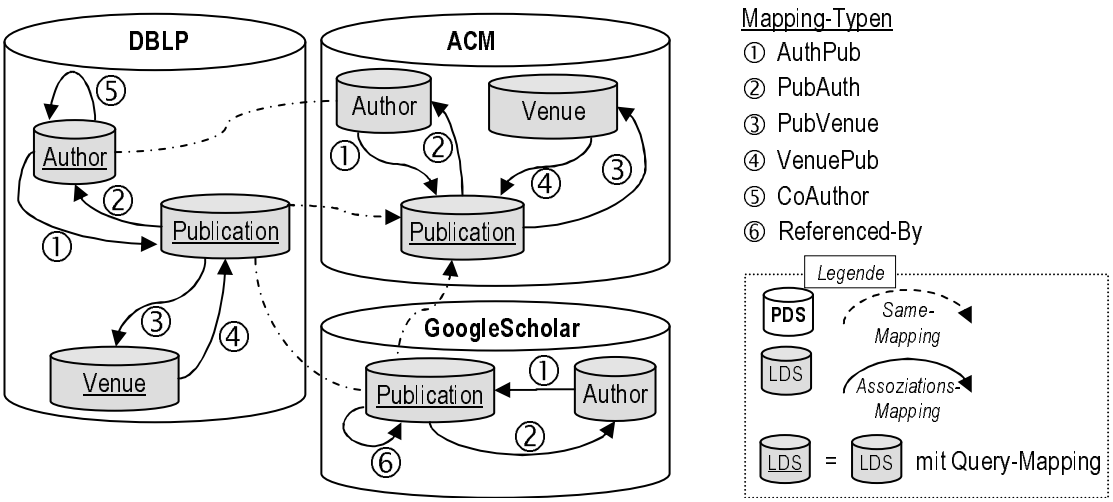


Abbildung 7.6: Beispiel eines Source-Mapping-Modells unter Verwendung des Domänenmodells von Abbildung 7.6

ziehungen zwischen den Objektinstanzen repräsentiert. So können für einen Autor alle Publikationen (Mapping-Typ: AuthPub) und umgekehrt für eine Publikation alle Autoren (PubAuth) ermittelt werden. Analog repräsentieren die Beziehungen zwischen Publikation und Venues (PubVenue bzw. VenuePub), welche Publikationen bei welchen Venues veröffentlicht wurden. Für die LDS Author@DBLP gibt es zusätzlich ein Mapping des Typs CoAuthor, das für einen DBLP-Autoren A alle Koautoren, d. h. alle anderen Autoren, die mit A mindestens eine gemeinsame Publikation verfasst haben, bestimmt. Weiterhin stellt die Datenquelle Google Scholar ein Mapping zur Verfügung, das für eine GS-Publikation alle referenzierenden GS-Publikationen ermittelt.

Das gegegebene Source-Mapping-Modell enthält zusätzlich vier Same-Mappings. Zwischen Publication@DBLP und Publication@ACM gibt es ein traversierbares Same-Mapping, so dass für jede DBLP-Publikation die gleiche ACM-Publikation bestimmt werden kann. Analog gibt es ein traversierbares Same-Mapping von Publication@GoogleScholar zu Publication@ACM, das sich z. B. bei seiner Implementierung die bei manchen GS-Publikationen angebotenen Hyperlinks zu ACM zunutze machen (d. h. extrahieren) kann. Für Publikationen zwischen DBLP und GS steht als Same-Mapping ein Match-Mapping zur Verfügung, das für eine gegebene Menge von DBLP- und GS-Publikationen die Korrespondenzen zwischen gleichen Publikationen ermittelt. Ebenfalls durch ein Match-Mapping können gleiche Autoren zwischen DBLP und ACM erkannt werden.

Das in Abbildung 7.6 dargestellte Source-Mapping-Modell stellt ein mögliches Modell für die bibliografische Domäne dar. Da der iFuice-Ansatz domänenunabhängig ist, können beliebige Source-Mapping-Modelle verwendet werden, deren Metadatenmodell in Abbildung 7.7 als UML-Diagramm angegeben ist. Es erlaubt eine ein-

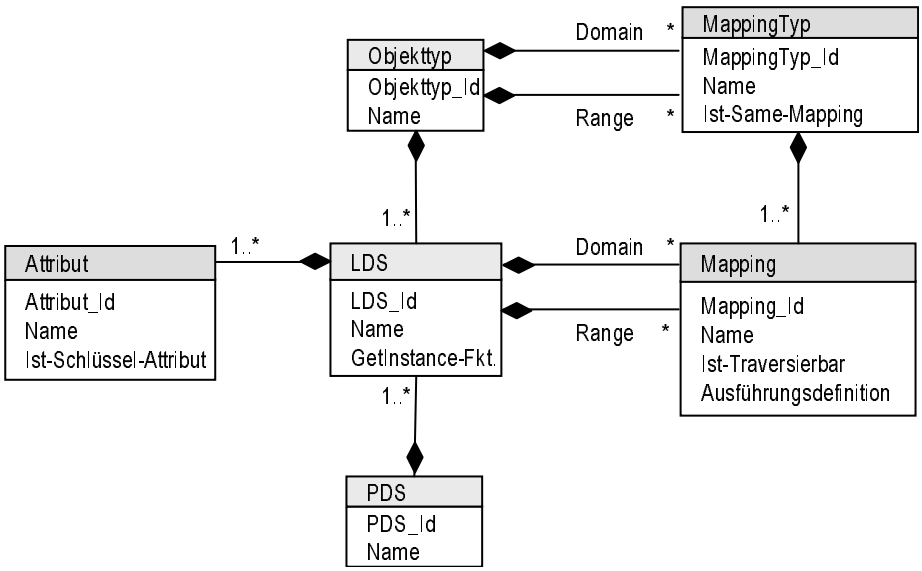


Abbildung 7.7: Metadatenmodell des Source-Mapping-Modells (UML)

heitliche Repräsentation der Datenquellen, Attribute und Mappings. Für jede PDS und LDS wird ein eindeutiger Name definiert. Zusätzlich werden bei jeder LDS die Ausführungsdefinition der GetInstance-Funktion (analog zu Mappings; siehe weiter unten) sowie die verfügbaren Attribute, wobei mindestens das ID-Attribut spezifiziert werden muss, angegeben. Für jedes im Source-Mapping-Modell definierte Mapping gibt es eine Implementation. Neben der Referenz zum Mapping-Typ sowie den Domain- und Range-LDS werden dazu ein eindeutiger Name sowie die Ausführungsdefinition angegeben. Handelt es sich z. B. um ein Mapping, welches durch Absetzen einer SQL-Anfrage ausgeführt wird, enthält die Ausführungsdefinition den Verweis zur Datenbank (z. B. Host, Datenbankname, Nutzername, Passwort) sowie das (ggf. parametrisierte) SQL-Statement. Wird bei der Mapping-Ausführung z. B. eine Java-Klasse zur Ausführung gebracht, werden Name sowie aufzurufende Methode der Java-Klasse hinterlegt.

7.2.4 Hinzufügen neuer Datenquellen und Mappings

Ein Ziel von iFuice ist es, dass Hinzufügen neuer Quellen und Mappings einfach zu gestalten. Um eine neue (physische) Datenquelle in ein bestehendes Source-Mapping- bzw. Domänenmodell einzufügen, muss (mindestens) eine logische Datenquelle registriert werden. Zur Registrierung einer LDS muss der Objekttyp definiert werden. Sollte der Objekttyp noch nicht im Domänenmodell verzeichnet sein, wird das Domänenmodell entsprechend um diesen Objekttyp erweitert. Im Source-Mapping-Modell werden neben dem Namen der LDS das ID-Attribut sowie die Realisierung einer GetInstance-Funktion spezifiziert.

PubID	InstId	Name	Land	Typ
P-1107505	Uni Leipzig	University of Leipzig	Germany	University
P-248616	MS	Microsoft Research	USA	Industry
P-255479	Uni Stanford	Stanford University	USA	University

Abbildung 7.8: Ausschnitt einer Excel-Datei mit Zuordnung zwischen Publikationen (in der Reihenfolge [157, 30, 66]) und Institutionen

Die Verknüpfung der neuen LDS mit mindestens einer anderen LDS erfolgt durch ein ID-Mapping. Für das Hinzufügen eines Mappings muss der semantische Mapping-Typ angegeben werden. Sollte – analog zum Objekttyp – dieser noch nicht im Domänenmodell vorhanden sein, wird er entsprechend ergänzt. Im Source-Mapping-Modell wird ein eindeutiger Name sowie die Realisierung des Mappings, d. h. wie das Mapping zur Laufzeit verwendet werden kann, verzeichnet. Es kann sich dabei z. B. um ein Same-Mapping handeln, um Instanzen der neuen LDS mit den gleichen Instanzen der anderen LDS zu verknüpfen. Es ist aber auch möglich, ein Assoziations-Mapping zwischen zwei LDS zu etablieren. Zusätzlich kann für jede LDS auch ein Query-Mapping bereitgestellt werden. Es ermöglicht durch eine Anfrage das Erzeugen von Instanzen der jeweiligen LDS, die wiederum als Ausgangspunkt zur weiteren Verarbeitung (z. B. durch Mapping-Ausführung) dienen.

Zur Illustration sollen das Domänenmodell (Abbildung 7.5) bzw. das Source-Mapping-Modell (Abbildung 7.6) um die physische Datenquelle Excel erweitert werden. Hierbei handelt es sich um eine Excel-Datei, die für ausgewählte Publikationen die Institution zuordnet, an der die Publikation maßgeblich entstanden ist.[42] Abbildung 7.8 zeigt einen Ausschnitt der Datei mit drei Publikationen. Die PDS Excel besteht dabei aus zwei LDS mit den Objekttypen Publication und Institution. Letzterer Objekttyp tritt noch nicht im Domänenmodell auf und wird daher ergänzt. Zur Registrierung der LDS werden anschließend die ID-Attribute der Publikationen (PubID) und Institutionen (InstID) angegeben.

Zur Verknüpfung der beiden Datenquellen werden drei Mappings erstellt. Ein Query-Mapping für Publication@Excel ermöglicht eine Suche nach verfügbaren Publikationen. Ein Assoziations-Mapping Excel.PubInst liefert für eine Publikation die zugehörige Institution. Dazu wird das Domänenmodell um einen entsprechenden Mapping-Typ erweitert. Abschließend verbindet ein Publikations-Same-Mapping die LDS Publication@Excel mit Publication@ACM. Dieses Mapping ist im Beispiel sehr einfach zu erstellen, da für das Erstellen der Excel-Datei die Publikationsvolltexte von ACM genutzt wurden und somit als ID für die Excel-Publikationen die jeweilige ID der zugehörigen ACM-Publikationen verwendet wurde.

Abbildung 7.9 zeigt das erweiterte Source-Mapping-Modell. Das Domänenmodell ist im Vergleich zu 7.5 um den Objekttyp Institution sowie um den Mapping-Typ

[42] Zur Vereinfachung wird – analog zu [157] – jede Publikation nur der Institution des Erstautors zugeordnet.

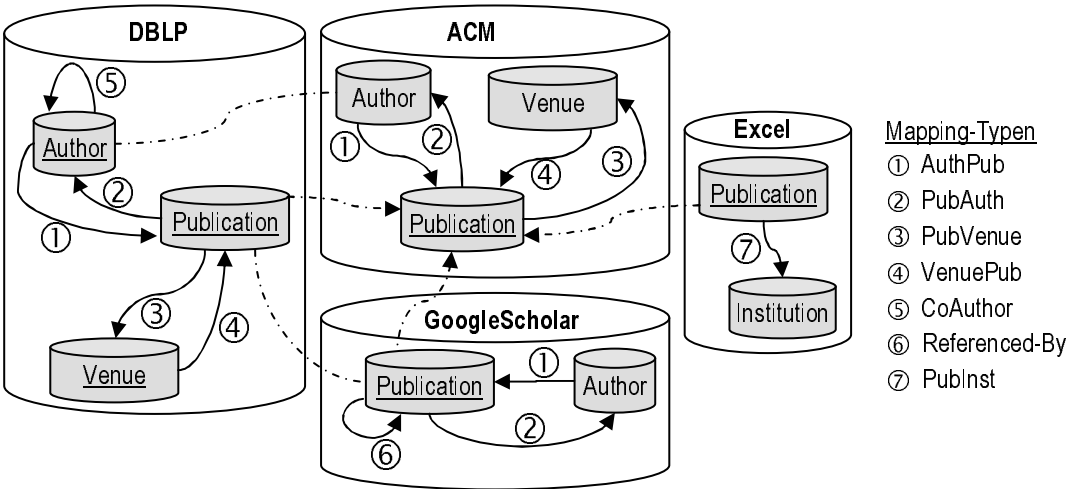


Abbildung 7.9: Erweitertes Source-Mapping-Modell aus Abbildung 7.6

PubInst erweitert. Es verdeutlicht, dass sich durch die Anbindung der Excel-Datei an die ACM-Datenquelle auf Grund der bestehenden Mappings weitreichende Verknüpfungen ergeben. So können z. B. für Institutionen die zugehörigen Google-Scholar-Publikationen inklusive ihrer Zitierungszahl bestimmt werden. Dadurch ist es z. B. möglich, Institutionen bzgl. ihrer Zitierungszahlen miteinander zu vergleichen.

7.3 Workflow-basierte Datenintegration

Schemabasierte Datenintegrationsansätze ermöglichen Anfragen an das globale Schema bzw. an das Schema einer Quelle. Mit Hilfe von Schema-Mappings werden die Anfragen transformiert, in verschiedenen Quellen ausgeführt und die Ergebnisse zusammengeführt. Diese anfragebasierte Art der Nutzerinteraktion ist beim instanzbasierten iFuice-Ansatz nicht möglich, da keine Schema-Mappings vorliegen.

Stattdessen verwendet iFuice einen Workflow-basierten Ansatz, der eine koordinierte Ausführung mengenwertiger Operatoren innerhalb von Skripten ermöglicht. Diese Art der Steuerung ist inspiriert vom Model-Management [14] und wird u. a. im Prototypen Rondo [124] angewendet. Dabei werden Schemata und Schema-Mappings mittels High-Level-Operatoren (z. B. `compose`, `merge` und `match`) manipuliert, um z. B. neue Schema-Mappings effizient zu erstellen oder bei Schemaänderungen bestehende Mappings zwischen den alten Schemata anzupassen. Im Unterschied zum Model-Management operiert iFuice auf Instanzdaten. Durch die Definition von Skripten können sowohl instanzbasierte Mappings erstellt als auch zur Informationsfusion genutzt werden. Die innerhalb der Skripte zur Verfügung stehenden Operatoren zeichnen sich durch folgende Merkmale aus:

Transparenter Datenzugriff: Die Operatoren realisieren einen transparenten Zugriff auf die Instanzen bzw. Korrespondenzen unabhängig von den Datenquellen. So wird z. B. zur Mapping-Ausführung einheitlich der `map`-Operator angewendet, welcher den Namen des Mappings als Parameter übergeben bekommt. Ein Mediator (siehe Abschnitt 7.5) übernimmt dann die konkrete Ausführung, die z. B. durch eine SQL-Query oder das Aufrufen einer Java-Klasse realisiert werden kann.

Generische Datenstrukturen: Alle Operatoren arbeiten mit generischen Datenstrukturen, die eine einheitliche Repräsentation von Objekten, Attributen und Korrespondenzen realisieren. Dadurch wird eine uniforme Verarbeitung der Daten gewährleistet, was zu großer Flexibilität bei der Kombination von Operatoren innerhalb von Workflows führt. Es lassen sich u. a. die Ergebnisse eines Operatoraufrufs wieder als Aufrufparameter für den Aufruf weiterer Operatoren verwenden.

Polymorphismus: Jeder Operator lässt sich mit Parametern unterschiedlichen Typs aufrufen. Dadurch können insbesondere bereits ermittelte Ergebnisse wieder verwendet werden. Für den Operator `map` bedeutet das zum Beispiel, dass neben dem Mapping-Namen auch ein bereits bestimmtes Mapping, welches z.B innerhalb eines anderen Skriptes berechnet wurde, verwendet werden kann. Je nach Art des Parameters wählt der Mediator einen anderen Ausführungsplan des Operators. Gleichzeitig ist es dem Nutzer dadurch möglich, Skripte sowohl sehr spezifisch zu definieren, z. B. die Ausführung eines konkreten Mappings zu erzwingen, als auch ohne genauere Kenntnisse der Quellen und Mappings, z. B. durch Angabe des Mapping-Typs oder der Zieldatenquelle, zu arbeiten. In letzterem Fall setzt iFuice den Aufruf in eine Folge konkreter Mapping-Ausführungen um.

Mengenbasiert: Alle Operatoren sind mengenbasiert, d. h. sie operieren auf Mengen von Objekten und Korrespondenzen. Dadurch können die gleichen Skripte sowohl für wenige Daten, z. B. innerhalb einer Online-Anwendung, als auch für große Datenmengen im Rahmen einer Offline-Verarbeitung verwendet werden.

Nutzerdefinierte Prozeduren: iFuice ermöglicht die nutzerdefinierte Definition neuer Prozeduren durch Workflows. Prozeduren stellen parametrisierte Anweisungsfolgen bestehender Operatoren oder Prozeduren dar und liefern bei ihrer Ausführung einen Rückgabewert zurück. Dadurch können Prozeduren analog zu Operatoren in Skripten aufgerufen werden.

Operatoren werden zu Skripten kombiniert, indem verschiedene Operatoren sequenziell oder innerhalb einer Schleife ausgeführt und deren Ergebnisse in Variablen gespeichert werden. Die Variablen können als Parameter für spätere Operatoren verwendet werden, so dass eine prozedurale Programmierung ermöglicht wird. Im

Folgenden werden die Datenstrukturen definiert, die durch Operatoren verarbeitet werden, bevor die verschiedenen Arten von Operatoren vorgestellt werden.

7.3.1 Datenstrukturen

Die durch die Query- und ID-Mappings erzeugten Objektinstanzen und Korrespondenzen werden generisch repräsentiert und weiterverarbeitet. Dabei werden – wie bei den Operatoren beschrieben – innerhalb der Skripte stets Mengen von Objekten bzw. Korrespondenzen verarbeitet. Die iFuice-Skriptsprache stellt daher zwei Basisdatentypen zur Verfügung.

Object Instances (O) ist eine Menge von Objektinstanzen einer LDS. Sie besitzen daher alle denselben Objekttyp und stammen aus derselben PDS. Weiterhin definiert die LDS ein ID-Attribut, für welches jedes Objekt einen nichtleeren Wert besitzt. Zusätzlich hat jedes Objekt eine (möglicherweise leere) Liste von weiteren Attributen.

Mapping Result (MR) ist eine Menge von Korrespondenzen zwischen Objekten. Dabei stammen sowohl alle Domain-Objekte als auch alle Range-Objekte jeweils aus der gleichen LDS. Jede Korrespondenz besitzt mit der Konfidenz ein „Korrespondenzattribut“, das die Güte der Korrespondenz charakterisiert. Die Konfidenz ist eine reelle Zahl im Intervall [0,1], welche die Sicherheit, Glaubwürdigkeit oder Zuversicht der Korrespondenz definiert. Dabei bedeutet der Wert 1 (0) vollständige (Un-)Sicherheit.

Die Definitionen von Object Instances und Mapping Result ähneln denen von LDS und Mapping, weshalb im Folgenden kurz die Unterschiede bzw. Besonderheiten erläutert werden.

Eine Variable vom Typ Object Instances repräsentiert i. Allg. eine Teilmenge einer LDS, d. h. sie beinhaltet einen Teil der Objektinstanzen einer LDS. Dies ermöglicht eine zielgerichtete Mapping-Verarbeitung für einen relevanten Teil der verfügbaren Daten. Sollen z. B. die DBLP-Publikationen einer ausgewählten Menge von Autoren, z. B. der Mitglieder einer Arbeitsgruppe, ermittelt werden, so kann zunächst die Menge der relevanten Autoren durch eine entsprechende Anfrage an die LDS Author@DBLP bestimmt und innerhalb einer Variable des Typs Object Instances gespeichert werden. Nur für diese Objekte braucht dann anschließend das Mapping DBLP.AuthorPub ausgeführt werden.

Das Ergebnis der Mapping-Ausführung ist dann ein Mapping Result, das in diesem Fall einen Teil des Mappings DBLP.AuthorPub repräsentiert. Abbildung 7.10 (links oben) zeigt einen entsprechenden Ausschnitt für zwei ausgewählte Autoren. Ein Mapping Result ist dabei stets ein traversierbares ID-Mapping, d. h. auch das

Author@DBLP	Publication@DBLP	Conf.
Erhard Rahm	journals/vldb/RahmB01	1
Erhard Rahm	conf/vldb/StohrMR00	1
Philip A. Bernstein	journals/vldb/RahmB01	1

Publication@DBLP	Venue@DBLP	Conf.
journals/vldb/RahmB01	journals/vldb/2001	1
conf/vldb/StohrMR00	conf/vldb/2000	1
journals/vldb/RahmB01	journals/vldb/2001	1

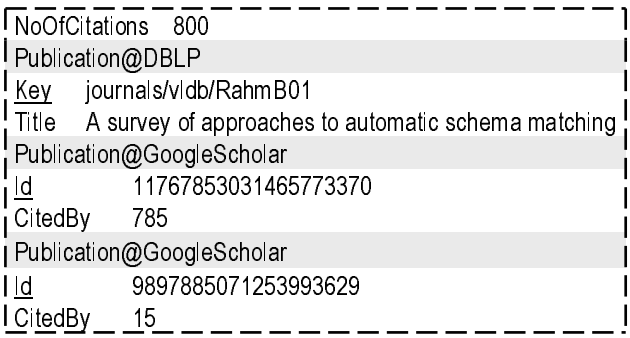
Author@DBLP	Venue@DBLP	Conf.
Erhard Rahm	journals/vldb/2001	1
Erhard Rahm	conf/vldb/2000	1
Philip A. Bernstein	journals/vldb/2001	1

Abbildung 7.10: Beispiel zur Bildung neuer Mappings durch Komposition

Ergebnis der Ausführung eines (nicht traversierbaren) Match-Mappings lässt sich später als traversierbares Mapping weiterverarbeiten. Neben der Tatsache, dass ein Mapping Result ein Teil eines Mappings (d. h. eine Teilmenge der Korrespondenzen) darstellt, kann ein Mapping Result auch Korrespondenzen zwischen Objektinstanzen enthalten, die in keinem der Mappings aus dem Source-Mapping-Modell auftreten. Im genannten Beispiel von Abbildung 7.10 werden für die ausgewählten Autoren das Mapping DBLP.AuthorPub mit dem Mapping DBLP.PubVenue durch Komposition verknüpft, d. h. zu den Publikationen der Autoren werden die assoziierten Venues bestimmt. Es resultieren Korrespondenzen zwischen Autoren und Venues, wenn der Autor mindestens an einer Publikation, die bei dem Venue erschienen ist, beteiligt war (siehe Abbildung 7.10 (unten)). Das Mapping Result repräsentiert dadurch neue Korrespondenzen, deren Semantik durch keines der im Domänen- bzw. Source-Mapping-Modell aufgeführten Mapping-Typen erfasst wird.

Neben der Arbeit mit Objektinstanzen und Korrespondenzen ermöglicht iFuice zusätzlich die Aggregation von als gleich erkannten Objektinstanzen zu *aggregierten Objekten*. Es können dabei mehrere Objektinstanzen – aus der gleichen oder unterschiedlichen Datenquellen – des gleichen Objekttyps zusammengefasst und gemeinsam verarbeitet werden. Die Aggregation ermöglicht dabei zum einen eine Informationsfusion, d. h. die Zusammenfassung aller – als Attribute repräsentierten – Informationen zu einem (realen) Objekt in verschiedenen Datenquellen. Darüber hinaus versetzt die Aggregation den Nutzer in die Lage, Skripte auf einem „abstrakteren“ Level zu formulieren. So können (fast) alle Operatoren, die auf Objektinstanzen angewendet werden können, z. B. die Ausführung eines Mappings, auch auf aggregierten Objekten ausgeführt werden. Ein Operatoraufruf auf aggregierten Objekten führt dabei automatisch zu einer mehrfachen Operatorausführung auf den einzelnen Objektinstanzen mit anschließender Aggregation der resultierenden Daten. Zu diesem Zweck definiert iFuice die folgenden zwei aggregierten Datentypen.

Aggregated Objects (AO) ist eine Menge aggregierter Objekte, die wiederum jeweils eine nicht-leere Menge von Objekten des gleichen Objekttyps (nicht notwendigerweise der gleichen LDS) darstellen. Zusätzlich zu den Attributen der



Publication@DBLP	Author@DBLP	Conf
[journals/vldb/RahmB01, 11767853031465773370, 9897885071253993629]	[Erhard Rahm]	1
[journals/vldb/RahmB01, 11767853031465773370, 9897885071253993629]	[Philip A. Bernstein]	1

Abbildung 7.11: Beispiel für ein aggregiertes Objekt (oben) und ein Aggregated Mapping Result (unten)

Objekte kann jedes aggregierte Objekt noch sogenannte fusionierte Attribute besitzen, welche keiner Objektinstanz sondern ausschließlich dem aggregierten Objekt zugeordnet sind.

Aggregated Mapping Result (AMR) ist eine Menge von Korrespondenzen zwischen aggregierten Objekten. Analog zu MR besitzen alle Domain- und alle Range-Objekte jeweils denselben Objekttyp. Alle Korrespondenzen werden ebenfalls mit einer Konfidenz annotiert.

Abbildung 7.11 (oben) zeigt ein Beispiel für ein aggregiertes Objekt des Typs Publikation. Es besteht aus drei Objektinstanzen, wobei eine Publikation aus der LDS Publication@DBLP und die beiden anderen aus Publication@GoogleScholar stammen. Neben den Attributen der Objektinstanzen besitzt das Beispiel ein fusioniertes Attribut „NoOfCitation“, welches ausschließlich dem aggregierten Objekt zugeordnet ist. Im gezeigten Beispiel wurde das fusionierte Attribut als Summe der Zitierungsattribute der beiden Google-Scholar-Publikationen bestimmt. Die Erstellung fusionierter Attribute ähnelt der in [20, 22] vorgestellten Datenfusion mittels Resolutionsfunktionen (`FUSE BY`-Operator). Das aggregierte Objekt besitzt keine eigene ID, sondern wird eindeutig durch die Menge der IDs der zugehörigen Objektinstanzen identifiziert. Durch diese Menge wird es auch in einem AMR referenziert, wofür ebenfalls ein Beispiel in Abbildung 7.11 (unten) gegeben ist. Die Struktur eines AMR entspricht der eines MR. Das Beispiel zeigt zwei Korrespondenzen zwischen dem bereits angesprochenen aggregierten Publikationsobjekt und je einem aggregierten Autor-Objekt, welche jeweils aus genau einer DBLP-Objektinstanz bestehen. Im folgenden Abschnitt zu den iFuice-Operatoren wird u. a. gezeigt, wie ein solches AMR gebildet werden kann und wie z. B. zu den DBLP-Autoren noch die zugehörigen ACM-Autoren „hinzuaggregiert“ werden können.

7.4 Operatoren

Das Kernelement der durch Skripte definierten iFuice-Workflows bilden die Operatoraufrufe. Ein Aufruf erfolgt in der Form Y `:= operator(`X_1`, ..., `X_n`)`, wobei `operator` durch den jeweiligen Operatornamen ersetzt wird. Die Parameter X_1 bis X_n variieren je nach Operator und sind meist vom Typ der vier Datenstrukturen O, MR, AO und AMR. Zusätzlich gibt es noch weitere Parameter, z. B. Zahlen, Strings oder Konstanten. Y bezeichnet den Rückgabewert, der für die meisten Operatoren ebenfalls von einem Typ der vier wesentlichen iFuice-Datenstrukturen ist. Dadurch sind die Operatoren flexibel kombinierbar, da das Ergebnis eines Operatoraufrufs als Parameter für einen anderen Operatoraufruf genutzt werden kann. Üblicherweise werden dazu Rückgabewerte von Operatoren in Variablen gespeichert und die Variablen als Parameter für spätere Operatoraufrufe verwendet. Es ist aber auch möglich, den Operatoraufruf direkt als Parameter eines anderen Operators „einzusetzen".

iFuice besitzt sieben Klassen von Operatoren, die in Tabelle 7.2 (ab Seite 143) mit allen zugehörigen Operatoren aufgelistet sind und im Folgenden kurz beschrieben werden.

Query-Operatoren werden verwendet, um Anfragen an Datenquellen zu stellen. iFuice unterstützt dabei einfache Anfragen mittels Bedingungen, die auch auf Ergebnisse (d. h. O, MR, AO, AMR) angewendet werden können. Dabei kann sowohl auf die Attributwerte der Objektinstanzen als auch auf fusionierte Attribute aggregierter Objekte zurückgegriffen werden. Zusätzlich kann bei MR und AMR die Konfidenz der Korrespondenzen herangezogen werden. Geben z. B. die Konfidenzwerte eines Same-Mappings die Ähnlichkeit der korrespondierenden Objekte an (z. B. ermittelt durch einen Attributwertvergleich), erlaubt z. B. die Filterung aller Korrespondenzen mit einem Konfidenzwert größer als 80% eine Identifikation „sicherer" Korrespondenzen (siehe Kapitel 9).

Attribut-Operatoren dienen dazu, die Attribute der Objektinstanzen zu manipulieren. Dabei können Attributwerte für Objektinstanzen aktualisiert werden, d. h. es werden die in der zugehörigen Datenquelle aktuell verfügbaren Attribute übernommen. Des Weiteren können sowohl neue Attribute erstellt als auch bestehende gelöscht werden. So kann z. B. das Attribut *Seitenzahl* aus der Differenz der Attributwerte *Startseite* und *Endseite* einer Publikations-Instanz berechnet werden. Bei aggregierten Objekten lassen sich analog fusionierte Attribute erstellen und verarbeiten.

Mapping-Operatoren führen ID-Mappings aus, d. h. für gegebene Domain-Objekte – und Range-Objekte bei Match-Mappings – werden die Korrespondenzen des Mappings ermittelt. Traversierbare Mappings können zusätzlich traversiert werden, d. h. es werden „nur" die korrespondierenden Range-Objekte

bestimmt und zurückgeliefert. Statt den im Source-Mapping-Modell gespeicherten Mappings können auch Variablen vom Typ MR oder AMR verwendet werden, da jedes MR bzw. AMR ein traversierbares Mapping darstellt. Weiterhin beinhaltet iFuice auch einen eingebauten Attribut-Matcher, der Korrespondenzen zwischen Objekten anhand syntaktischer Ähnlichkeit von Attributwerten bestimmt.

Aggregations-Operatoren ermöglichen das Zusammenfassen von Objektinstanzen des gleichen Objekttyps zu aggregierten Objekten. Dadurch können z. B. Instanzen, die das gleiche Objekt der realen Welt beschreiben, zusammengefasst werden. Aggregation kann mittels Deaggregation umgekehrt werden, so dass die aggregierten Objekte wieder in ihre Ursprungsinstanzen zerlegt werden. Analog können Mappings in aggregierte Mappings umgewandelt werden und umgekehrt. Die Polymorphie der anderen Operatoren ermöglicht eine gleiche Verarbeitung von Objektinstanzen/Mappings und aggregierten Objekten/Mappings, d. h. wie in Tabelle 7.2 dargestellt, lassen alle anderen Operatoren neben Aufrufparametern vom Typ O und MR auch die entsprechenden aggregierten Varianten AO und AMR zu.

Kombinations-Operatoren führen einfache mengentheoretische Operationen auf Variablen des gleichen Typs aus. Es können Durchschnitt, Vereinigung und Differenz von zwei oder mehr Variablen bestimmt werden. Zusätzlich kann für zwei Mappings die Komposition („Hintereinanderausführung“) ermittelt werden. Bei der Bildung von Durchschnitt, Vereinigung und Komposition von Mappings wird mittels Konfidenzfunktionen – die als Parameter übergeben werden – definiert, wie sich der Konfidenzwert der resultierenden Korrespondenzen aus den Ursprungskorrespondenzen bestimmt.

Hilfs-Operatoren dienen allgemein der Verarbeitung von Zwischenergebnissen. So können z. B. aus Mappings Domain- oder Range-Objekte extrahiert werden oder das inverse Mapping bestimmt werden.

Funktionen liefern skalare Werte, z. B. die Anzahl der Korrespondenzen eines Mappings.

Tabelle 7.2 enthält neben einer kurzen Funktionsbeschreibung zusätzlich für jeden Operator auch eine vereinfachte Signatur, d. h. eine vereinfachte Darstellung der möglichen Aufrufparametertypen sowie des Rückgabewertes. Die Vereinfachung geschieht dabei durch das Fokussieren auf die wichtigen Parameter, die zum Verständnis der Operatoren nötig sind. Eine vollständige Auflistung aller möglichen Parametrisierungen findet sich im Anhang A.

Kurzbescheibung	Signatur (vereinfacht)
Query-Operatoren	
`queryInstances`: Bestimmung von Objekten mittels einer Bedingung (Cond)	$LDS \times Cond \rightarrow O$ $O \times Cond \rightarrow O$ $AO \times Cond \rightarrow AO$
`queryMap`: Bestimmung von Korrespondenzen mittels einer Bedingung	$MR \times Cond \rightarrow MR$ $AMR \times Cond \rightarrow AMR$
`join`: Bestimmung von Korrespondenzen zwischen Objektmengen mittels Join-Bedingung	$O \times O \times Cond \rightarrow MR$ $AO \times AO \times Cond \rightarrow AMR$
Attribut-Operatoren	
`getInstances`: Ermittlung aller in der jeweiligen LDS verfügbaren (aktuellen) Attributwerte für die übergebenen Objektinstanzen	$O \rightarrow O$ $MR \rightarrow MR$ $AO \rightarrow AO$ $AMR \rightarrow AMR$
`setAttr`: Erstellung eines neuen nutzerdefinierten Attributs	$O \times name \times value \rightarrow O$ $MR \times name \times value \rightarrow MR$ $AO \times name \times value \rightarrow AO$ $AMR \times name \times value \rightarrow AMR$
`delAttr`: Löschen eines Attributs (außer ID-Attribut)	$O \times name \rightarrow O$ $MR \times name \rightarrow MR$ $AO \times name \rightarrow AO$ $AMR \times name \rightarrow AMR$
Mapping-Operatoren	
`traverse`: Traversierung eines traversierbaren Mappings für eine gegebene Menge von Objekten, d. h. Bestimmung aller korrespondierenden Range-Objekte	$O \times map \rightarrow O$ $O \times MR \rightarrow O$ $O \times PDS \rightarrow O$ $AO \times maptype \rightarrow AO$ $AO \times AMR \rightarrow AO$
`map`: Ausführung eines traversierbaren Mappings für eine gegebene Menge von Objekten, d. h. Bestimmung aller zugehörigen Korrespondenzen	$O \times map \rightarrow MR$ $O \times MR \rightarrow MR$ $O \times PDS \rightarrow MR$ $AO \times maptype \rightarrow AMR$ $AO \times AMR \rightarrow AMR$
`match`: Ausführung eines Match-Mappings, d. h. Bestimmung aller Korrespondenzen zwischen zwei gegebenen Objektmengen	$O \times O \times map \rightarrow MR$ $MR \times map \rightarrow MR$
`attrMatch`: Ausführung des generischen Attributmatchers, d. h. Bestimmung von Korrespondenzen zwischen Objektmengen mittels Attributwertähnlichkeiten (*sim* definiert zu vergleichende Attribute sowie Ähnlichkeitsfunktion)	$O \times O \times sim \rightarrow MR$ $MR \times sim \rightarrow MR$ $AO \times AO \times sim \rightarrow AMR$ $AMR \times sim \rightarrow AMR$

Kurzbescheibung	Signatur (vereinfacht)
Aggregations-Operatoren	
`aggregate`: Bildung aggregierter Objekte sowie aggregierter Korrespondenzen mittels eines Same-Mappings	$O \times MR \rightarrow AO$ $MR \times MR \rightarrow AMR$ $AO \times MR \rightarrow AO$ $AMR \times MR \rightarrow AMR$
`disAgg`: Deaggregation aggregierter Objekte bzw. aggregierter Korrespondenzen, d. h. Extraktion von Objektinstanzen bzw. Korrespondenzen	$AO \times LDS \rightarrow O$ $AMR \times LDS \times LDS \rightarrow MR$
Kombinations-Operatoren	
`union`, `intersect`, `diff`: Bildung der Vereinigung (`union`), des Durchschnitts (`intersect`) sowie der Differenz (`diff`) von Objekten und Korrespondenzen	$O \times O \rightarrow O$ $MR \times MR \rightarrow MR$ $AO \times AO \rightarrow AO$ $AMR \times AMR \rightarrow AMR$
`compose`: Komposition, d. h. Verkettung, zweier (aggregierter) Mappings	$MR \times MR \rightarrow MR$ $AMR \times AMR \rightarrow AMR$
Hilfs-Operatoren	
`domain`, `range`: Bestimmung der Domain- bzw. Range-Objekte von Korrespondenzen	$MR \rightarrow O$ $AMR \rightarrow AO$
`inverse`: Bestimmung des inversen Mappings, d. h. Vertauschung der Domain- und Range-Objekte von Korrespondenzen	$MR \rightarrow MR$ $AMR \rightarrow AMR$
`exec`: Ausführen einer iFuice-Prozedur, deren Name als erster Parameter übergeben wird. Die weiteren Parameter sowie der Rückgabewert richten sich nach der aufzurufenden Prozedur.	$S \times X_1 \times \ldots \times X_n \rightarrow Y$
Funktionen	
`count`: Ermittlung der Anzahl von Objekten bzw. Korrespondenzen	$O \rightarrow I$ $MR \rightarrow I$ $AO \rightarrow I$ $AMR \rightarrow I$
`print`: String-Serialisierung von Objekten bzw. Korrespondenzen (zur textuellen Ausgabe)	$O \rightarrow S$ $MR \rightarrow S$ $AO \rightarrow S$ $AMR \rightarrow S$

Tabelle 7.2: Übersicht der iFuice-Operatoren

Die Kombinations-Operatoren werden innerhalb des Kapitels zum Object Matching mit MOMA (Abschnitt 9.3) ausführlich vorgestellt. Hilfs-Operatoren und Funktionen dienen der „Abrundung“ der Operatorenliste zur Vereinfachung der Erstellung von Skripten. Daher werden im Folgenden lediglich die ersten vier Operatorklassen näher vorgestellt. Wie bereits erwähnt, findet sich eine vollständige Darstellung aller Operatoren mit ihren Aufrufparametern inklusive Beispielen im Anhang A. Es werden daher an dieser Stelle nur ausgewählte Operatoren der jeweiligen Klasse näher vorgestellt, um die grundlegende Funktionsweise zu illustrieren. Das Zusammenspiel verschiedener Operatoren innerhalb eines Skriptes wird in Kapitel 8 im Rahmen eines Anwendungsfalls zur Zitierungsanalyse vorgestellt.

7.4.1 Query-Operatoren

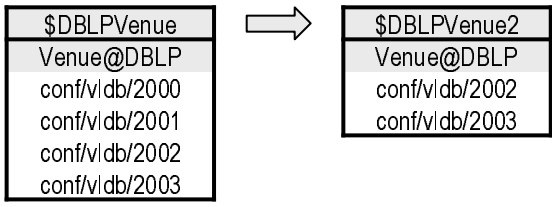
Ein wichtiger Vertreter dieser Klasse ist der Operator `queryInstances`. Er dient zur Ermittlung von Objektinstanzen auf Basis von einfachen Anfragen mit Bedingungen. Die Anfragen können dabei direkt an eine LDS oder eine Variable vom Typ O gestellt werden. Im ersten Fall wird die Anfrage durch ein entsprechendes Query-Mapping verarbeitet, wohingegen im zweiten Fall die Anfrage direkt durch den Operator beantwortet wird. Die Query-Syntax richtet sich nach der verwendeten Quelle bzw. nach der vom entsprechenden Query-Mapping zur Verfügung gestellten Anfrageschnittstelle. Es können also z. B. SQL-Bedingungen angegeben werden (bei relationalen Datenbanken) oder Suchbegriffe (bei Suchmaschinen, z. B. Google Scholar). Das folgende Beispiel zeigt eine Anfrage an die LDS Venue@DBLP, welche alle VLDB-Konferenzen der Jahre nach 1999 bis einschließlich 2003 ermittelt.

Beispiel:
```
$DBLPVenue := queryInstances (Venue@DBLP, "[series] =
                  'VLDB' and [year]>1999 and [year]<=2003")
```

$DBLPVenue
Venue@DBLP
conf/vldb/2000
conf/vldb/2001
conf/vldb/2002
conf/vldb/2003

Das Ergebnis umfasst vier Objektinstanzen des Typs Venue@DBLP, die in der Abbildung jeweils durch ihren ID-Wert charakterisiert sind. Um eine Menge von ermittelten Objektinstanzen weiter einzuschränken, kann `queryInstances` auch auf eine Variable des Typs O angewendet werden.

Beispiel: `$DBLPVenue2 := queryInstances ($DBLPVenue, "[year]>=2002")`

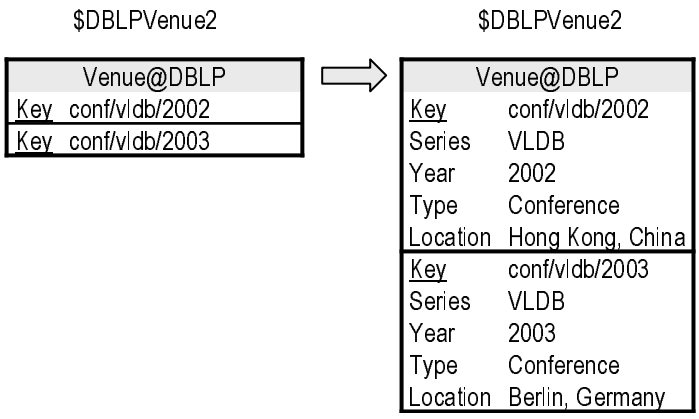


In obigem Beispiel wurde die Menge der DBLP-Venues auf diejenigen ab dem Jahr 2002 eingeschränkt.

7.4.2 Attribut-Operatoren

Mit `getInstances` werden die aktuellen Attribute für Objektinstanzen bestimmt. Dazu wird die entsprechende GetInstance-Funktion der LDS aufgerufen. Die Realisierung dieser GetInstance-Funktion obliegt der Datenquelle, so dass z. B. mittels einer SQL-Query in einer relationalen Datenbank oder per Informationsextraktion von einer Website die Informationen ermittelt werden.

Beispiel: `getInstances ($DBLPVenue2)`



Im Beispiel werden für alle Instanzen der Variable `DBLPVenue2` die aktuellen in der Datenquelle verfügbaren Attribute ermittelt. `getInstances` kann auch für die anderen drei Datenstrukturen aufgerufen werden, wobei dann für Objektinstanzen aus verschiedenen Datenquellen die aktuellen Attribute ermittelt werden. Wird z. B. eine Variable vom Typ MR verwendet, werden die Attribute für alle Domain- und alle Range-Objekte aktualisiert.

7.4.3 Mapping-Operatoren

Als Vertreter der Mapping-Operatoren werden `traverse` und `map` näher vorgestellt. Beide dienen der Ausführung eines oder mehrerer traversierbarer Mappings. Während der Operator `traverse` ein Mapping traversiert, d. h. „lediglich“ die zu einer gegebenen Menge von Domain-Objekten korrespondierenden Range-Objekte ermittelt, erzeugt der Operator `map` Instanzkorrespondenzen zwischen Domain- und Range-Objekten. Als Beispiel werde für zwei DBLP-Publikationen das Mapping DBLP.PubAuth ausgeführt, dass für eine Publikation die korrespondierenden Autoren ermittelt.

Beispiel:
```
$DBLPAuth := traverse ($DBLPPubs, DBLP.PubAuth)
$DBLPPubAuth := map ($DBLPPubs, DBLP.PubAuth)
```

$DBLPPubs
Publication@DBLP
journals/vldb/RahmB01
conf/vldb/DoR02

$DBLPAuth
Author@DBLP
Erhard Rahm
Philip A. Bernstein
Hong-Hai Do

$DBLPPubAuth		
Publication@DBLP	Author@DBLP	Conf
journals/vldb/RahmB01	Erhard Rahm	1
journals/vldb/RahmB01	Philip A. Bernstein	1
conf/vldb/DoR02	Hong-Hai Do	1
conf/vldb/DoR02	Erhard Rahm	1

`map` „extrahiert“ den Teil des Mappings (im Beispiel: DBLP.PubAuth), der die übergebenen Objekte als Domain-Objekte enthält, so dass das Ergebnis wieder ein Mapping (genauer: Mapping Result) ist. Die Konfidenzwerte entsprechen denen des Mappings. Sind nur die Objektinstanzen von Interesse, wird `traverse` verwendet, das i. Allg. zu „kleineren“ Ergebnisse führt. Im Beispiel stehen den vier Korrespondenzen als Ergebnis von `map` drei Instanzen als `traverse`-Ergebnis gegenüber, da gleiche Objektinstanzen (im Beispiel: Erhard Rahm) nur einmal auftreten.

Die Ausführung von `traverse` und `map` erfordert, dass die übergebenen Objektinstanzen (im Beispiel: `$DBLPPubs`) vom gleichen Typ sind wie die Domain-Objekte des auszuführenden Mappings. Das Ergebnis hat den Typ der Range-Objekte (`traverse`) bzw. den gleichen Typ wie das Mapping (`map`).

Um die Verwendung der Operatoren einfacher zu gestalten, sind `traverse` und `map` – wie auch alle anderen Operatoren – polymorph, d. h. sie können mit Eingabeparametern unterschiedlichen Typs aufgerufen werden. Dies soll am Beispiel des Operators `traverse` illustriert werden, für den Abbildung 7.12 die möglichen unterschiedlichen Ausführungsformen zeigt. Zunächst kann `traverse` für ein Mapping ausgeführt werden, so dass das entsprechende im Source-Mapping-Modell registrierte Mapping verwendet wird. Alternativ kann auch ein bestehendes MR oder AMR verwendet werden, das z. B. durch eine vorherige Ausführung von `map` entstanden ist. Dabei werden die korrespondierenden Objekte anhand der im MR bzw. AMR vorliegenden Korrespondenzen bestimmt.

Nicht in der Abbildung dargestellt ist die Möglichkeit, mehrere Mappings und MR in

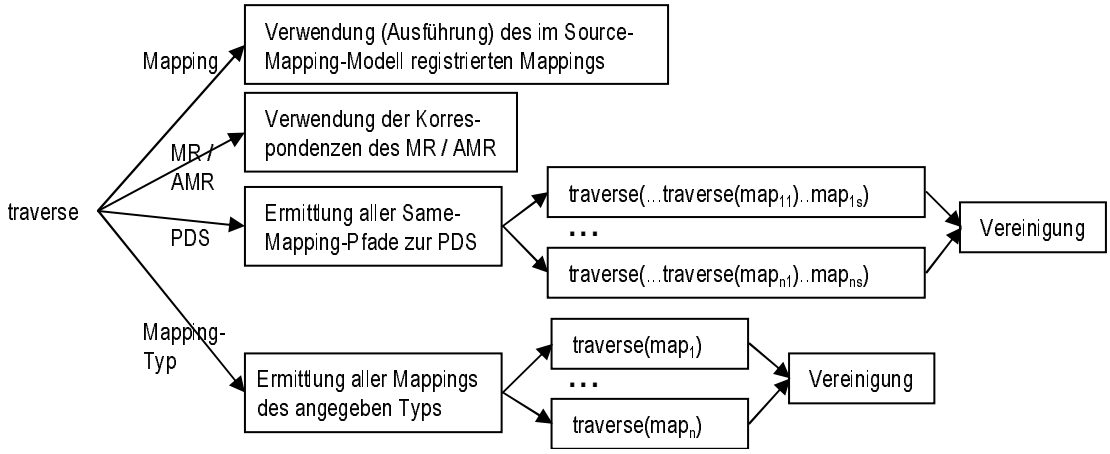


Abbildung 7.12: Schematische Darstellung zur Verwendung des Operators `traverse`

einer angegebenen Reihenfolge zu traversieren, was einer Hintereinanderausführung von `traverse` mit jeweils einem Mapping oder MR entspricht. Neben der Angabe konkreter Mappings kann der Nutzer auch eine PDS angeben, um die gleichen Objektinstanzen in einer anderen Datenquelle zu ermitteln. Dabei werden alle (zyklenfreien) Same-Mapping-Pfade von der LDS der Eingabeobjekte zur Zieldatenquelle bestimmt. Die Angabe der PDS ist dabei ausreichend, da Eingabeobjekte und Zieldatenquelle denselben Objekttyp besitzen, da nur Same-Mappings betrachtet werden. Anschließend wird jeder Pfad schrittweise traversiert, d. h. auf das `traverse`-Ergebnis des ersten Mappings wird `traverse` mit dem zweiten Mapping des Pfades angewendet usw. Abschließend werden die auf verschiedenen Pfaden erhaltenen Objektinstanzen vereinigt. Als letzte Möglichkeit kann – bei aggregierten Objekten – statt eines Mappings auch ein Mappingtyp angegeben werden. Dabei werden alle Mappings dieses Typs traversiert und die erhaltenen Objektinstanzen vereinigt.

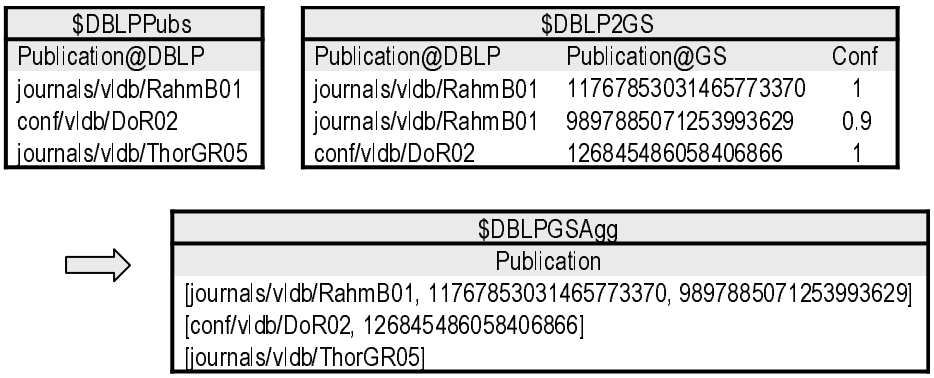
In der Klasse der Mapping-Operatoren gibt es neben `traverse` und `map` noch `match` und `attrMatch`, die im Anhang A detailliert vorgestellt werden. Während der Operator `map` zur Ausführung traversierbarer Mappings verwendet wird, dient `match` analog zur Ausführung von Match-Mappings. Die Erstellung eines Mappings mittels Attributvergleich ermöglicht der Operator `attrMatch`, wobei die Attributwerte auf Basis verschiedener Ähnlichkeitsfunktionen miteinander verglichen werden können (siehe Anhang A). Die Operatoren `match` und `attrMatch` spielen insbesondere für das Object Matching eine wichtige Rolle (siehe Kapitel 9).

7.4.4 Aggregations-Operatoren

Aggregation ermöglicht das Zusammenfassen mehrerer als gleich erkannter Objektinstanzen zu einem aggregierten Objekt, so dass diese bei der weiteren Verarbeitung als logische Einheit verwendet werden können. Die Aggregation wird durch den Ope-

rator `aggregate`, die Umkehrung, d. h. die Deaggregation, durch `disAgg` realisiert. Die Information, welche Objektinstanzen als gleich anzusehen sind und somit zu einem aggregierten Objekt zusammengefasst werden sollen, wird durch ein Same-Mapping repräsentiert. Das Vorgehen soll durch folgendes Beispiel illustriert werden.

Beispiel: `$DBLPGSAgg := aggregate ($DBLPPubs, $DBLP2GS)`



Ausgangspunkt sind drei DBLP-Publikationen (`$DBLPPubs`) und ein Same-Mapping zwischen DBLP- und GS-Publikationen (`$DBLP2GS`). Das Same-Mapping kann z. B. durch Anwendung eines Mapping-Operators erzeugt werden, wobei die Konfidenz die Ähnlichkeit der korrespondierenden Objekte angibt. Im Beispiel hat eine DBLP-Publikation Korrespondenzen zu zwei GS-Publikationen, wobei eine Korrespondenz eine Konfidenz von 0.9 habe, z. B. weil durch einen Tipp- oder Extraktionsfehler der GS-Titel der Publikation verfälscht ist. Das Ergebnis der Aggregation enthält drei aggregierte Objekte, weil als Ausgangspunkt drei Objektinstanzen übergeben wurde. Jedes aggregierte Objekt besteht aus der DBLP-Publikation sowie aus den im übergebenen Same-Mapping korrespondierenden GS-Publikationen. Auch wenn (wie im Beispiel die dritte DBLP-Publikation) eine Objektinstanz keine Korrespondenz im Same-Mapping besitzt, erscheint sie im Ergebnis als aggregiertes Objekt.

Die aggregierten Objekte lassen sich nun in verschiedener Art und Weise weiterverarbeiten. Zum einen lassen sich mittels Attribut-Operatoren fusionierte Attribute bilden. Abbildung 7.11 (Seite 140) zeigt das erste aggregierte Objekt des Beispiels mit einem zusätzlichen fusionierten Attribut „NoOfCitations“, das sich als Summe der Zitierungszahlen der zugehörigen GS-Publikationen ergibt. Neben der Fusionierung der Attribute ermöglicht die Aggregation auch eine gemeinsame Mapping-Verarbeitung. So kann – wie in Abbildung 7.12 (Seite 148) illustriert – auf die aggregierten Publikationen der Operator `traverse` mit dem Mapping-Typ PubAuth angewendet werden. Dabei werden automatisch alle „passenden“ Mappings im Source-Mapping-Modell identifiziert, d. h. alle Mappings des übergebenen Typs, deren Domain-LDS mit der LDS mindestens einer Objektinstanz in den aggregierten Objekten übereinstimmt. Für das Beispiel wurden die Mappings DBLP.PubAuth sowie GS.PubAuth ausgewählt, so dass als Ergebnis sowohl DBLP- als auch GS-Autoren

entstehen, die anschließend selbst wieder mittels eines Same-Mappings, das z. B. mit Hilfe des Attribut-Matchers erstellt wurde, aggregiert werden können.

Das Zusammenspiel der verschiedenen Operatoren innerhalb eines Workflows zur Datenintegration wird in Kapitel 8 dargestellt, wobei insbesondere unterschiedliche Möglichkeiten zur Mapping-Erstellung sowie zur Informationsfusion aufgezeigt werden. Die vorgestellten Operatoren in diesem Kapitel bilden lediglich einen Ausschnitt der gesamten iFuice-Operatoren. Die gewählte Darstellung zielte auf eine, an je einem Beispiel illustrierte, allgemeinverständliche Erläuterung der Funktionsweise der Operatoren ab. Anhang A hält eine Beschreibung aller Operatoren bereit.

7.5 Architektur

Der iFuice-Ansatz operiert mit Datenquellen, die durch Instanz-Mappings in einer Peer-to-Peer-artigen Weise miteinander verknüpft sind. Dennoch stellt die Architektur des iFuice-Ansatzes einen zentralen Mediator in den Mittelpunkt. Dabei greift der Mediator auf ein gegebenes Domänen- und Source-Mapping-Modell zu und ist somit in beliebigen Domänen anwendbar. Durch das Source-Mapping-Modell sind alle Quellen und Mappings registriert, so dass der Mediator durch die angegebenen Ausführungsparameter auf alle Quellen und Mappings zugreifen kann. Der Mediator nimmt die nutzerdefinierten Skripte entgegen und führt sie aus. Die Ergebnisse werden in Variablen gespeichert, deren Inhalte in einem generischen Warehouse abgelegt werden.

Im Folgenden wird die Architektur des Mediators dargestellt. Anschließend erfolgt die Beschreibung des generischen Warehouses, bevor kurz auf die prototypische Implementation eingegangen wird.

7.5.1 Mediator

Die dreischichtige Architektur des Mediators ist in Abbildung 7.13 dargestellt. Die oberste Schicht nimmt das auszuführende Skript entgegen und wandelt es mit einem Compiler nach einer Syntaxprüfung in eine interne Ausführungsvorschrift um. Während der Skriptausführung werden die im Skript eingesetzten Operatoren bzw. iFuice-Prozeduren aufgerufen. Letztere sind in einer Bibliothek zusammengefasst und werden selbst als iFuice-Skripte definiert. Für den Aufruf der Operatoren werden i. Allg. Variablen benötigt, deren Inhalt in einem generischen Warehouse gespeichert wird (siehe Abschnitt 7.5.2). Die von den Operatoren und Prozeduren zurückgelieferten Ergebnisse werden anschließend Variablen zugeordnet und im Warehouse gespeichert.

Bei der Ausführung der Operatoren greift der Mediator auf die entsprechenden Operatorimplementationen zurück. Dabei kann eine Operatorausführung (z. B. `map`) eine

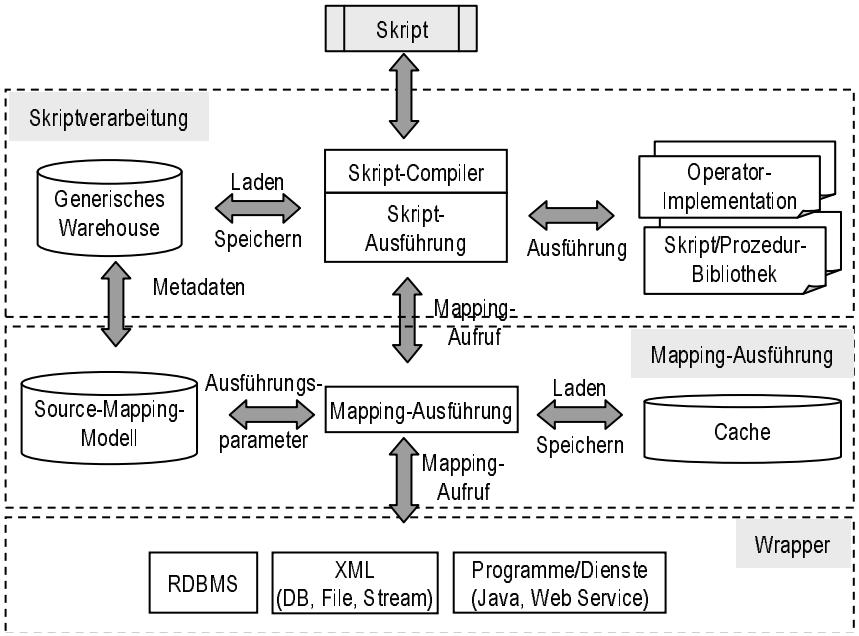


Abbildung 7.13: Schematische Darstellung des iFuice-Mediators

oder mehrere Mapping-Ausführungen nach sich ziehen. Auf der anderen Seiten können z. B. Kombinations-Operatoren (z. B. `union`) gänzlich ohne weiteren Zugriff auf Mappings ausgeführt werden. Benötigte Mapping-Aufrufe werden an die Mapping-Ausführungskomponente weitergereicht. Diese hat Zugriff auf das Source-Mapping-Modell und „kennt“ somit alle verfügbaren Mappings inklusive ihrer Ausführungsparameter. Um unnötige Mapping-Aufrufe zu vermeiden, werden die Ergebnisse jedes Mapping-Aufrufs in einem Cache abgelegt. Zur schnelleren Ausführung kann ein Mapping dann z. B. lediglich nur noch für den Teil der Daten ausgeführt werden, für den noch keine Ergebnisse im Cache gespeichert sind. Es ist aber auch möglich, die im Cache gespeicherten Daten zu ignorieren und ein bereits ausgeführtes Mapping später erneut auszuführen, um z. B. die Daten zu aktualisieren.

Zur Ausführung der Mappings wird ein entsprechender Wrapper verwendet, um verschiedene Implementierungsarten einheitlich im Mediator verwenden zu können. Die unterste Schicht der Architektur bildet daher eine erweiterbare Menge der Wrapper. Dabei gibt es für jeden Mapping-Typ (Query- und ID-Mapping) und für jede Implementationsart (z. B. relationale Datenbank, Java-Programm, usw.) je einen Wrapper, welcher ein Mapping des entsprechenden Typs ausführen kann und die Ergebnisse in einem definierten Format zurückliefert. Analog wird mit Wrappern auf die von den Quellen angebotene GetInstance-Funktion zugegriffen, um die aktuellen Attribute der Objektinstanzen zu akquirieren.

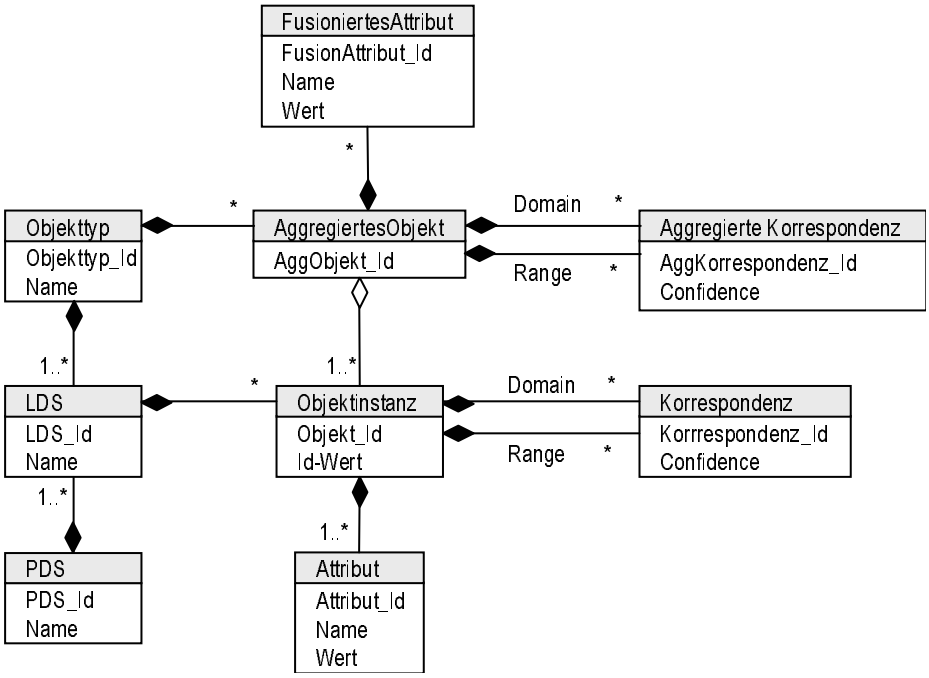


Abbildung 7.14: Aufbau des generischen iFuice-Warehouse

7.5.2 Generisches Warehouse

Sämtliche Daten des Mediators, d. h. insbesondere die von den Mappings ermittelten Objektinstanzen und Korrespondenzen, werden in einem generischen Warehouse gespeichert. Abbildung 7.14 zeigt das generische Datenmodell mittels UML. Objektinstanzen besitzen einen ID-Wert, der jedoch nur innerhalb ihrer LDS eindeutig ist. Daher werden sie mit einer global eindeutigen ID identifiziert. Die zugehörigen Attribute werden generisch als Name-Wert-Paar gespeichert. Zusätzlich verweist jede Objektinstanz auf die zugehörige LDS, die wiederum Objekttyp und PDS festlegt. Jede Korrespondenz verweist auf ein Domain- und ein Range-Objekt und wird zusätzlich mit einer Konfidenz annotiert. Des Weiteren können Objektinstanzen des gleichen Objekttyps zu aggregierten Objekten zusammengefasst werden, die ihrerseits wieder durch Korrespondenzen miteinander verbunden sein können. Aggregierte Objekte können – neben den über die Objektinstanzen verbundenen Attributen – zusätzlich fusionierte Attribute besitzen.

Zusätzlich verwaltet der Mediator Variablen, die jeweils durch einen Namen eindeutig gekennzeichnet werden und einen der Datentypen O, MR, AO oder AMR besitzen. Je nach Datentyp ist der Variable eine – möglicherweise leere – Menge von Objektinstanzen (O), Korrespondenzen (MR), aggregierten Objekten (AO) oder aggregierten Korrespondenzen (AMR) zugeordnet.

7.5.3 Implementation

Der iFuice-Ansatz wurde – unter Mithilfe von Herrn Toralf Kirsten – prototypisch in Java implementiert. Als Datenbank kommt mySQL[43] zum Einsatz. Für die Skript-Verarbeitung wird ein Compiler durch eine Grammatik des JavaCC-Tools[44] generiert. Der Mediator stellt mehrere Wrapper für die Mapping-Ausführung zur Verfügung. Gegenwärtig werden folgende Arten der Implementation unterstützt:

RDBMS: Die Ausführung wird durch das Absetzen einer SQL-Anfrage an eine relationale Datenbank realisiert.

Text-File: Eine Textdatei im CSV-Dateiformat[45], d. h. strukturierte Daten werden zeilenweise mit durch Trennzeichen separierten Feldern dargestellt, wird als relationale Tabelle aufgefasst und mittels SQL-Anfragen verarbeitet.

XML-DB: Eine innerhalb einer Datenbank abgelegte XML-Struktur wird mit einer XQuery angefragt.

XML-Stream: Ein XML-Dokument wird als Stream mittels XQuery verarbeitet.

Java: Die Ausführung wird durch eine Java-Klasse realisiert, welche mittels eines definierten Interfaces aufgerufen wird. Innerhalb der Java-Klasse kann u. a. ein Web-Service aufgerufen werden.

Die einzelnen Implementationen der Wrapper werden innerhalb des Source-Mapping-Modells als XML-Dateien beschrieben. Mit den dort definierten Metadaten (z. B. Datenbankverbindung oder aufzurufende Java-Klasse) ist der Mediator in der Lage, zur Laufzeit verschiedene Implementationen auszuführen.

7.6 Zusammenfassung

Der neuartige Datenintegrationsansatz iFuice wurde in diesem Kapitel vorgestellt, wobei vor allem die Besonderheiten des iFuice-Ansatzes hervorgehoben wurden. Durch instanzbasierte Mappings wird eine flexible Verknüpfung von Datenquellen ermöglicht, deren Semantik in einem Domänen- bzw. Source-Mapping-Modell reflektiert wird. Nutzer können mit Hilfe von Operatoren komplexe Datenintegrations-Workflows als iFuice-Skripte definieren und ausführen. Um die Einsatzfähigkeit von iFuice zu demonstrieren, wird mit der Zitierungsanalyse wissenschaftlicher Publikation ein Anwendungsfall im nächsten Kapitel näher vorgestellt.

[43] http://www.mysql.com/
[44] https://javacc.dev.java.net/
[45] http://de.wikipedia.org/wiki/CSV-Datei

8

Anwendungsfall: Zitierungsanalyse

In diesem Kapitel wird die Nutzung des iFuice-Ansatzes zur Datenintegration an einem Beispielszenario vorgestellt. Nach einer kurzen Einführung in die Domäne der Zitierungsanalyse (Abschnitt 8.1) erfolgt die Vorstellung eines komplexen Workflows, mit dessen Hilfe Daten für eine Zitierungsanalyse integriert werden. Dabei wird der Schwerpunkt auf die Erstellung relevanter Mappings (Abschnitt 8.2) sowie deren Verwendung zur Informationsfusion (Abschnitt 8.3) gelegt. Abschließend erfolgt eine kurze Darstellung der Durchführung und der Ergebnisse einer konkreten Zitierungsanalyse (Abschnitt 8.4) sowie eine Präsentation zweier auf iFuice-basierender Applikationen zur Zitierungsanalyse (Abschnitt 8.5).

8.1 Szenario: Zitierungsanalyse

Die Häufigkeit, mit der wissenschaftliche Publikationen zitiert bzw. referenziert werden, gibt einen Hinweis auf den Einfluss der wissenschaftlichen Arbeit. Neben einer einfachen Liste der am häufigsten zitierten Publikationen erlauben Zitierungszahlen weitere Analysen bzgl. verschiedener Kriterien, z.B. Autoren, Institutionen (z.B. an welcher Universität die Arbeit entstanden ist) oder Venues. Beispiele für Online-Zitierungsdatenbanken mit solchen Analysefunktionen sind *Scopus*[46] und *ISI Web of Science*[47].

[46] http://www.scopus.com
[47] http://portal.isiknowledge.com/

8.1.1 Metriken

Die DBLP Bibliography [2] veröffentlicht eine einfache Liste der am häufigsten referenzierten Papiere[48] aus dem Datenbankbereich. Die meisten Zitierungen (Stand: Oktober 2007) erhalten demnach Klassiker aus den 70ern und 80ern, wie z. B. der bei TODS erschienene Artikel von Chen über das Entity-Relationship-Modell, welcher die meisten Zitierungen erhält. Solche einfachen Listen, die als Metrik die absolute Anzahl der Zitierungen verwenden, bevorzugen ältere Papiere, so dass aktuelle Forschungsergebnisse nur ungenügend betrachtet werden.

Eine einfache Art des Vergleichs von Venues ist die Bestimmung der durchschnittlichen Zitierungszahl pro Publikation eines Venues. Citeseer erstellt z. B. solch eine nach der durchschnittlichen Zitierungszahl sortierte Liste[49]. Da auch hier ältere Arbeiten auf Grund der längeren Zeitspanne für mögliche Zitierungen bevorzugt werden, können weitere Maße, wie z. B. der *Impact Factor* [6], abgeleitet werden. Dieser Wert gibt z. B. für ein Venue an, wie häufig die in den Jahren $X-1$ und $X-2$ erschienenen Artikel von Publikationen aus dem Jahre X zitiert worden sind. Dadurch lassen sich auch vergleichende Analysen über verschiedene Jahrgänge hinweg realisieren, um z. B. den zeitlichen Einfluss der Venues zu charakterisieren – u.a. im Vergleich zur zeitlichen Veränderung der Akzeptanzraten und Einreichungszahlen [15].

Auch um die wissenschaftliche Leistung von Autoren anhand der Zitierungszahlen ihrer Publikationen abzuschätzen, versagen einfache statistische Maße. Die Gesamtsumme der Zitierungen bevorzugt fleißige Autoren mit vielen Publikationen, auch wenn die Zitierungszahl pro Publikation gering ist. Die durchschnittliche Anzahl der Zitierungen pro Publikation ist insbesondere für Autoren mit wenig Publikationen statistisch nicht aussagekräftig und benachteiligt junge Autoren gegenüber älteren. Daher wurde in [91] der h-Index vorgeschlagen. Dabei hat ein Autor mit n Publikationen einen h-Index h, wenn h seiner Publikationen jeweils mindestens h Zitierungen und die verbleibenden $n-h$ Publikationen höchstens h Zitierungen besitzen. Dadurch charakterisiert der h-Index eine Balance zwischen dem „Fleiß“ eines Autors (d. h. der Anzahl seiner Publikationen) und der „Qualität“ seiner Publikationen (d. h. der Anzahl der Zitierungen pro Publikation). Der h-Index wird durch neu hinzukommende, nicht (oder nur wenig) zitierte Publikationen nicht beeinflusst. Auf der anderen Seite bewirkt eine weitere Erhöhung der Zitierungszahl der besten Publikationen ebenfalls keine Änderung des h-Index. Um dem letzten Punkt Rechnung zu tragen und insbesondere außergewöhnlich viel zitierte Publikationen stärker in die Berechnung eines Index eingehen zu lassen, wurde der g-Index [55] vorgeschlagen. Für eine Liste von Publikationen ist der g-Index die größte Zahl g, so dass die Summe der Zitierungen der meistzitierten g Publikationen größer oder gleich g^2 ist.

[48] http://www.informatik.uni-trier.de/~ley/db/about/top.html
[49] http://citeseer.ist.psu.edu/impact.html

Platz	Publikation	#Zitierungen
1	iFuice - Information Fusion utilizing Instance Correspondences and Peer Mappings [158]	5
2	AWESOME - A Data Warehouse-based System for Adaptive Website Recommendations [191]	4
3	Citation Analysis of Database Publications [157]	2
4	Adaptive Website Recommendations with AWESOME [189]	0

Tabelle 8.1: Auszug aus Publikationsliste mit Zitierungszahlen (Stand: Juni 2007)

Tabelle 8.1 illustriert die Berechnung der beiden Index-Werte für eine Liste von vier Publikationen. Der h-Index ist 2, da die ersten beiden Publikationen mindestens zwei Zitierungen und die verbleibenden Publikationen nicht mehr als zwei Zitierungen besitzen. Der g-Index ist 3, da die summierte Zitierungszahl der ersten drei Publikationen ($5 + 4 + 2 = 11$) größer ist als 9 ($=3^2$). Eine mögliche Veränderung der Zitierungszahl der ersten beiden Publikationen hat keinen Einfluss auf den h-Index. Eine Erhöhung des h-Index auf 3 ist nur möglich, wenn mindestens eine der beiden anderen Publikationen eine Zitierungszahl von 3 erreicht. Demgegenüber würden z. B. fünf zusätzliche Zitierungen der iFuice-Publikation (auf dann insgesamt 10 Zitierungen) zu einer Erhöhung des g-Index auf 4 führen, da dann die summierte Zitierungszahl der ersten vier Publikationen $10 + 4 + 2 + 0$ den Wert 16 ($=4^2$) erreichen würde.

8.1.2 Datenquellen

Voraussetzung für aussagekräftige Zitierungsanalysen unter Verwendung der geschilderten Metriken ist eine hohe Datenqualität, da ansonsten die abgeleiteten Berechnungen nicht relevant sind („garbage in, garbage out“). Für eine Zitierungsanalyse sind insbesondere die beiden folgenden Aspekte entscheidend.

Hohe Qualität der bibliografischen Daten: Publikationen werden innerhalb anderer Publikationen zitiert und dabei treten neben Tippfehlern auch unterschiedliche Formatierungen auf. Um die Zitierungszahl einer Publikation korrekt zu erfassen, müssen daher die unterschiedlichen Referenzen der gleichen Publikation exakt einander zugeordnet werden. Weiterhin setzt eine Zitierungsanalyse bzgl. Autoren oder Venues für jede Publikation die korrekten bibliografischen Daten voraus, d. h. welche Autoren an der Publikation beteiligt waren und wo sie erschienen ist.

Vollständige Erfassung der Referenzen: Das Beispiel der oben genannten Citeseer-Liste verdeutlicht den negativen Einfluss fehlender Zitierungen. Wenn nicht für alle Publikationen die notwendigen Zitierungen erfasst sind, können große Konferenzen (z. B. SIGMOD oder VLDB) mit deutlich mehr Publikationen als kleinere Workshops (z. B. WebDB) bei der durchschnittlichen Zitierungszahl pro Publikation schlechter abschneiden. Als Folge erscheint z. B. der im Rahmen der SIGMOD-Konferenz stattfindende WebDB-Workshop auf Platz 35 während SIGMOD und VLDB auf den Plätzen 66 und 106 rangieren (Stand: Oktober 2007).

Auf Grund der obigen Analyse werden für die in diesem Kapitel exemplarisch durchgeführte Zitierungsanalyse die folgenden vier Datenquellen ausgewählt:

DBLP Bibliography (DBLP) [2] ist eine manuell gepflegte bibliografische Website, welche komplette Publikationslisten vieler Konferenzen, Journals und Workshops im Informatik-Bereich auflistet.[50] Für jeden Artikel werden Titel, Jahr, Autoren und das zugehörige Venue in sehr guter Qualität erfasst. Es stehen jedoch keine Zitierungszahlen zur Verfügung. (Für einige Publikationen wurden mittlerweile Zitierungen ergänzt, die aber nicht für eine Zitierungsanalyse ausreichen.) Sämtliche Daten können innerhalb einer Datei heruntergeladen und für eine Zitierungsanalyse z. B. in einer relationalen Datenbank gespeichert werden.

ACM Digital Library (ACM) [1] umfasst u. a. die kompletten Publikationslisten von Konferenzen und Journals der ACM. Zusätzlich werden Zitierungen derjenigen Publikationen erfasst, die selbst bei der ACM erfasst sind. Für diese Zitierungsanalyse wurde ein Teil der ACM mit einem Crawler heruntergeladen, so dass alle Daten als HTML-Dateien zur Verfügung stehen.

Google Scholar (GS) [3] ist eine bibliografische Suchmaschine, welche lediglich Einträge zu Publikationen enthält und Informationen zu Autoren und zugehörigen Venues als Attribute den Publikationen zuordnet. Google Scholar ermittelt die Einträge durch das Crawling relevanter Websites und extrahiert Referenzen u. a. aus den zu Grunde liegenden PDF-Dateien. Dadurch erreicht GS eine riesige Datengrundlage zur Ermittlung von Zitierungen, was sich im Vergleich zu ACM in wesentlich größeren Zitierungszahlen niederschlägt. Durch die automatische Informationsextraktion sowie durch die i. Allg. schlechte Qualität von Referenzlisten (u. a. durch heterogene Formatierung oder Tippfehler) ergibt sich allerdings eine Vielzahl von Duplikaten. Gesuchte Informationen von Google Scholar können nur über Anfragen generiert werden. Uneinheitliche Repräsentationen, z. B. der Venue-Namen, führen weiterhin dazu, dass keine kompletten Publikationslisten von Venues oder Autoren generiert werden können.

[50] Mehr als 945.000 Publikationen (Stand: Oktober 2007)

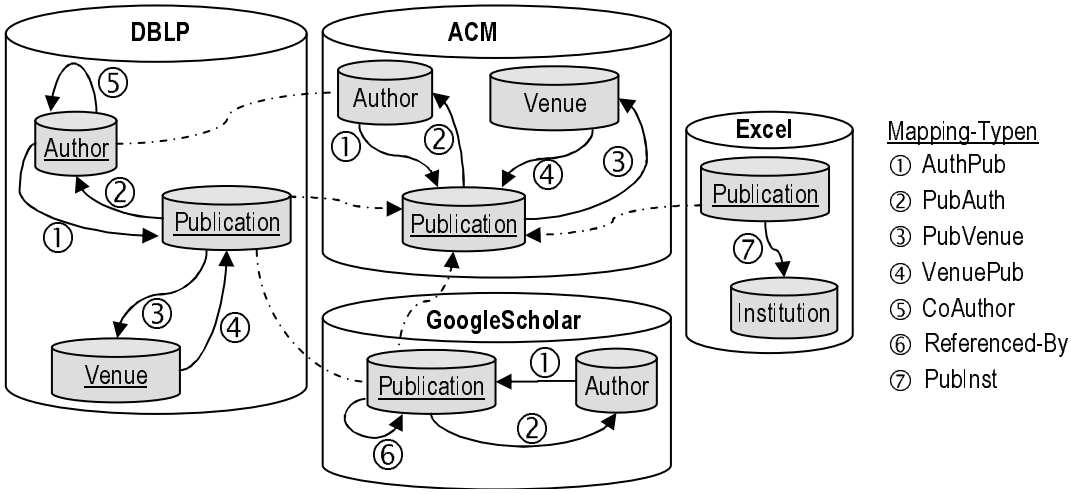


Abbildung 8.1: Source-Mapping-Modell für Zitierungsanalyse

Institution.csv ist ein Excel-File (abgespeichert als CSV-Datei), welches eine Liste von Publikationen mit der zugehörigen Institution enthält. Diese Liste wurde manuell erstellt, indem die zugehörigen Volltexte der Publikationen betrachtet wurden. Als Institution der Publikation wurde die Institution des Erstautors gewählt, d. h. seine Universität bzw. sein Forschungsinstitut. Da die Volltexte von ACM heruntergeladen wurden[51], wird jede Publikation durch ihre ID bei der ACM definiert.

Abbildung 8.1 zeigt das im Rahmen der Zitierungsanalyse verwendete – und bereits aus Kapitel 7 bekannte – Source-Mapping-Modell. Durch iFuice-Skripte werden die Daten für eine Zitierungsanalyse integriert und entsprechend aufbereitet. Die Generierung von Publikations-Same-Mappings zwischen den Datenquellen dient zur Identifikation gleicher Instanzen (Abschnitt 8.2). Dabei wird im Wesentlichen das Prinzip der Mapping-Erstellung, z. B. durch Ausführung bestehender Mappings, dargestellt. (Die Bildung komplexer Workflows auf Basis von iFuice-Skripten zur Bestimmung hochqualitativer Same-Mappings wird gesondert in Kapitel 9 betrachtet.) Anschließend können dadurch einer *realen* Publikation alle Attribute der Quellen zugeordnet werden, z. B. Titel und Autoren (DBLP), Institution (Excel) und summierte Zitierungszahlen (GS und ACM) (Abschnitt 8.3). Um Selbstzitierungen nicht mit einzubeziehen, werden diese erkannt und gesondert behandelt.

Zur Darstellung der Zitierungsanalyse wird in den beiden folgenden Abschnitten jeweils ein kurzes Skript vorgestellt, dessen Ausführung beispielhaft mit weiteren Abbildungen[52] illustriert wird. Dabei referenzieren die Zahlen an den Pfeilen die jeweilige Zeilennummer des Skripts.

[51] Auf Ausnahmen wird in Abschnitt 8.4 eingegangen.
[52] Für Hinweise zur Notation siehe Abschnitt A.2.1.

```
 1: $DBLPVenue:= queryInstances(Venue@DBLP,
       "[year] >= 1994 and [year] <= 2003");
 2: $DBLPVenue:= queryInstances($DBLPVenue, "[series] in
       ('VLDB', 'SIGMOD', 'VLDB J.', 'SIGMOD Record', 'TODS')");
 3: $DBLP:= traverse($DBLPVenue, DBLP.VenuePub);

 4: $GS:= queryInstances(Publication@GS, "intitle:[[title]]", $DBLP);
 5: $DBLP2GS:= match($DBLP, $GS, GSMatcher);

 6: $DBLP2ACM:= map($DBLP, DBLP2ACM);
 7: $DBLP2ACM:= queryMap($DBLP2ACM, "[_confidence]>0.8");

 8: $Excel:= queryInstances (Publication@Excel, "1=1");
 9: $Excel2ACM:= map ($Excel, Excel2ACM);
10: $DBLP2Excel:= compose($DBLP2ACM, inverse($Excel2ACM));
```

Abbildung 8.2: iFuice-Skript zur Mapping-Erstellung bei Zitierungsanalyse

8.2 Mapping-Erstellung

Mit den ersten drei Zeilen des in Abbildung 8.2 dargestellten Skripts werden die relevanten Publikationen bei DBLP ermittelt (siehe Abbildung 8.3). Dazu wird zunächst eine Query an die Datenquelle gesendet (Zeile 1), welche alle Venues im betrachteten Zeitraum von 1994-2003 zurückliefert. Die erhaltene Menge der Objektinstanzen wird an die Variable `$DBLPVenue` gebunden. Anschließend wird nun mit demselben Operator eine Anfrage auf der Variable `$DBLPVenue` durchgeführt (Zeile 2), welche nun als „virtuelle Datenquelle“ (vom Typ Venue@DBLP) fungiert. Dadurch wird die Menge der Venues weiter eingeschränkt. Abschließend wird das Mapping `DBLP.VenuePub` traversiert (Zeile 3), d. h. dass für jedes Venue in `$DBLPVenue` die im Mapping assoziierten Publikationen ermittelt werden und die Gesamtmenge der Publikationen zurückgeliefert wird.

Nachdem nun alle DBLP-Publikationen zur Verfügung stehen, müssen die Same-Mappings zu den anderen Datenquellen etabliert werden. Das Skript verdeutlicht dabei verschiedene Techniken. Da Google Scholar seine Einträge automatisch durch Webextraktionstechniken generiert, enthält es viele Duplikate und falsche bzw. fehlende Werte. Daher steht ein Match-Mapping zur Verfügung, das die Besonderheiten von Google Scholar innerhalb eines Object-Match-Prozesses beachtet. Vor der Anwendung des Match-Mappings (Zeile 5) werden in Zeile 4 alle relevanten Google-Scholar-Publikationen bestimmt (siehe Abbildung 8.4). Dies wird mittels Anfragen an Google Scholar realisiert. Für jede DBLP-Publikation erstellt `queryInstances` eine Query, deren Aufbau durch den Parameter `"intitle:[[title]]"` definiert

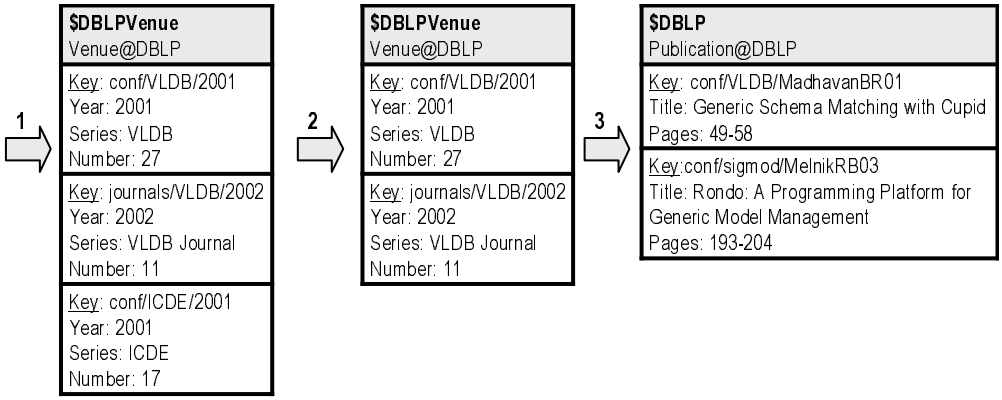


Abbildung 8.3: Ermittlung der relevanten DBLP-Publikationen

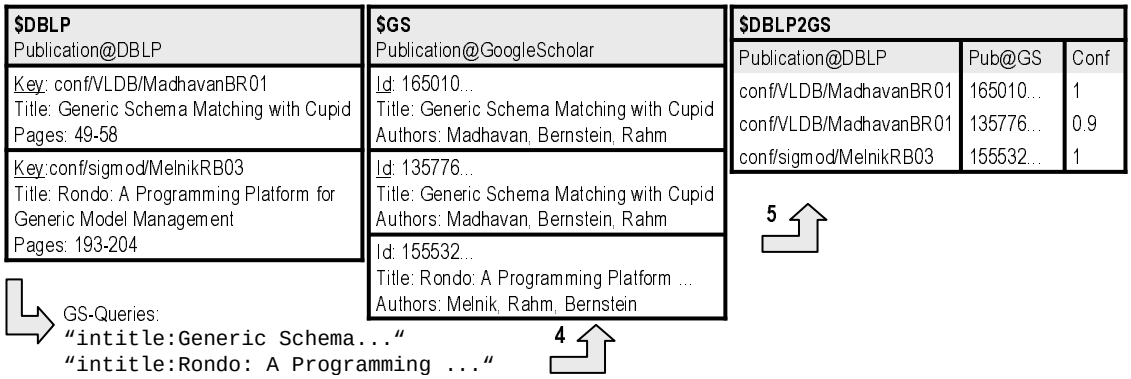


Abbildung 8.4: Same-Mapping-Bestimmung zwischen DBLP und Google Scholar

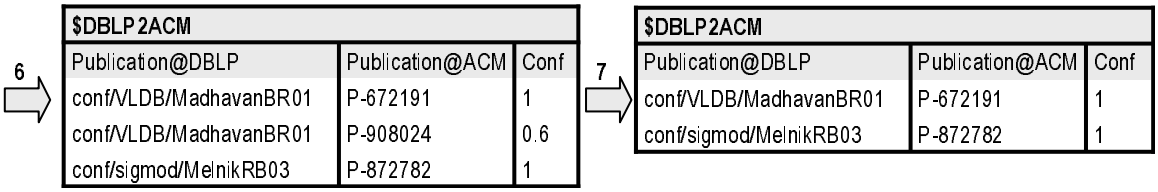


Abbildung 8.5: Bestimmung des Same-Mappings zwischen DBLP und ACM

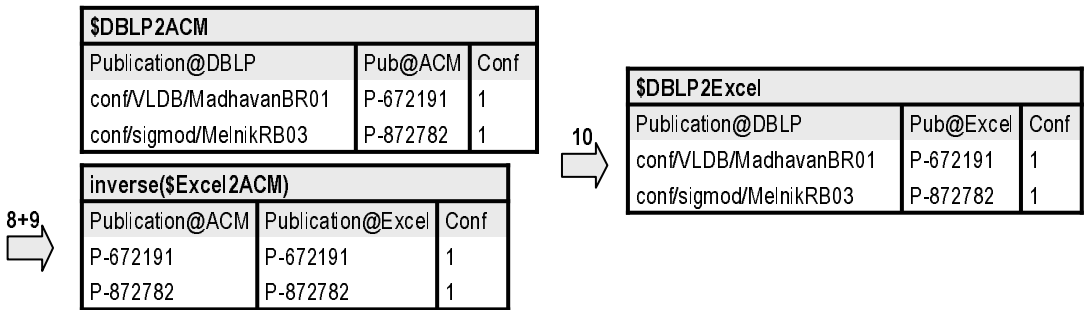


Abbildung 8.6: Bestimmung des Same-Mappings zwischen DBLP und Excel

ist. Dabei wird `[[title]]` für jede DBLP-Publikation durch den Wert des Attributs `title`, d. h. dem Publikationstitel, ersetzt. Die resultierenden Anfragen werden durch das entsprechende Query-Mapping ausgeführt. Die erhaltenen GS-Publikationen werden als Ergebnis zurückgeliefert.

Eine andere Art der Mapping-Erstellung ist die Ausführung eines traversierbaren Mappings (Zeile 6, Abbildung 8.5), wie es im verwendeten Source-Mapping-Modell zwischen den Datenquellen Publication@DBLP und Publication@ACM existiert. Das Mapping ermittelt für jede der übergebenen DBLP-Publikationen (z. B. durch eine entsprechende Suchanfrage bei der ACM-Website mit anschließendem Ähnlichkeitsvergleich von Titel und Publikationsjahr) alle (vermutlich) gleichen ACM-Publikationen. Dabei wird jeder Korrespondenz eine Konfidenz entsprechend der Ähnlichkeit zugeordnet, z. B. auf Basis der Ähnlichkeit von Titel und Publikationsjahr. Das traversierbare Mapping (definiert durch seinen Namen `DBLP2ACM`) wird mit dem Operator `map` für die DBLP-Publikationen `$DBLP` aufgerufen. Das resultierende Mapping Result wird an die Variable `$DBLP2ACM` gebunden. Aus Gründen der Datenqualität sollen jedoch nur Korrespondenzen des Mapping Results verwendet werden, deren Konfidenz über einem Schwellwert (im Beispiel 80%) liegen. Diese Selektion wird mit dem Operator `queryMap` durchgeführt. Er ist das entsprechende Gegenstück zu `queryInstances`, welches auf Objektinstanzen arbeitet.

Als letztes wird das Mapping zwischen DBLP und Excel bestimmt, was in zwei Schritten realisiert wird (siehe Abbildung 8.6). Zunächst wird – nachdem alle Excel-Publikationen durch `queryInstances` mit Hilfe einer immer geltenden Bedingung (z. B. 1=1) in der Variablen `$Excel` zur Verfügung stehen (Zeile 8) – das Mapping zwischen Excel und ACM durch Ausführung des traversierbaren Same-Mappings realisiert (Zeile 9). Anschließend werden mittels `compose` zwei Mapping Results verknüpft (Zeile 10). Das „komponierte“ Mapping ordnet dann jeder DBLP-Publikation die entsprechende Excel-Publikation zu. Dabei muss vorher das in `$Excel2ACM` gespeicherte Mapping invertiert werden, wobei bei jeder Korrespondenz einfach Domain- und Range-Objekte vertauscht werden. Die Invertierung ist nötig, um die Hintereinanderausführung der Mappings „DBLP-ACM-Excel“ zu gewährleisten.

Das Ergebnis des Skripts aus Abbildung 8.2 sind drei Same-Mappings, welche die betrachteten DBLP-Publikationen mit Google Scholar (`$DBLP2GS`), ACM (`$DBLP2ACM`) und Excel (`$DBLP2Excel`) verknüpfen. Diese Mappings bilden die Grundlage für die im folgenden Abschnitt durchgeführte Informationsfusion.

8.3 Informationsfusion

Nach der Mapping-Erstellung wird das in Abbildung 8.7 dargestellte iFuice-Skript zur Informationsfusion verwendet. Ziel ist es, die von Google Scholar erfassten Zitierungen den zugehörigen DBLP-Publikationen zuzuordnen. Dabei kann die für jede

```
11: $GSReferenced:= map(range($DBLP2GS), GS.ReferencedBy);
12: $GSReferencedSelf:= queryMap($GSReferenced, "[_confidence]>0.5");
13: setAttr($GSReferencedSelf, "Citation", "count([id])");

14: $Pubs:= aggregate(aggregate($DBLP, $DBLP2ACM), $DBLP2GS);
15: $Pubs:= aggregate($Pubs, $DBLP2Excel);

16: setAttr($Pubs, "GSCitation", "sum([GS.Citation])");
17: setAttr($Pubs, "ACMCitation", "sum([ACM.Citation])");

18: $PubVenue:= map($Pubs, PubVenue);
19: $PubVenue:= inverse(aggregate(inverse($PubVenue),$DBLP2ACMVenue));
```

Abbildung 8.7: iFuice-Skript zur Informationsfusion bei Zitierungsanalyse

GS-Publikation vorhandene Zitierungszahl von Google Scholar verwendet werden, die jedoch Selbstzitierungen enthält. Um Selbstzitierungen eliminieren zu können, wird das Assoziations-Mapping GS.ReferencedBy ausgeführt (Zeile 11), welches für jede GS-Publikation die referenzierenden Publikationen ermittelt. Aus Gründen der Übersicht zeigt Abbildung 8.8 nur einige der referenzierenden Publikationen (P1 bis P6), wobei P2 eine Selbstzitierung sei. Das Assoziations-Mapping liefert alle Zitierungen und „kennzeichnet“ in diesem Beispiel Selbstzitierungen durch einen geringeren Konfidenzwert (z.B. 0.5). Durch die Anwendung von `queryMap` können anschließend Selbstzitierungen entfernt werden (Zeile 12). Abschließend wird das resultierende Mapping Result genutzt, um die Anzahl der Zitierungen zu bestimmen. Dazu wird mittels `setAttr` jedem Domain-Objekt ein neues Attribut `Citation` zugewiesen, dessen Wert sich anhand einer Gruppierungsfunktion (hier: `count`) ableiten lässt, d. h. dass für jedes Domain-Objekt die Anzahl der assoziierten Objekte bestimmt wird (Zeile 13). Dies entspricht der `GROUP BY`-Semantik von SQL, wobei nach dem Domain-Objekt gruppiert wird und der gruppierte Wert als Attribut den Domain-Objekten hinzugefügt wird.

Im zweiten Schritt (Zeilen 14 und 15) werden die berechneten Same-Mappings zur Aggregation genutzt, d. h. per Same-Mapping korrespondierende Publikationen werden zu einem aggregierten Objekt zusammengefasst. Der entsprechende Operator `aggregate` erhält dazu die Objektmenge der DBLP-Publikationen und die Same-Mappings. Die resultierenden aggregierten Objekte werden in Abbildung 8.9 durch die IDs der entsprechenden Objektinstanzen dargestellt. Auch auf bereits aggregierte Objekte kann `aggregate` erneut angewendet werden, so dass nicht alle Same-Mappings gleichzeitig bei der Aggregation angegeben werden müssen.

Aggregierte Objekte repräsentieren eine Menge von (als gleich angesehenen) Objektinstanzen des gleichen Objekttyps. Aus den Attributen der Objektinstanzen lassen

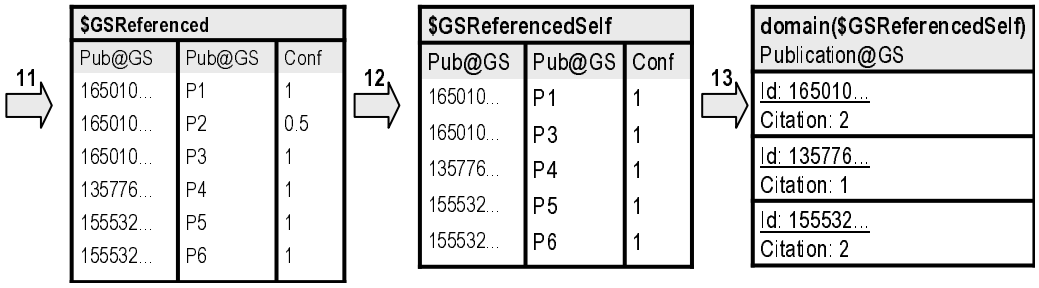


Abbildung 8.8: Ermittlung der GS-Zitierungszahl ohne Selbstzitierungen

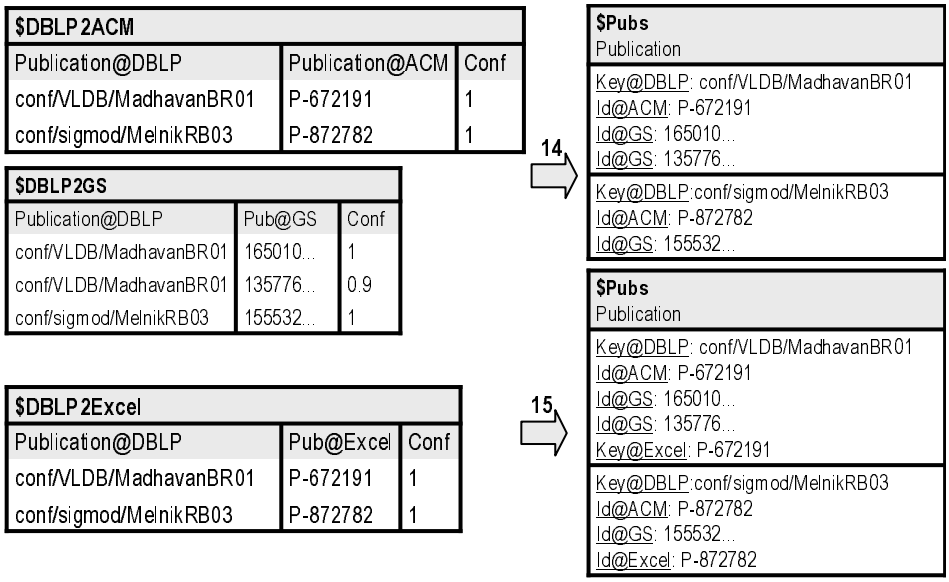


Abbildung 8.9: Aggregation gleicher Publikationen

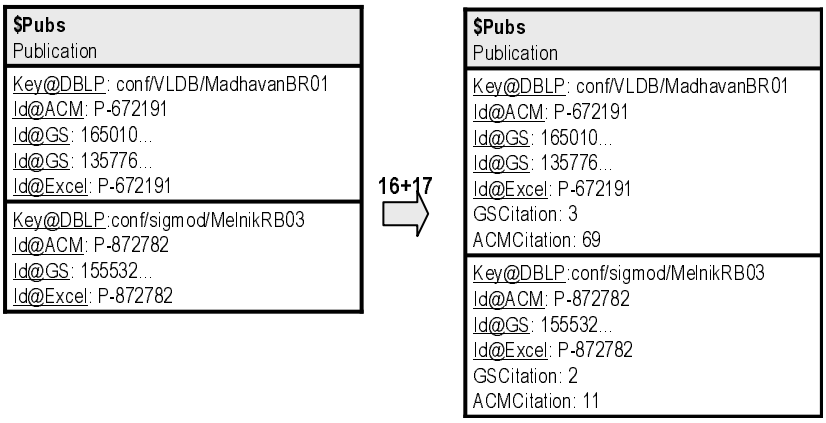


Abbildung 8.10: Bestimmung der Zitierungszahl als fusioniertes Attribut

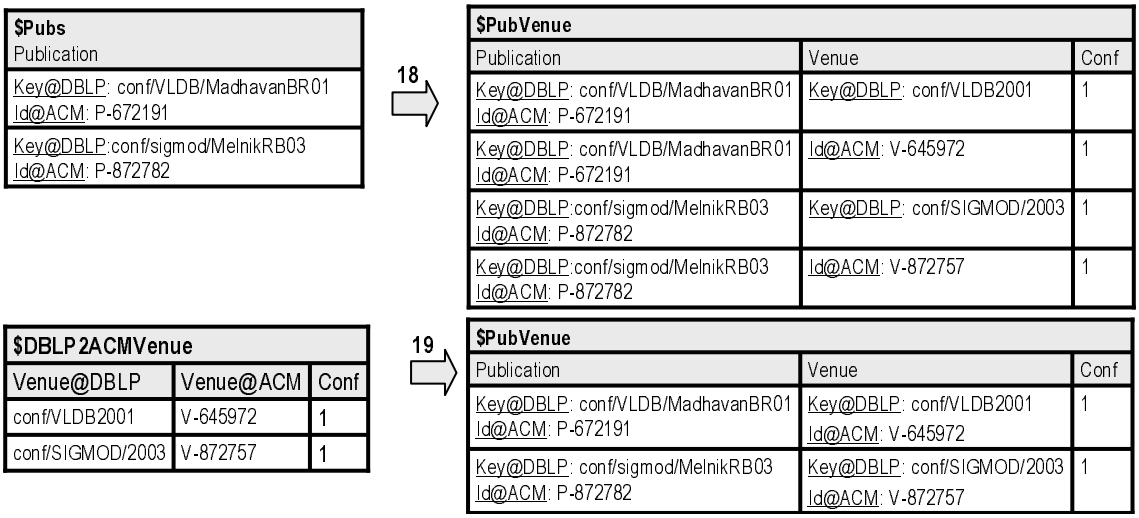


Abbildung 8.11: Bildung des aggregierten Mapping Results zwischen Publikationen und Venues

sich nun fusionierte Attribute ableiten, welche dem aggregierten Objekt zugewiesen werden. Das Beispielskript berechnet dabei in den Zeilen 16 und 17 die Attribute GSCitation und ACMCitation, welche sich jeweils aus der Summe der Zitierungen aller GS- bzw. ACM-Publikationen ergeben. Abbildung 8.10 illustriert die Berechnung. Da das Cupid-Papier zwei Google-Scholar-Einträge (165010... und 135776...) aufweist, werden deren Zitierungszahlen (2 bzw. 1) addiert und dem aggregierten Objekt zugewiesen.[53]

Analog zu Objektinstanzen können auch auf aggregierten Objekte Mappings angewendet werden. Da es sich um Objektinstanzen verschiedener Datenquellen handelt, können mehrere Mappings des gleichen semantischen Mapping-Typs gleichzeitig ausgeführt werden. Das Beispielskript zeigt in Zeile 18 für die aggregierten Objekte (Variable: `$Pubs`) die Ausführung aller Mappings des Typs `PubVenue`. Da es nur für DBLP und ACM entsprechende Mappings gibt, werden diese ausgeführt und das Ergebnis als aggregiertes Mapping Result dargestellt. Dabei werden die bei der Mapping-Ausführung entstehenden assoziierten Objekte in aggregierte Objekte umgewandelt, können aber zunächst nicht mit anderen Objekten zusammengefasst werden (Abbildung 8.11 oben). Der Grund ist die Tatsache, dass die Information, welche Objekte (im Beispiel: Venues) als gleich anzusehen sind, fehlt. Allein die Tatsache, dass zwei Objektinstanzen über ein Assoziations-Mapping mit dem gleichen Objekt verknüpft sind, lässt noch keine Aussage darüber zu, ob die beiden Instanzen als gleich anzusehen sind und somit zu einem aggregierten Objekt zusam-

[53] Die Annahme, dass es zwei GS-Einträge für das Cupid-Papier gibt, die jeweils eine bzw. zwei Zitierungen haben, wird hier zur Vereinfachung der Darstellung angenommen. Derzeit (Stand: Oktober 2007) weist Google Scholar mindestens sieben Einträge mit zusammen fast 600 Zitierungen auf.

mengefasst werden können. Im konkreten Beispiel mit den Venues würde das zwar funktionieren, weil eine Publikation nur jeweils einem Venue zugeordnet ist, aber z. B. für den Mapping-Typ PubAuth, der für Publikationen die zugehörigen Autoren bestimmt, würden (unterschiedliche) Autoren zu einem aggregierten Objekt zusammengefasst werden, nur weil sie an der gleichen Publikation beteiligt waren. Daher erfolgt die weitere Aggregation in einem separaten Schritt. Dies geschieht erneut über den `aggregate`-Operator (Zeile 19), welcher das aggregierte Mapping Result und das entsprechende Venue-Same-Mapping (`$DBLP2ACMVenue`, was im Vorfeld analog zu den Publikations-Same-Mappings bestimmt werden kann – siehe dazu Abschnitt 10.4) als Parameter enthält (Abbildung 8.11 unten). Da `aggregate` bei Mappings die Domain-Objekte aggregiert, muss `$PubVenue` vorher invertiert werden, damit die Venues aggregiert werden.

Das Ergebnis des gesamten Skripts ist in den Variablen `$Pubs` bzw. `$PubVenue` repräsentiert. Die gleichen „realen“ Publikationen der Datenquellen DBLP, ACM, GS und Excel werden durch jeweils ein aggregiertes Objekt zusammengefasst. Dadurch lassen sich abgeleitete Informationen bilden, die durch fusionierte Attribute (Gesamtzitierungszahl von ACM und GS) dargestellt sind. Analog sind als gleich erkannte Venues in aggregierten Objekten zusammengefasst. Die Verbindung zwischen Publikationen und Venues wird durch das *AMR* `$PubVenue` ausgedrückt, das für jede Publikation das zugehörige Venue assoziiert.

8.4 Durchführung einer Zitierungsanalyse

Die in den beiden vorangegangenen Abschnitten vorgestellten Skripte zeigen die grundlegende Methodik, mit der unter Verwendung von iFuice Zitierungsanalysen durchgeführt werden können. Die Auswahl der Datenquellen sowie die Realisierung der verfügbaren Mappings im Source-Mapping-Modell hängt vom konkreten Einsatzfall (z. B. Forschungsgebiet) ab. Das Ziel solcher Analysen ist eine vergleichende Evaluation zwischen Autoren, Publikationen, Venues und Institutionen, wobei insbesondere auch die zeitliche Entwicklung der Zitierungszahlen betrachtet werden kann. Dabei ist eine Fokussierung auf einen Forschungsbereich und die damit verbundenen relevanten Venues sinnvoll. Der Vergleich zwischen unterschiedlichen Forschungsbereichen ist i. Allg. nicht aussagekräftig, da für die Zitierungszahlen weitere Faktoren, z. B. die Anzahl der Forscher und Venues im jeweiligen Bereich, eine Rolle spielen.

Auf Basis der in den beiden vorangegangenen Abschnitten beschriebenen Vorgehensweise wurde im August 2005 eine umfangreiche Zitierungsanalyse durchgeführt [157][54]. Dabei wurden die im Datenbankbereich international führenden Konferenzen VLDB und SIGMOD sowie die Journals VLDB Journal, TODS und SIGMOD

[54] Für die deutsche Datenbanktagung BTW erschien Anfang 2007 ebenfalls eine auf iFuice basierende Analyse der Zitierungshäufigkeiten [105].

Record für den Zeitraum von 1994 bis 2003 betrachtet. Als Zitierungszahl wurde der Wert von Google Scholar ohne Selbstzitierungen verwendet. Grundlage der Analyse waren die in Abschnitt 8.2 und 8.3 vorgestellten Skripte. Um eine hohe Qualität der Analyse zu gewährleisten, wurden die folgenden Punkte zusätzlich durchgeführt.

- Das Same-Mapping zwischen DBLP und Google Scholar wurde manuell überprüft und gegebenenfalls entsprechend korrigiert, um eine hohe Datenqualität zu erreichen.

- Für die bei DBLP verfügbaren Autoren wurde eine Duplikaterkennung durchgeführt (siehe Abschnitt 10.4.2) und – analog zu Venues – gleiche Autoren zu einem aggregierten Objekt zusammengefasst.

- Auf Grund der Fülle der Publikationen – und dem damit verbundenen manuellen Aufwand – wurden nur für Publikationen mit mindestens 20 Zitierungen die Institutionen bestimmt. Dies entspricht 50% aller Papiere, die zusammen 91% aller Zitierungen auf sich vereinen.

- ACM enthält keine Einträge für die VLDB-Publikationen aus den Jahren 2002 und 2003. Daher mussten die Volltexte aus anderen Quellen (z. B. Homepages der Autoren) bezogen werden, um die Informationen zu den Institutionen zu erfassen. Dabei wurde jedem Volltext manuell die DBLP-ID zugeordnet, so dass das Same-Mapping zwischen DBLP und Excel direkt, d. h. ohne Verknüpfung mittels ACM, erstellt werden konnte.

- Es wurden alle Publikation manuell nach ihrem Typ kategorisiert, wobei u. a. zwischen Forschungs-, Demo- und Industriebeiträgen unterschieden wurde. Dies ermöglicht eine genauere Zitierungsanalyse, da z. B. Demo- und Industriebeiträge i. Allg. kürzer als Forschungsarbeiten sind und daher tendenziell weniger zitiert werden.

- Es wurde eine Aufteilung von Zitierungszahlen für Google-Scholar-Einträge vorgenommen, die offenbar mehrere DBLP-Publikationen zusammenfassten. Ein typisches Beispiel sind VLDB-Publikationen, die unter gleichem Namen ein Jahr später (in einer längeren Version) im VLDB Journal erscheinen (z. B. [36] und [37]). Während DBLP und ACM beide Versionen differenzieren, fasst Google Scholar sie in einigen Fällen zu einem Eintrag zusammen. Um eine möglichst gute Approximation der Zitierungszahl für die unterschiedlichen Publikationen in solchen Fällen zu erreichen, wurde die GS-Zitierungszahl gemäß dem Verhältnis der ACM-Zitierungszahlen aufgeteilt. Haben z. B. die beiden ACM-Publikationen jeweils 20 und 5 Zitierungen und der zugehörige (zusammengefasste) GS-Eintrag 100 Zitierungen, so werden den Publikationen jeweils 80 bzw. 20 GS-Zitierungen zugeordnet.

Konferenz/Journal	#Publikationen	#Zitierungen	Durchschnitt (#Zit./#Pub.)
SIGMOD	446	31.069	70
VLDB	570	28.659	50
SIGMOD Record	327	7.724	24
VLDB Journal	189	4.919	26
TODS	130	4.162	32
Overall	1.662	76.533	46

Tabelle 8.2: Übersicht der Anzahl der Forschungspublikationen und Zitierungen (Zitierungszahl von Google Scholar, Stand: August 2005)

Tabelle 8.2 gibt einen Überblick über die Anzahl der Forschungspapiere, d. h. ohne Demo- oder Industriebeiträge, und ihre Zitierungen für alle betrachteten Konferenzen und Journals. Es zeigt sich, dass die beiden Konferenzen deutlich mehr Publikationen hervorbringen und diese im Durchschnitt auch wesentlich öfter zitiert werden als Journal-Publikationen. Bei den Konferenzen dominiert SIGMOD gegenüber VLDB, wohingegen TODS das Journal mit den meisten Zitierungen pro Publikation ist. Eine nähere Untersuchung, wie sich die Zitierungen auf die Publikationen verteilen, zeigt ein starkes Ungleichgewicht. Sortiert man die Publikationen absteigend nach ihrer Zitierungszahl, erhalten die oberen 25% zusammen – je nach Konferenz bzw. Journal – zwischen 60% und 80%. Die jeweiligen Top-5-Papiere erhalten bei Konferenzen ca. 40% und bei Journals sogar ca. 70% aller Zitierungen. Für weitere Ergebnisse sei auf [157] verwiesen.

8.5 Applikationen

Im Rahmen des gewählten Anwendungsfalls zur Zitierungsanalyse wurden neben dem iFuice-Mediator zwei auf iFuice basierende Applikationen entwickelt. Mit Hilfe eines GUIs (Graphical User Interface) wurde die interaktive Erstellung von Skripten zur Zitierungsanalyse unterstützt. Mit dem Online Citation Service wurde eine zweite Applikation entwickelt, die über ein Webinterface einfache Zitierungsanalysen ermöglicht.

8.5.1 Generisches iFuice-GUI

Für die Durchführung der in [157] und [105] vorgestellten Zitierungsanalysen mussten die jeweiligen iFuice-Skripte erstellt werden. Dabei wurde das von Herrn Nick Golovin entwickelte GUI verwendet. Abbildung 8.12 zeigt einen entsprechenden Screenshot.

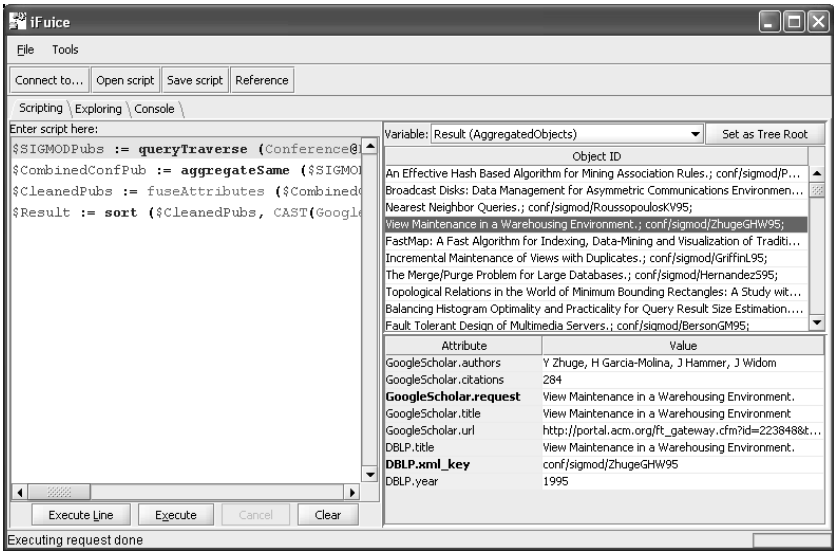


Abbildung 8.12: Screenshot des GUI für iFuice

Im linken Bereich des Hauptfensters werden die entsprechenden Skripte editiert und ausgeführt. Zur Unterstützung bei der Skripterstellung bietet das GUI ein Syntax-Highlighting sowie die Möglichkeit der schrittweisen Ausführung der Skripte. Zusätzlich können innerhalb der Skriptausführung Statusinformationen zu den Zwischenergebnissen (z. B. Anzahl der ermittelten Korrespondenzen) ausgegeben werden, um die Korrektheit der Skripte zu überprüfen.

Der rechte Bereich dient der Inspektion der Variablen, d. h. über eine Liste aller verfügbaren Variablen können die Inhalte einzelner Variablen – entsprechend ihres Typs – dargestellt werden, so dass Objekte mit ihren Attributen sowie Korrespondenzen betrachtet werden können. Dabei können insbesondere die Match-Ergebnisse manuell überprüft und gegebenenfalls angepasst werden. Die ermittelten Objektinstanzen und Mappings können zusätzlich durch definierte Austauschformate (z. B. XML, CSV) ex- und importiert werden und stehen somit auch für andere Programme (z. B. Microsoft Excel für die Generierung von Diagrammen) zur Verfügung.

8.5.2 Online Citation Service

Der Schwerpunkt der in [157] und [105] durchgeführten Analysen liegt im Vergleich von Autoren, Publikationen und Venues bzgl. der Zitierungszahlen sowie in der Analyse der zeitlichen Entwicklung der Zitierungen. Basis für solche detaillierten Untersuchungen ist eine relativ große Datengrundlage, z. B. 81680 Zitierungen bei Google Scholar für 2338 Publikationen in [157]. Ein wichtiger Aspekt der Zitierungsanalysen zur Sicherung einer hohen Datenqualität ist die manuelle Nachbearbeitung der Daten. Beispiele dafür sind die Einteilung der Publikationen in verschiedene Typen

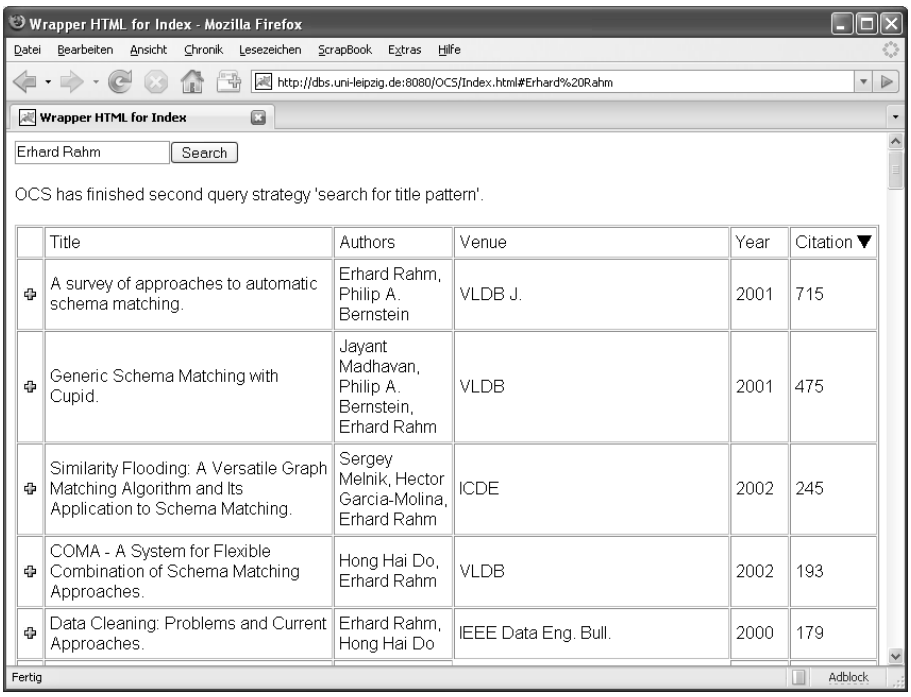


Abbildung 8.13: Screenshot der Web-Applikation zur Online-Zitierungsanalyse

(Forschung, Industrie, Demo, etc.) sowie die Überprüfung und ggf. Anpassung der Match-Ergebnisse.

Der geschilderte große Aufwand verdeutlicht, dass solche komplexen und hochqualitativen Analysen nicht online (d. h. in sehr kurzer Zeit) und automatisch durchgeführt werden können. Um dennoch einfache Analysen für kleinere Datenmengen, z. B. für die Publikationen eines Autors oder eines Venues, online durchführen zu können, wurde mit dem *Online Citation Service*[55] *(OCS)* eine auf iFuice basierende Webapplikation geschaffen [188]. Abbildung 8.13 zeigt einen Screenshot der Anwendung. Für einen Autor oder ein Venue wird die Liste der zugehörigen DBLP-Publikationen samt ihrer aktuell bei Google Scholar verzeichneten Zitierungen (inklusive Selbstzitierungen) ermittelt und tabellarisch dargestellt. Zusätzlich werden einfache Maße, u. a. die durchschnittliche Zitierungszahl und der h-Index, berechnet.

Die wesentliche Problemstellung bei der OCS-Anwendung liegt darin, eine möglichst gute Datenqualität für die Analyse in möglichst kurzer Zeit zu realisieren. Die Qualität der präsentierten Ergebnisse hängt im Wesentlichen von zwei Faktoren ab. Zunächst müssen nach Möglichkeit alle relevante Einträge bei Google Scholar gefunden werden. Daraus ergibt sich die Anforderung, mit möglichst wenigen Anfragen, deren Anzahl im Wesentlichen die Laufzeit der Applikation bestimmt, möglichst alle relevanten Einträge zu ermitteln. Der zweite Faktor umfasst die Qualität des Match-Ergebnisses, die den Nutzer zufriedenstellen muss.

[55] http://labs.dbs.uni-leipzig.de/ocs

Strategie	Anfrage	#Anfragen
Name	Name des Autoren bzw. Venues	1
Titelmuster	Disjunktion von Titelmustern	≈ #Publikationen/10
Schlagwort	Titel als Liste von Schlagworten	#Publikationen

Tabelle 8.3: Übersicht über die drei Anfragestrategien des OCS

Die Zitierungsanalyse erfolgt dazu bei OCS in mehreren Schritten. Zunächst werden mit einer Schlagwortsuche (z. B. „Rahm“) Listen von in DBLP verzeichneten Autoren und Venues erzeugt. Daraus wählt der Nutzer einen Autoren oder ein Venue aus und OCS bestimmt die zugehörigen DBLP-Publikationen. Zur Bestimmung der korrespondierenden GS-Publikationen wird eine dreistufige Anfragestrategie durchgeführt, die in Tabelle 8.3 zusammengefasst ist und nachfolgend kurz erklärt wird. Anschließend erfolgt ein Matching der gefundenen GS-Publikationen mit den DBLP-Publikationen auf Basis der Ähnlichkeit von Titel und Autoren sowie des Erscheinungsjahres. Dabei kann der Nutzer den Match-Prozess durch Angabe entsprechender Schwellwerte für die Ähnlichkeiten interaktiv beeinflussen. Es erfolgt eine gruppierte Darstellung des Match-Ergebnisses, welches für jede DBLP-Publikation alle gefundenen Google-Scholar-Einträge darstellt und deren Zitierungszahl summiert.

Die erste Anfragstrategie („Name“) ist sehr einfach und schnell, da nur eine einzige Anfrage mit dem Namen des Autors bzw. Venues an Google Scholar gestellt wird. Sie ist besonderes effektiv für Autoren mit ungewöhnlichen Namen, die eine kurze Publikationsliste besitzen. Auf der anderen Seite führt sie zu vielen irrelevanten Ergebnissen bei Autoren mit häufigen Namen (z. B. Smith). Ein ähnliches Verhalten lässt sich für Venues beobachten. Venues mit eindeutigem Namen (z. B. WebDB oder IIWeb) und relativ wenigen Publikationen liefern gute Ergebnisse. Demgegenüber liefert eine Suche nach „VLDB“ viele irrelevante Ergebnisse, u. a. wegen Publikationen des VLDB Journals sowie der VLDB-Workshops, wobei gleichzeitig viele relevante Ergebnisse nicht gefunden werden, z. B. wenn der Venue-Name mit „Very Large Databases“ anstatt „VLDB“ beschrieben wird.

Die zweite Strategie („Titelmuster“) verwendet die wichtigsten Terme der Publikationstitel auf Basis des TF·IDF-Maßes. Dabei wurde im Vorfeld unter Verwendung aller DBLP-Publikationen die minimale Liste der Terme bestimmt, die den Publikationstitel eindeutig innerhalb der DBLP-Datenquelle bestimmen. (Für verschiedene Publikationen mit dem gleichen Titel, z. B. einer Konferenz- und Journal-Version einer Publikation, ergeben sich an dieser Stelle die gleichen Termlisten. Sie werden jedoch im anschließenden Match-Verfahren, z. B. auf Grund des Erscheinungsjahres, unterschieden.) Ausgehend vom Publikationstitel werden Titelmuster für Google Scholar gebildet, indem alle anderen (d. h. irrelevanten) Terme im Titel durch ein „*“ (Wildcard-Symbol) ersetzt werden. Publikation [154] besitzt viele häufig vorkommende Terme, weshalb das zugehörige Titelmuster („survey * approaches * * schema

matching“) vier Terme enthält. Wegen des ungewöhnlichen Akronyms iFuice kommt das Titelmuster für [158] mit nur einem Term aus („ifuice“)[56]. Google Scholar erlaubt bis zu 32 Schlagworte pro Anfrage, weshalb die Strategie im Durchschnitt ca. 10 Titelmuster mit „OR“ verknüpft („intitle:<Titelmuster1> OR intitle:<Titelmuster2> ...“). Dieser Ansatz erlaubt eine präzise und parallele Suche nach einer Menge von Publikation innerhalb einer Anfrage.

Die dritte Strategie („Schlagwort“) wird nur für diejenigen DBLP-Publikationen angewendet, für welche die beiden ersten Strategien keine korrespondierenden GS-Publikationen ermitteln konnten. Für jede Publikation wird dabei der Publikationstitel als Anfrage verwendet. Auf die Verwendung von Anführungszeichen wird verzichtet, um auch Publikationen mit fehlerhaften Titeln (z. B. auf Grund von Rechtschreib- oder Extraktionsfehlern) zu bestimmen. Diese Strategie hat das Ziel, eine bestimmte Publikation zu finden, und nimmt dafür eine große Anzahl irrelevanter Ergebnisse in Kauf, die im anschließenden Match-Prozess entfernt werden.

Nach der Ausführung der ersten Anfragestrategie erhält der Nutzer bereits die komplette Liste der Publikationen mit den Zitierungszahlen, die durch die erste Strategie ermittelt wurden. Der Nutzer kann bereits die tabellarisch dargestellten Ergebnisse inspizieren, z. B. durch entsprechende Sortierungen. Währenddessen führt OCS bereits die zweite Strategie aus und aktualisiert nach deren Beendigung die Ergebnisse mittels AJAX[57]. Die Ausführung der dritten Strategie erfolgt analog, erfordert jedoch ein explizites Starten durch den Nutzer. Die OCS-Webanwendung realisiert dabei den Kompromiss zwischen der schnellen Verfügbarkeit (maßgeblich beeinflusst durch die Anzahl der Google-Scholar-Anfragen) sowie der Vollständigkeit (Erfassung aller relevanten Google-Scholar-Einträge) der Ergebnisse.

8.6 Zusammenfassung

Als ein Anwendungsfall für die flexible Datenintegration mit iFuice wurde das Thema „Zitierungsanalyse“ vorgestellt. Dabei wurde gezeigt, dass sich mittels iFuice-Skripten Workflows repräsentieren und ausführen lassen, deren Ergebnisse für Zitierungsanalysen unter Verwendung unterschiedlicher Datenquellen genutzt werden können. Die wesentlichen Aspekte des Workflows waren die Erstellung von Same-Mappings, um u. a. gleiche Publikationen in den verschiedenen Datenquellen zu identifizieren, und die Informationsfusion, d. h. die Aggregation gleicher Objekte sowie die Ableitung neuer Attribute. Gleichzeitig wurde verdeutlicht, dass der Qualität der Same-Mappings eine entscheidende Bedeutung für die Relevanz späterer Analysen zukommt. Daher thematisiert das folgende Kapitel die Erstellung von Same-Mappings durch Object Matching.

[56] Führende und abschließende Wildcard-Symbole werden aus dem Titelmuster entfernt.
[57] Asynchronous Javascript and XML, http://en.wikipedia.org/wiki/AJAX

9 Mapping-basiertes Object Matching

In diesem Kapitel wird das Object-Matching-Framework MOMA (Mapping-based Object Matching) vorgestellt. Dabei werden zunächst die Motivation und Ideen des Frameworks präsentiert (Abschnitt 9.1). Anschließend erfolgt die Darstellung der Framework-Architektur (Abschnitt 9.2), wobei insbesondere auf das zentrale Element der Mapping-Kombination (Abschnitt 9.3) eingegangen wird. Das Kapitel wird durch die Vorstellung verschiedener Match-Strategien abgeschlossen (Abschnitt 9.4), die in verschiedenen Domänen flexibel einsetzbar sind.

9.1 Idee

Mit MOMA wird ein auf iFuice aufsetzendes Object-Matching-Framework vorgestellt. Es beinhaltet keinen starren Match-Algorithmus, sondern bietet die Möglichkeit, eigene Match-Workflows einfach zu definieren und auszuführen. Da das MOMA-Framework auf iFuice basiert, verwendet es dessen Datenstrukturen, d. h. Objektinstanzen und Mappings. Daher ergibt sich folgende Problemstellung für das Matching.

Problemstellung: Object Matching ist ein Prozess, der als Eingabe zwei Mengen von Objektinstanzen $A \subseteq LDS_A$ und $B \subseteq LDS_B$ des gleichen Objekttyps erhält. Die Ausgabe ist ein Same-Mapping zwischen A und B, welches Korrespondenzen zwischen den Objekten beinhaltet, die das gleiche Abbild der realen Welt beschreiben. Das Ziel ist ein möglichst perfektes Mapping hin-

sichtlich Precision und Recall, d. h. alle relevanten Korrespondenzen sollen im Ergebnis auftreten (Recall=1), aber auch keine falschen (Precision=1).[58]

Object Matching wird somit durch das Generieren von Same-Mappings realisiert. Der Spezialfall der Duplikaterkennung innerhalb einer Datenquelle wird durch Self-Mappings, d. h. Same-Mappings innerhalb einer LDS, repräsentiert. Die Konfidenzwerte der resultierenden Korrespondenzen drücken dabei den Grad der Gleichheit aus und werden im Folgenden auch synonym als Ähnlichkeitswerte bezeichnet.

Das MOMA-Framework unterstützt dabei die in Abschnitt 6.3 vorgestellten drei Phasen des Object-Matching-Prozesses (Finden von Match-Kandidaten, Bestimmung der Ähnlichkeitswerte, Definition der Matching-Regeln) wie folgt:

Match-Kandidaten: MOMA erlaubt für das Matching die Bestimmung von Korrespondenzen zwischen zwei Mengen A und B von Objektinstanzen, wobei jede Objektinstanz mit jeder verglichen wird. Optional kann die Menge der Match-Kandidaten durch ein Same-Mapping als zusätzliche Eingabe eingeschränkt werden. Dabei enthält das resultierende Match-Ergebnis lediglich Korrespondenzen zwischen Objektinstanzen, wenn für diese bereits im übergebenen Same-Mapping eine Korrespondenz vorhanden war. Dies ermöglicht ein effektiveres Matching durch die Reduzierung der nötigen Vergleiche, da nur diejenigen Objektinstanzen miteinander verglichen werden, die im übergebenen Same-Mapping durch eine Korrespondenz miteinander verbunden sind. Solch ein Eingabe-Mapping kann u. a. ein „approximiertes“ Match-Ergebnis sein, das z. B. alle DBLP- und ACM-Publikationen miteinander verknüpft, deren Publikationsjahr gleich ist (analog der Identifikation von Match-Kandidaten mit „billiger“ Vergleichsfunktion in Abschnitt 6.3.1). Anschließend muss z. B. für eine DBLP-Publikation der aufwändigere Vergleich des Publikationstitels nicht mehr für alle ACM-Publikationen, sondern nur für diejenigen des gleichen Jahres durchgeführt werden.

Ähnlichkeitswerte: MOMA unterstützt zur Berechnung von Ähnlichkeitswerten sowohl attribut- als auch kontextbasierte Verfahren. Für attributbasierte Verfahren steht zum einen der integrierte Attribut-Matcher zur Verfügung, der bereits verschiedene generische String-Ähnlichkeitsfunktionen anwenden kann. Zum anderen können nutzerdefinierte Matcher dem MOMA-Framework hinzugefügt werden, die eigene (z. B. domänenspezifische) Ähnlichkeitsfunktionen verwenden. Kontextinformationen werden in MOMA durch Mappings repräsentiert, d. h. dass über Korrespondenzen Informationen zu assoziierten Objektinstanzen verfügbar sind. Durch eine entsprechende Mapping-Verarbeitung, z. B. bei der Neighborhood-Match-Strategie (siehe Abschnitt 9.4.2), lassen sich Kontextinformationen zur Bestimmung von Ähnlichkeitswerten ableiten.

[58] Zur Definition von Precision und Recall sei auf Abschnitt 10.1 verwiesen.

Matching-Regeln: In MOMA lassen sich flexible, nutzerdefinierte Matching-Regeln durch die gezielte Selektion von Korrespondenzen (siehe Abschnitt 9.3.3) innerhalb von Mappings realisieren. Dabei können u. a. die Ähnlichkeitswerte für die Selektion verwendet werden, z. B. um alle Korrespondenzen über einem definierten Ähnlichkeitsgrenzwert zu selektieren oder für jedes Domain-Objekt das ähnlichste Range-Objekt, d. h. das Range-Objekt der Korrespondenz mit dem höchsten Ähnlichkeitswert für das Domain-Objekt, zu bestimmen.

Die Grundidee von MOMA besteht darin, dass das Matching nicht durch einen starren Algorithmus durchgeführt wird, sondern flexibel im Rahmen eines Match-Workflows definiert werden kann. So können z. B. die drei Phasen des Object Matchings mehrfach innerhalb eines Workflows durchlaufen werden. Gleichzeitig wird durch die Framework-Struktur eine einfache Integration verschiedener Match-Verfahren ermöglicht, deren Ergebnisse miteinander kombiniert werden können. Der MOMA-Ansatz wurde dabei von den Arbeiten zu COMA [47] und dessen Erweiterung COMA++ [9] inspiriert. COMA ist ein erweiterbares Framework, das den Einsatz unterschiedlicher Match-Verfahren für ein gegebenes Schema-Matching-Problem. Die einzelnen Match-Ergebnisse lassen sich dabei in vielfältiger Weise miteinander kombinieren. Im Unterschied zu COMA und COMA++, die für Metadaten (u. a. relationale Datenbankschemata, XML-Schemata und OWL-Ontologien) ausgelegt sind, ist das MOMA-Framework auf die Verarbeitung von Instanzdaten ausgerichtet. Alleinstellungsmerkmal von MOMA ist die explizite Unterstützung instanzbasierter Mappings, die sich in vielfältiger Weise zur Bestimmung neuer Mappings kombinieren lassen. Durch die Kombination von Mappings werden insbesondere die folgenden drei Ziele verfolgt:

Qualitätsverbesserung: Durch die Kombination unabhängig voneinander erstellter Match-Ergebnisse für das gleiche Match-Problem kann die Qualität abgesichert werden. So können z. B. bei der Bildung des Durchschnitts nur Korrespondenzen übernommen werden, die in allen Mappings vorkommen, was i. Allg. zu einer Verbesserung der Precision führt. Eine andere Möglichkeit ist die Erweiterung eines bestehenden Same-Mappings um neue, nicht widersprüchliche Korrespondenzen eines zweiten Mappings, d. h. um Korrespondenzen, bei denen es keine Korrespondenz für das Domain-Objekt im ersten Same-Mapping gibt. Dadurch kann bei der Kombination eine Erhöhung des Recalls ohne Verminderung der Precision erreicht werden, so dass i. Allg. eine allgemeine Verbesserung der Match-Qualität resultiert.

Verfeinerung: Die Kombination von Mappings kann zur einer iterativen Verfeinerung der Match-Ergebnisse genutzt werden. Dabei kann ein Match-Verfahren auf den Ergebnissen eines anderen aufbauen, wodurch eine effizientere Berechnung erzielt wird, da durch das erste Mapping bereits eine Einschränkung des Match-Ergebnisses erzielt wird. Ein Beispiel ist die – bereits oben beim

Punkt Match-Kandidaten angesprochene – Berechnung eines Same-Mappings zwischen DBLP- und ACM-Publikationen, bei dem zunächst ein Match-Verfahren das Publikationsjahr vergleicht und dabei typische Problemfälle, z. B. fehlende oder verkürzte Jahrangaben ('07 statt 2007), behandelt. Anschließend analysiert ein anderes Match-Verfahren die Ähnlichkeit der Publikationstitel, wobei u. a. Tippfehler oder typische Titelzusätze (z. B. Demo oder Abstract) beachtet werden. Die Kombination kann dabei die Besonderheiten bzw. Vorteile beider Match-Verfahren in einem Match-Workflow vereinigen.

Wiederverwendung: Auch die Wiederverwendung anderer Match-Ergebnisse ist ein wichtiges Element der Mapping-Kombination. Gibt es z. B. – wie in Abschnitt 8.2 dargestellt – ein Same-Mapping zwischen DBLP- und ACM-Publikationen sowie ein weiteres zwischen ACM und Excel, so kann durch Komposition, d. h. durch „Verkettung" der Korrespondenzen, ein Mapping zwischen DBLP- und Excel-Publikationen erstellt werden. Die beiden Mappings ermöglichen damit eine effiziente Bestimmung eines Match-Ergebnisses. Auch Same-Mappings eines anderen Objekttyps können für das Matching wieder verwendet werden. Das angesprochene Same-Mapping zwischen DBLP- und ACM-Publikationen kann z. B. verwendet werden, um ein Same-Mapping zwischen den Venues der beiden Datenquellen zu bestimmen. Durch die Kombination mit den Assoziations-Mappings zwischen Publikationen und Venues können z. B. zwei Venues als gleich erkannt werden, wenn 80% ihrer Publikationen im Publikations-Same-Mapping als gleich definiert sind.

9.2 Architektur und Match-Workflows

Zentrales Element des MOMA-Frameworks sind sogenannte Match-Workflows.

Match-Workflow: Ein Match-Workflow erhält als Eingabe zwei Mengen von Objektinstanzen $A \subseteq LDS_A$ und $B \subseteq LDS_B$ des gleichen Objekttyps. Die Ausgabe eines Match-Workflows ist ein Same-Mapping des Typs $LDS_A \times LDS_B$ mit Korrespondenzen zwischen den Eingabeobjekten. Optional kann als zusätzliche Workflow-Eingabe ein Same-Mapping zwischen LDS_A und LDS_B verwendet werden, um die Ausführung des Workflows effizienter zu gestalten. Dabei wird das Matching nur zwischen denjenigen Paaren von Objektinstanzen durchgeführt, für die bereits im Eingabe-Mapping eine Korrespondenz vorhanden war. Zusätzlich kann ein Match-Workflow beliebig viele weitere Mappings während seiner Ausführung verwenden.

Aufgabe der Match-Workflows ist die Generierung von Same-Mappings. Workflows können als iFuice-Skripte repräsentiert werden und dadurch innerhalb von iFuice

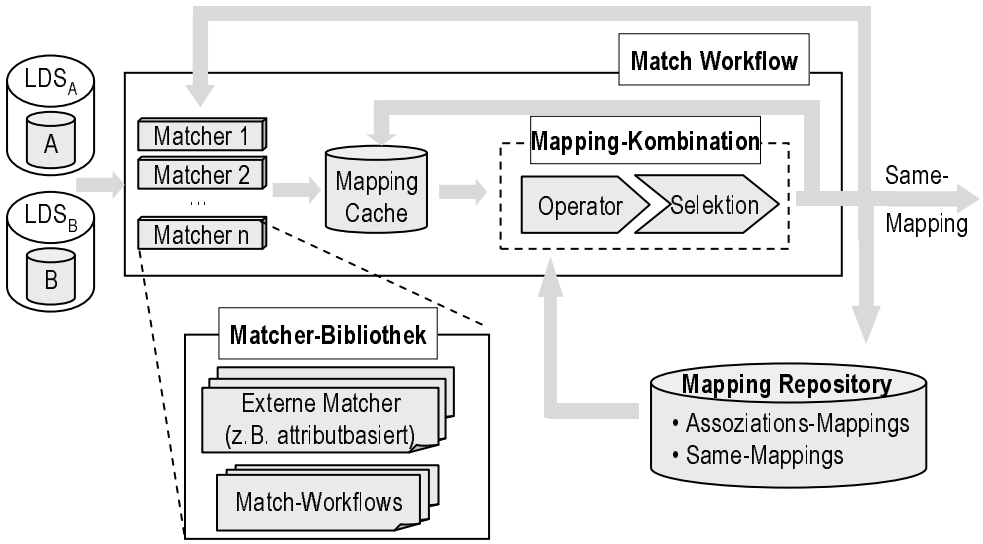


Abbildung 9.1: Architektur von MOMA

ausgeführt werden. Mit MOMA wurde ein Framework entwickelt, welches die Konstruktion verschiedener Match-Workflows erlaubt. Dabei können die Ergebnisse verschiedener Matcher flexibel miteinander kombiniert werden. Abbildung 9.1 zeigt sowohl die Architektur des MOMA-Frameworks als auch den allgemeinen Prozess des Object Matchings, d. h. den prinzipiellen Aufbau der Match-Workflows. Die zwei Hauptkomponenten des MOMA-Frameworks sind eine erweiterbare Matcher-Bibliothek und das generische Mapping Repository.

Matcher-Bibliothek: In der Matcher-Bibliothek sind alle verfügbaren Matcher zusammengefasst. Jeder Matcher wird eindeutig durch einen Namen charakterisiert, so dass die Ausführung eines Matchers anhand seines Namen und der benötigten Eingabe-Parameter von der Matcher-Bibliothek durchgeführt werden kann. Ein Matcher erhält als Eingabe zwei Mengen von Objektinstanzen $A \subseteq LDS_A$ und $B \subseteq LDS_B$ des gleichen Objekttyps oder ein Same-Mapping des Typs $LDS_A \times LDS_B$.[59] Die Ausgabe eines Matchers ist ein Same-Mapping des Typs $LDS_A \times LDS_B$.

Die von der Matcher-Bibliothek geforderte Schnittstellendefinition eines Matchers ermöglicht die folgenden drei Arten von Matchern.

- Ein Matcher kann durch ein *Match-Mapping* (siehe Abschnitt 7.2.2) definiert werden. Das Match-Mapping wird dazu im Source-Mapping-Modell registriert und ermittelt ein Same-Mapping zwischen zwei definierten LDS LDS_A

[59] Bei der Angabe eines Same-Mappings zur Einschränkung des Match-Ergebnisses kann die Angabe der Objektinstanzen A und B entfallen, da diese sich implizit aus den Domain- bzw. Range-Objekten des Mappings ergeben.

und LDS_B. Die Implementation wird durch den iFuice-Mediator mit seinen Wrappern gekapselt, so dass beliebige „externe“ Match-Verfahren in das MOMA-Framework integriert werden können. Ein Match-Mapping kann also z. B. durch ein Java-Programm oder entsprechende Datenbankfunktionalitäten, z. B. Fuzzy Match im SQL Server 2005 [32], realisiert sein. Dabei können z. B. sowohl Hilfsinformationen in Form von Wörterbüchern, Abkürzungsverzeichnissen oder Synonymlisten als auch domänenspezifische Ähnlichkeitsmaße, z. B. die geografische Nähe zweier Objekte, genutzt werden.

- MOMA selbst besitzt bereits einen *generischen Attribut-Matcher*, welcher durch den iFuice-Operator `attrMatch` ausführbar ist. Der generische Attribut-Matcher kann durch eine entsprechende Parametrisierung (mehrere) Attributwerte verschiedener Objektinstanzen miteinander vergleichen. Dabei stehen verschiedene String-Ähnlichkeitsmaße (u. a. Trigram, Edit Distance, TF·IDF) zur Verfügung, mit deren Hilfe die Konfidenzen des resultierenden Same-Mappings bestimmt werden. Die genaue Parametrisierung ist im Anhang (siehe Abschnitt A.5.4) aufgeführt.

- Die dritte Art von Matchern sind *Match-Workflows*, da auch sie die von der Matcher-Bibliothek geforderte Schnittstellendefinition aufweisen. Dadurch kann jeder definierte Match-Workflow in der Bibliothek gespeichert werden und im Rahmen eines anderen Match-Workflows ausgeführt werden. Match-Workflows können dabei als parametrisierte iFuice-Prozeduren definiert werden, was insbesondere die Erstellung sogenannter Match-Strategien als flexibles Instrument für das Object Matching ermöglicht. Match-Strategien werden dazu später in Abschnitt 9.4 vorgestellt.

Die zweite Hauptkomponente des MOMA-Frameworks ist das Mapping Repository.

Mapping Repository: Im Mapping Repository sind alle verfügbaren Mappings gespeichert, d. h. sowohl Same- als auch Assoziations-Mappings. Diese Mappings entstehen zunächst durch die Ausführung von ID-Mappings des Source-Mapping-Modells. Zusätzlich kann das Ergebnis jedes Match-Workflows zur späteren Wiederverwendung im Repository abgespeichert werden. Mit Hilfe des Repository kann somit innerhalb jedes Match-Workflows auf alle verfügbaren Mappings zurückgegriffen werden.

Neben der Architektur des MOMA-Frameworks zeigt Abbildung 9.1 auch den prinzipiellen Aufbau der Match-Workflows. Jeder Match-Workflow gliedert sich in einen oder mehrere Match-Schritte, die jeweils aus zwei Teilen bestehen:

Matcher-Ausführung: Im ersten (optionalen) Teil werden ein oder mehrere Matcher der Matcher-Bibliothek unabhängig voneinander zur Ausführung gebracht. Die Ergebnisse jedes Matchers werden im Mapping-Cache zwischengespeichert.

Mapping-Kombination: Im zweiten Teil werden Mappings miteinander kombiniert. Dabei werden sowohl Mappings aus dem Mapping-Cache als auch aus dem Repository verwendet. Die Kombination erfolgt durch (mehrfache) Anwendung sogenannter Mapping-Kombinatoren, die jeweils aus einem Mapping-Operator und einer anschließenden (optionalen) Selektion bestehen. Ein Mapping-Operator kombiniert zwei oder mehrere Mappings miteinander, z. B. durch Bildung der Vereinigung oder Komposition. Die anschließende Selektion filtert die resultierenden Korrespondenzen nach einem oder mehreren Kriterien, z. B. einem Grenzwert für die Konfidenz. Das Ergebnis jedes Kombinators wird im Mapping-Cache gespeichert. Dadurch wird gewährleistet, dass weitere Kombinatoren auf die Ergebnisse zurückgreifen können.

Nach Ausführung beider Teile kann der Match-Workflow durch eine weitere Iteration, d. h. erneute Ausführung eines Match-Schrittes, fortgeführt werden. Alternativ kann der Workflow beendet werden. Das Ergebnis des Match-Workflows kann als Same-Mapping im Repository gespeichert werden, um für weitere Workflows zur Verfügung zu stehen.

Der vorgestellte Aufbau von Match-Workflows ist sehr flexibel, da er sowohl die Wiederverwendung von Mappings als auch die Kombination unterschiedlicher Match-Verfahren unterstützt. Des Weiteren können verschiedene Match-Workflows sich gegenseitig unterstützen. Zum Beispiel kann das Ergebnis eines Match-Workflows zwischen Publikationen zweier Datenquellen für das Matching von Venues genutzt werden – und umgekehrt kann ein Venue-Same-Mapping zur Bestimmung eines Publikations-Same-Mappings genutzt werden.

Im folgenden Abschnitt 9.3 werden alle für die Mapping-Kombination zur Verfügung stehenden Operatoren näher vorgestellt. Anschließend zeigt Abschnitt 9.4 die Erstellung von Match-Workflows unter Verwendung sogenannter Match-Strategien.

9.3 Operatoren zur Mapping-Kombination

MOMA verwendet für die Mapping-Kombination die Operatoren `intersect`, `union` und `compose` der Datenintegrationsplattform iFuice (siehe Abschnitt 7.4). Die Mapping-Kombinationsoperatoren erhalten zwei (oder mehr) Mappings als Eingabe und liefern jeweils wieder ein Mapping zurück. Die Bestimmung der resultierenden Korrespondenzen setzt sich aus zwei Teilen zusammen. Zunächst ermittelt der Operator die korrespondierenden Objektinstanzen im Ergebnis-Mapping. Anschließend – und darauf liegt insbesondere der Schwerpunkt beim Object Matching – werden die zugehörigen Ähnlichkeitswerte bestimmt. Das bedeutet z. B. bei der Vereinigung zweier Mappings, dass, wenn z. B. eine Korrespondenz (a, b, c_1) im ersten Mapping und eine weitere Korrespondenz (a, b, c_2) im zweiten Mapping zwischen den gleichen Objekten existieren, definiert wird, wie sich der Konfidenzwert c für die Korrespondenz

zwischen a und b im vereinigten Mapping bestimmt. Das kann im Beispiel u. a. der minimale oder maximale Wert von c_1 und c_2 sein. Zur Definition, wie die Ähnlichkeitswerte berechnet werden, erhält daher jeder Operator als zusätzlichen Parameter eine (bei `intersect` und `union`) bzw. zwei (bei `compose`) Konfidenzfunktionen.

Im Folgenden werden mit Durchschnitt, Vereinigung und Komposition die Arten der Mapping-Kombination vorgestellt, ehe abschließend die verschiedenen Selektionsstrategien präsentiert werden.

9.3.1 Durchschnitt und Vereinigung von Mappings

Für den Durchschnitt und die Vereinigung von Mappings stehen die Operatoren `intersect` bzw. `union` zur Verfügung.

$$\texttt{intersect}(MR_1, MR_2, ..., MR_n, \vec{h})$$

$$\texttt{union}(MR_1, MR_2, ..., MR_n, \vec{h})$$

Beide Operatoren erhalten als Eingabe n ($n \geq 2$) Mappings des gleichen Typs $LDS_A \times LDS_B$, d. h. nur Mappings zwischen Objektinstanzen der gleichen Domain- und Range-LDS LDS_A und LDS_B können vereinigt werden. `intersect` ist ein sehr restriktiver Operator, der lediglich diejenigen Korrespondenzen ermittelt, welche in allen Eingabe-Mappings auftreten. Die Anwendung von `intersect` erhöht daher i. Allg. die Precision auf Kosten des Recalls. Im Gegensatz dazu fasst `union` alle auftretenden Korrespondenzen zusammen und realisiert dadurch einen hohen Recall. Auf Grund der Tatsache, dass eine Korrespondenz nur in einem Eingabe-Mapping vorhanden sein muss, kann die Erhöhung des Recalls zu einer Verringerung der Precision führen.

Beiden Operatoren wird zusätzlich eine Konfidenzfunktion $\vec{h}$ übergeben, welche die Berechnung der Konfidenz c für alle resultierenden Korrespondenzen (a, b, c) aus den Konfidenzen c_i der Ursprungskorrespondenzen $(a, b, c_i) \in MR_i$ bestimmt. Nachfolgend werden alle zur Verfügung stehenden Konfidenzfunktionen definiert und Abbildung 9.2 illustriert an einem Beispiel für `union`[60] die Wirkungsweise der unterschiedlichen Funktionen. Dabei muss für den Operator `union` in den meisten Fällen geklärt werden, wie mit fehlenden Konfidenzen umzugehen ist. Diese treten auf, wenn eine im Ergebnis auftretende Korrespondenz nicht in allen Eingabe-Mappings vorkommt. Im Beispiel von Abbildung 9.2 ist dies für die Korrespondenzen (a_2, b_2) und (a_3, b_3) der Fall.

Max: Die Konfidenz ist das Maximum der Ursprungskonfidenzen.

[60] Bei der Verwendung von `intersect` erscheint jeweils nur die erste Korrespondenz zwischen a_1 und b_1 in den Ergebnis-Mappings.

MR_1

A	B	Conf
a_1	b_1	1
a_2	b_2	0.8

MR_2

A	B	Conf
a_1	b_1	0.6
a_3	b_3	0.6

union (Max)

A	B	Conf
a_1	b_1	1
a_2	b_2	0.8
a_3	b_3	0.6

union (Avg)

A	B	Conf
a_1	b_1	0.8
a_2	b_2	0.8
a_3	b_3	0.6

union (Weighted)

A	B	Conf
a_1	b_1	0.7
a_2	b_2	0.8
a_3	b_3	0.6

union (Min)

A	B	Conf
a_1	b_1	0.6
a_2	b_2	0.8
a_3	b_3	0.6

union (Ranked)

A	B	Conf
a_1	b_1	1
a_2	b_2	0.8
a_3	b_3	0.6

union (Avg-0)

A	B	Conf
a_1	b_1	0.8
a_2	b_2	0.4
a_3	b_3	0.3

union (Weighted-0)

A	B	Conf
a_1	b_1	0.7
a_2	b_2	0.2
a_3	b_3	0.45

union (Min-0)

A	B	Conf
a_1	b_1	0.6
a_2	b_2	0
a_3	b_3	0

Abbildung 9.2: Ergebnis von `union` für verschiedene Konfidenzfunktionen

Avg / Avg-0: Die Konfidenz ist das arithmetische Mittel der Ursprungskonfidenzen. Fehlende Konfidenzen werden bei *Avg* ignoriert, wohingegen sie bei *Avg-0* mit 0 angenommen werden. Im Beispiel von Abbildung 9.2 hat daher die Korrespondenz (a_2, b_2) für *Avg* eine Konfidenz von 0.8 und für *Avg-0* eine Konfidenz von $(0.8 + 0)/2 = 0.4$.

Weighted / Weighted-0: Die Konfidenz ist das gewichtete Mittel der Ursprungskonfidenzen. Für diese Funktion müssen dazu n ganzzahlige Gewichte angegeben werden. Für das Beispiel wurden als Gewichte (in dieser Reihenfolge) 1 und 3 ausgewählt. Die Korrespondenz (a_1, b_1) hat daher eine Konfidenz von $(1 \cdot 1 + 3 \cdot 0.6)/(1 + 3) = 0.7$. Analog zu *Avg* werden bei *Weighted* fehlende Konfidenzen ignoriert (z. B. Konfidenz $= 0.8$ für (a_2, b_2)) und bei *Weighted-0* durch Null ersetzt (z. B. Konfidenz $= (1 \cdot 0.8 + 3 \cdot 0)/(1 + 3) = 0.2$ für (a_2, b_2)).

Min / Min-0: Die Konfidenz ist das Minimum der Ursprungskonfidenzen. Das Ersetzen fehlender Konfidenzwerte durch Null führt bei *Min-0* automatisch zu einer resultierenden Konfidenz von 0 (siehe u. a. (a_2, b_2)).

Ranked: Als Konfidenz wird der Konfidenzwert des ersten Eingabe-Mappings verwendet. Falls die resultierende Korrespondenz nicht in dem ersten Mapping auftritt (nur bei `union` möglich), wird die Konfidenz des zweiten Mappings verwendet; falls sie nicht in den ersten beiden Mappings auftritt, wird die Konfidenz des dritten Mappings verwendet usw.

Abbildung 9.2 zeigt die Wirkungsweise der Funktionen für den Operator `union` und veranschaulicht insbesondere bei der Konfidenzfunktion *Min-0* das eingangs geschilderte Zusammenspiel von Operator und Konfidenzfunktion. Der Operator bestimmt, welche Korrespondenzen im Ergebnis-Mapping vertreten sind, weshalb unabhängig von der Konfidenzfunktion Korrespondenzen zwischen a_2 und b_2 bzw. a_3 und b_3 in

allen Ergebnis-Mappings erscheinen. Die Konfidenzfunktion definiert für jede Korrespondenz die Berechnung der Konfidenz, weshalb für die zwei genannten Korrespondenzen bei *Min-0* ein Konfidenzwert von 0 resultiert. Sollen Korrespondenzen mit einer Konfidenz von 0 nicht im Mapping vertreten sein, kann dies mit einer anschließenden Selektion realisiert werden.

Die Auswahl der Konfidenzfunktion h hängt vom konkreten Einsatzfall, d. h. sowohl von den verwendeten Eingabe-Mappings als auch vom Ziel der Mapping-Kombination, ab. Während z. B. *Min* sehr restriktiv ist, ist *Max* eine „optimistische" Konfidenzfunktion, die höhere Konfidenzwerte ermittelt. Letztere kann z. B. verwendet werden, wenn die Eingabe-Mappings (vermutlich) eine hohe Precision aufweisen. Das Ignorieren fehlender Korrespondenzen ist insbesondere dann sinnvoll, wenn Eingabe-Mappings auf Grund ihrer Entstehung nicht vollständig sind. Für einige Publikationen von Google Scholar liegt z. B. keine Jahreszahl vor, so dass ein entsprechender Attribut-Matcher für die Instanzen keine Korrespondenzen finden kann. Ein Fehlen dieser Korrespondenzen bedeutet jedoch nicht, dass die entsprechenden Instanzen nicht gleich sind.

9.3.2 Komposition von Mappings

Eine weitere Art der Mapping-Kombination stellt die Komposition dar, welche durch den Operator `compose` bestimmt wird.

$$\texttt{compose}(MR_1, MR_2, \vec{h}, \vec{v})$$

Eingabe sind ein Mapping MR_1 von LDS_A nach LDS_X und ein weiteres Mapping MR_2 von LDS_X nach LDS_B. Die Komposition bildet nun ein Mapping MR_3 von LDS_A nach LDS_B, welches Korrespondenzen (a, b, c) enthält, für die es mindestens ein $x \in LDS_X$ gibt, so dass $(a, x, c_1) \in MR_1$ und $(x, b, c_2) \in MR_2$ für beliebige c_1 und c_2 gilt. Die Berechnung des Ähnlichkeitswertes ist komplexer als bei der Mapping-Vereinigung und muss zwei verschiedene Informationen berücksichtigen:

Konfidenzfunktion $\vec{h}$: Die Konfidenz eines Kompositionspfades (a, x, b) setzt sich aus den zwei Konfidenzen c_1 und c_2 der Korrespondenzen (a, x, c_1) bzw. (x, b, c_2) zusammen.

Konfidenzfunktion $\vec{v}$: Eine resultierende Korrespondenz (a, b) kann über mehrere Kompositionspfade erreicht werden, d. h. es kann mehrere x_i geben, so dass $(a, x_i, c_{i_1}) \in MR_1$ und $(x_i, b, c_{i_2}) \in MR_2$. Daher ergibt sich die Konfidenz der Korrespondenz (a, b, c) aus der Menge der Konfidenzen $\vec{h}(c_{i_1}, c_{i_2})$ der zugehörigen Kompositionspfade.

Es werden daher sowohl eine („horizontale“) Konfidenzfunktion $\vec{h}$ als auch eine („vertikale“) Funktion $\vec{v}$ verwendet. Die Begriffe horizontal und vertikal stehen dabei sinnbildlich für die in Abbildung 9.3 (Seite 184) gewählte Illustration der Mappings, bei der die Kompositionspfade horizontal verlaufen und mehrere Pfade zwischen den gleichen Objekten vertikal zusammengefasst werden. Da für $\vec{h}$ stets genau zwei Konfidenzwerte zur Verfügung stehen, sind folgende Funktionen verfügbar:

Max / Avg / Min: Die Konfidenz ist das Maximum / arithmetische Mittel / Minimum der Ursprungskonfidenzen.

Weighted: Die Konfidenz ist das gewichtete Mittel der Ursprungskonfidenzen. Für diese Funktion müssen dazu zwei ganzzahlige Gewichte angegeben werden. *Avg* ist ein Spezialfall von *Weighted* mit den Gewichten 1 und 1.

Left / Right: Die Konfidenz c_1 / c_2 der Korrespondenz aus MR_1 / MR_2 wird verwendet.

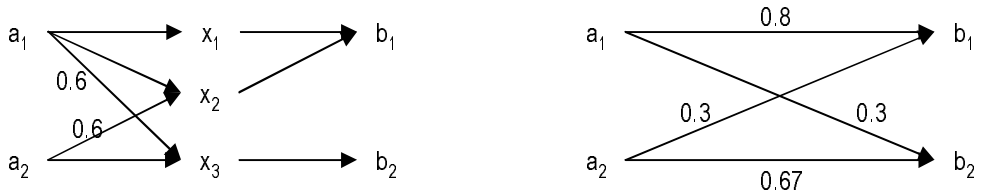
Die Bestimmung der Konfidenz aus den jeweiligen Kompositionspfaden kann analog zu den Konfidenzfunktionen bei `union` mittels Maximum, arithmetischem Mittel und Minimum erfolgen. Diese einfache Art der Verrechnung ignoriert allerdings die Tatsache, wie viele Kompositionspfade für eine Korrespondenz vorliegen. Wenn eine Korrespondenz über viele Kompositionspfade erreicht wird, kann dies als Indiz für die Qualität der Korrespondenz angesehen werden. Die absolute Anzahl der Kompositionspfade ist für eine Korrespondenz (a, b) nicht aussagekräftig, sondern vielmehr ihr Verhältnis zur Anzahl der Korrespondenzen mit dem Domain-Objekt a in MR_1 bzw. Range-Objekt b in MR_2, da das Minimum dieser beiden Anzahlen die maximale Anzahl möglicher Kompositionspfade (a, x, b) definiert.

Dazu werden zunächst mit $n(a)$ die Anzahl der Korrespondenzen für ein Domain-Objekt a in MR_1 und mit $n(b)$ die Anzahl der Korrespondenzen für ein Range-Objekt b in MR_2 bezeichnet. Zusätzlich gibt $s(a, b)$ für jede resultierende Korrespondenz die Summe der Konfidenzen aller Kompositionspfade an.

$$
\begin{aligned}
n(a) &= |\{x_i | \exists c_i : (a, x_i, c_i) \in MR_1\}| \\
n(b) &= |\{x_i | \exists c_i : (x_i, b, c_i) \in MR_2\}| \\
s(a, b) &= \sum_{x_i} \vec{h}\,(c_{i_1}, c_{i_2}) \quad \text{mit} \quad (a, x_i, c_{i_1}) \in MR_1 \wedge (x_i, b, c_{i_2}) \in MR_2
\end{aligned}
$$

Für $\vec{v}$ stehen die folgenden Funktionen zur Verfügung, die z. T. von den obigen Werten Gebrauch machen:

Max / Avg / Min: Die Konfidenz ist das Maximum / arithmetische Mittel / Minimum der Konfidenzen.



MR_1		
A	X	Conf
a_1	x_1	1
a_1	x_2	1
a_1	x_3	0.6
a_2	x_2	0.6
a_2	x_3	1

MR_2		
X	B	Conf
x_1	b_1	1
x_2	b_1	1
x_3	b_2	1

compose (Min, Dice)		
A	B	Conf
a_1	b_1	0.8 = 2 * (1+1) / (3+2)
a_1	b_2	0.3 = 2 * 0.6 / (3+1)
a_2	b_1	0.3 = 2 * 0.6 / (2+2)
a_2	b_2	0.67 = 2 * 1 / (2+1)

Abbildung 9.3: Beispiel für die Anwendung des `compose`-Operators mit den Funktionen $\vec{h} = Min$ und $\vec{v} = Dice$

DiceLeft: $= s(a,b)/n(a) =$ Die Summe der Konfidenzen der Kompositionspfade wird durch die Anzahl der (ausgehenden) Korrespondenzen für das Domain-Objekt a in MR_1 dividiert.

DiceRight: $= s(a,b)/n(b) =$ Die Summe der Konfidenzen der Kompositionspfade wird durch die Anzahl der (eingehenden) Korrespondenzen für das Range-Objekt b in MR_2 dividiert.

DiceMin: $= s(a,b)/min(n(a),n(b)) =$ Die Summe der Konfidenzen der Kompositionspfade wird durch das Minimum der Anzahl der (ausgehenden) Korrespondenzen für das Domain-Objekt a in MR_1 und der Anzahl der (eingehenden) Korrespondenzen für das Range-Objekt b in MR_2 dividiert.

Dice: $= 2 \cdot s(a,b)/(n(a) + n(b)) =$ Harmonisches Mittel von $DiceLeft$ und $DiceRight$, d. h. die doppelte Summe der Konfidenzen der Kompositionspfade wird durch die Summe der ausgehenden und eingehenden Korrespondenzen beider Mappings dividiert.

Abbildung 9.3 zeigt ein Beispiel für die Anwendung des `compose`-Operators mit den Funktionen $\vec{h} = Min$ und $\vec{v} = Dice$. Durch den Operator werden die Objektinstanzen a_1 und a_2 mit b_1 und b_2 durch jeweils eine Korrespondenz verbunden, da es für jede der vier resultierenden Korrespondenzen mindestens einen Kompositionspfad gibt. Am Beispiel der Korrespondenz (a_1, b_1) wird die Berechnung der Konfidenz erklärt. Die Korrespondenz (a_1, b_1) wird über zwei Kompositionspfade erreicht (via x_1 und x_2). Die Kompositionspfade erhalten durch die Min-Funktion jeweils den Konfidenzwert 1. Die vertikale Funktion $Dice$ dividiert nun die doppelte Summe $(2 \cdot (1 + 1) = 4)$ durch die Summe der Anzahl der Korrespondenzen mit a_1 in MR_1

(3) und der Anzahl der Korrespondenzen mit b_1 in MR_2 (2). Dadurch ergibt sich ein Konfidenz von $4/(3+2) = 0.8$ für die resultierende Korrespondenz (a_1, b_1).

Das Beispiel illustriert, dass die *Dice*-Funktionen neben den einzelnen Konfidenzen auch die Anzahl der Pfade, über die eine resultierende Korrespondenz entsteht, berücksichtigen. Im angeführten Beispiel erhält (a_1, b_1) im Wesentlichen deshalb eine höhere Konfidenz als (a_1, b_2), da erstere über mehr Pfade erreicht wird. Diese Idee wird insbesondere in Abschnitt 9.4.2 aufgegriffen, innerhalb dessen Match-Strategien mittels Kombination von Same- und Assoziations-Mappings vorstellt werden. Dabei wird z. B. ein Same-Mapping zwischen Venues unter Verwendung der assoziierten Publikationen bestimmt. Die Grundidee, dass zwei Venues um so ähnlicher sind, je ähnlicher die Menge der jeweils assoziierten Publikationen ist, wird u. a. mittels der *Dice*-Funktionen bei der Komposition umgesetzt. Der relative Anteil der Publikationen p eines Venues v, für welche die korrespondierenden Publikationen p' dem gleichen Venue v' zugeordnet sind, kann dadurch als Konfidenz der Korrespondenz (v, v') beschrieben werden.

9.3.3 Selektion

Für die Selektion steht der Operator `queryMap` zur Verfügung.

$$\texttt{queryMap}(MR, condition)$$

Dabei werden diejenigen Korrespondezen selektiert, welche die Filterbedingung erfüllen. Folgende Bedingungsarten stehen zur Verfügung.

Threshold: Selektiert alle Korrespondenzen, deren Konfidenz einen gegebenen Schwellwert t übersteigt.

Best-N: Selektiert für jedes Domain-Objekt diejenigen Korrespondenzen mit den N größten Konfidenzen.

Best-1+Delta: Selektiert für jedes Domain-Objekt diejenige Korrespondenz mit der höchsten Konfidenz c und zusätzlich alle Korrespondenzen des Domain-Objekts, deren Konfidenz größer als $c - \Delta$ ist.

Attributbedingung: Selektiert alle Korrespondenzen, die eine bestimmte Bedingung bzgl. der Attributwerte der korrespondierenden Objekte erfüllen.

Abbildung 9.4 illustriert jeweils ein Beispiel für die ersten drei Bedingungsarten. Ein Beispiel für den letzten Selektionstyp ist, dass bei einem Publikations-Same-Mapping der Unterschied der Jahreszahlen korrespondierender Publikationen nicht größer als 1 sein soll.

MR		
A	B	Conf
a_1	b_1	1
a_1	b_2	0.9
a_1	b_3	0.4
a_2	b_2	0.6

Threshold > 0.8		
A	B	Conf
a_1	b_1	1
a_1	b_2	0.9

Best-1		
A	B	Conf
a_1	b_1	1
a_2	b_2	0.6

Best-1 + Delta 0.2		
A	B	Conf
a_1	b_1	1
a_1	b_2	0.9
a_2	b_2	0.6

Abbildung 9.4: Beispiel für die Selektion von Korrespondenzen

9.4 Match-Strategien

Die Definition konkreter Workflows ist abhängig von den verwendeten Datenquellen, z. B. welche Attribute mittels eines Attribut-Matchers verglichen werden oder welche Mappings zur Kombination zur Verfügung stehen. Dennoch gibt es wiederkehrende Muster, die – unterschiedlich parametrisiert – in verschiedenen Workflows Anwendung finden (können). Daher werden im Folgenden sogenannte Match-Strategien vorgestellt, d. h. allgemeine Verfahren, nach denen (Teile von) Match-Workflows aufgebaut sein können. Solche Strategien ermitteln ein Same-Mapping auf Basis eines bestimmten Algorithmus und sind bei Verfügbarkeit entsprechender Mappings bzw. Matcher in verschiedenen Kontexten einsetzbar.

Alle präsentierten Match-Strategien werden als generische iFuice-Prozedur dargestellt, so dass sie domänenunabhängig eingesetzt werden können. Als Beispiel werden sie jeweils innerhalb eines konkreten Match-Workflows der bibliografischen Domäne verwendet, d. h. es werden Same-Mappings zwischen Publikationen, Autoren und Venues der Datenquellen DBLP, ACM und GS gebildet.

9.4.1 Kombination von Same-Mappings

Für die Kombination von Same-Mappings werden im Folgenden die drei Strategien *Merge*, *Prefer* und *Hub*[61] vorgestellt.

Match-Strategie: Merge

Für ein konkretes Match-Problem lassen sich i. Allg. verschiedene Matcher anwenden. Will man z. B. ein Same-Mapping zwischen DBLP- und ACM-Publikationen herstellen, kann man mit Hilfe eines Attribut-Matchers die Ähnlichkeit des Titels vergleichen. Dabei gibt es u. a. verschiedene Ähnlichkeitsmaße, z. B. TF·IDF oder Edit Distance. Weiterhin kann man nicht nur den Titel beim Matching verwenden, sondern z. B. auch die Liste der Autoren (z. B. als kommaseparierten String). Die

[61] *eng. fig.* hub = Mittel-, Angelpunkt

```
PROCEDURE MergeMatch ($map1, $map2)
  $merge := union ($map1, $map2, Avg-0);
  $result := queryMap ($merge, "[_confidence]>0.8");
  RETURN $result;
END
```

Abbildung 9.5: iFuice-Repräsentation der Match-Strategie Merge

```
PROCEDURE PreferMatch ($map1, $map2)
  $diff := diff (domain ($map2), domain ($map1));
  $add := map ($diff, $map2);
  $result := union ($map1, $add, Max);
  RETURN $result;
END
```

Abbildung 9.6: iFuice-Repräsentation der Match-Strategie Prefer

Ergebnisse der verschiedenen Matcher können dabei mit `union` kombiniert werden. Abbildung 9.5 zeigt die Repräsentation der Match-Strategie Merge als iFuice-Prozedur. Aus Gründen der Übersichtlichkeit wurden dabei nur zwei Mappings als Parameter gewählt. Selbstverständlich ist es möglich, weitere Parameter hinzuzufügen, um die Prozedur flexibler einsetzen zu können. Zum Beispiel kann der Schwellwert (in der Abbildung mit 0.8 angegeben) als zusätzlicher Parameter übergeben werden. Beim Aufruf des `queryMap`-Operators würde dann entsprechend die Bedingung dynamisch mit Hilfe des Parameters erzeugt.

Die Merge-Strategie lässt sich nun z. B. für das geschilderte Szenario für Publikations-Same-Mappings einsetzen. Mit folgendem iFuice-Skript werden zunächst zwei Mappings anhand der Stringähnlichkeit der Titel (DBLP-Attribut `title` bzw. ACM-Attribut `name`) bzw. Autoren bestimmt. Diese werden anschließend kombiniert.

```
$map1 := attrMatch ($DBLPPub, $ACMPub, Trigram,
           "[DBLP.title]", "[ACM.name]");
$map2 := attrMatch ($DBLPPub, $ACMPub, Trigram,
           "[DBLP.authors]", "[ACM.authors]");
$same := exec (MergeMatch, $map1, $map2);
```

Match-Strategie: Prefer

Während die Strategie Merge die Eingabe-Mappings gleich gewichtet, wird mittels Prefer ein Mapping bei der Kombination zweier Mappings bevorzugt. Dies ist insbesondere dann sinnvoll, wenn für ein Mapping eine besonders gute Precision bekannt

```
PROCEDURE HubMatch ($map1, $map2)
  $result := compose($map1,inverse($map2));
  RETURN $result;
END
```

Abbildung 9.7: iFuice-Repräsentation der Match-Strategie Hub

ist, so dass die entsprechenden Korrespondenzen bevorzugt behandelt werden sollen. Das bedeutet für eine Korrespondenz (a_1, b_1) des bevorzugten Mappings map_1, dass keine Korrespondenz der Art $(a_1, b_i) \notin map_1$ im resultierenden Mapping auftreten darf. Die implizite Annahme zur Anwendung der Prefer-Strategie besteht somit darin, dass, wenn das bevorzugte Mapping eine oder mehrere Korrespondenzen für ein Domain-Objekt besitzt, dies bereits alle korrekten Korrespondenzen sind. Abbildung 9.6 zeigt die entsprechende Prozedur. Mittels des Operators `diff` werden diejenigen Domain-Objekte aus map_2 bestimmt, für die es keine Korrespondenz in map_1 gibt. Nur diese Korrespondenzen aus map_2 werden durch `union` dem ersten Mapping hinzugefügt.

Die Prefer-Strategie kann z. B. für ein Publikations-Same-Mapping zwischen GS und ACM angewendet werden. Für einige Einträge liefert GS bereits Links zu den korrespondierenden ACM-Publikationen in sehr guter Qualität. Dieses Mapping wird mittels Prefer durch die Ergebnisse eines Attribut-Matchers vervollständigt.

```
$map1 := map ($GSPub, GS.ExtractLink2ACM);
$map1 := attrMatch ($GSPub, $ACMPub, Trigram,
           "[GS.title]", "[ACM.name]");
$same := exec (PreferMatch, $map1, $map2);
```

Match-Strategie: Hub

Die dritte Match-Strategie umfasst die Nutzung einer Hub-Datenquelle, d. h. einer Datenquelle, die zu vielen anderen Datenquellen Same-Mappings aufweist. Die bisherigen Strategien kombinierten Same-Mappings des gleichen Typs $LDS_A \times LDS_B$, wohingegen die Strategie Hub aus zwei Mappings der Typen $LDS_A \times LDS_X$ und $LDS_B \times LDS_X$ ein Same-Mapping zwischen LDS_A und LDS_B bestimmt. Die Ausführung der Hub-Strategie (siehe Abbildung 9.7) umfasst im Wesentlichen die Anwendung des `compose`-Operators auf die beiden Eingabe-Mappings. Die Verwendung von Hub ist eine effektive Wiederverwendung von Mappings, da durch diese Strategie ein Same-Mapping eines anderen Typs als die Eingabe-Mappings ermittelt wird.

Abbildung 9.8 zeigt ein mögliches Szenario für die bibliografische Domäne. Durch die Bildung von vier Mappings zur Hub-Datenquelle DBLP ist es nun möglich, die sechs Same-Mappings zwischen den jeweiligen Datenquellen außer DBPL zu bestimmen.

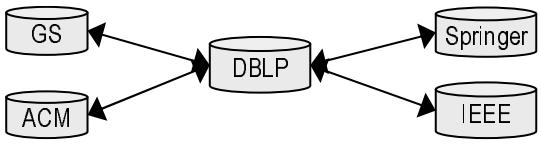


Abbildung 9.8: Beispiel für eine Mapping-basierte Hub-Struktur

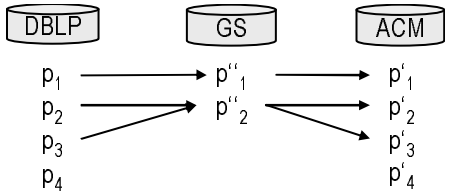


Abbildung 9.9: Beispiel für die Anwendung der Hub-Strategie

Das anschließende Beispiel illustriert die Verwendung der Hub-Datenquelle DBLP zur Bestimmung der Mappings zwischen ACM und GS.

```
$map1 := attrMatch ($ACMPub, $DBLPPub, Trigram,
           "[ACM.name]", "[DBLP.title]");
$map2 := attrMatch ($GSPub, $DBLPPub, Trigram,
           "[GS.title]", "[DBLP.title]");
$same := exec (HubMatch, $map1, $map2);
```

Die Qualität des `compose`-Ergebnisses wird durch die Sauberkeit und Vollständigkeit der verwendeten Quellen und Mappings beeinflusst. So können Same-Mappings für eine Objektinstanz Korrespondenzen zu verschiedenen Instanzen (Duplikaten) einer anderen Datenquelle besitzen. Auf der anderen Seite kann die Vollständigkeit der Quellen variieren, so dass Instanzen keine korrespondierenden Objekte in anderen Datenquellen haben können. Abbildung 9.9 illustriert den Einfluss von Sauberkeit und Vollständigkeit an einem Beispiel mit Google Scholar als Hub-Datenquelle. Dabei seien p_2 und p_3 gleichnamige Konferenz- und Journal-Papiere (vergleiche [36] und [37]). ACM und DBLP unterscheiden zwischen beiden Publikationen korrekt, wohingegen GS als automatisch erstellte Datenquelle beide Papiere unter einem Eintrag subsumiert. Dadurch entstehen durch die Komposition neben den korrekten Korrespondenzen (p_2, p_2'') und (p_3, p_3'') auch die falschen Korrespondenzen (p_2, p_3'') und (p_3, p_2''), was zu einer Verminderung der Precision führt. Zusätzlich illustriert Publikation p_4 die Reduzierung des Recalls bei Verwendung einer unvollständigen Hub-Datenquelle. Im Beispiel der Abbildung 9.9 habe Google Scholar keine korrespondierende Objektinstanz p_4'. Dadurch kann — obwohl p_4 und p_4'' in DBLP bzw. ACM vorhanden sind — keine Korrespondenz (p_4, p_4'') durch Komposition ermittelt werden.

Die angeführten Probleme lassen sich im Wesentlichen durch zwei Ansätze beheben.

```
PROCEDURE NeighborhoodMatch ($Asso1, $Same, $Asso2)
  $Temp := compose (inverse($Asso1), $Same, Right, Max);
  $Result := compose ($Temp, $Asso2, Left, Dice);
  RETURN $Result;
END
```

Abbildung 9.10: iFuice-Repräsentation der Match-Strategie Neighborhood

Zunächst muss innerhalb der Domäne eine möglichst saubere und vollständige Datenquelle als Hub definiert werden. In der bibliografischen Domäne erfüllt DBLP – zumindest für die in Kapitel 8 betrachten Venues bzw. Publikationen – diese Anforderungen (siehe Abbildung 9.8). Zweitens kann das durch die Hub-Strategie erzielte Match-Ergebnis durch weitere Matcher verfeinert werden. So kann z. B. für Instanzen, die im Ergebnis der Hub-Strategie keine Korrespondenzen besitzen, zusätzlich ein Attribut-Matcher ausgeführt werden. Hub- und Attribut-Match-Ergebnis können dann durch die Merge- oder Prefer-Strategie kombiniert werden, so dass fehlende Korrespondenzen der Hub-Strategie durch den Attribut-Matcher ergänzt werden.

9.4.2 Verwendung von Assoziations-Mappings

Die bisherigen Strategien kombinierten lediglich Same-Mappings um neue Same-Mappings zu erzeugen. Im Folgenden werden zwei Match-Strategien – *Neighborhood* und *Self-Similarity* – vorgestellt, welche auch Assoziations-Mappings verwenden.

Match-Strategie: Neighborhood

Ausgehend von einem Same-Mapping des Typs $LDS_{A_1} \times LDS_{A_2}$ und zwei Assoziations-Mappings $LDS_{A_1} \times LDS_{B_1}$ und $LDS_{A_2} \times LDS_{B_2}$ des gleichen semantischen Mapping-Typs ermittelt die Neighborhood-Strategie ein Same-Mapping zwischen LDS_{B_1} und LDS_{B_2}. Die zu Grunde liegende Annahme ist, dass aus der Gleichheit assoziierter Objekte a_{1_i} bzw. a_{2_i} auf die Gleichheit zwischen Objekten b_1 und b_2 geschlossen wird. Dadurch ist es möglich, ein Same-Mapping zwischen Objektinstanzen ohne Verwendung der Attribute zu erstellen.

Die in Abbildung 9.10 dargestellte iFuice-Prozedur der Neighborhood-Strategie besteht im Wesentlichen aus der schrittweisen Komposition der drei Mappings, wobei das erste Assoziations-Mapping zunächst invertiert wird. Im ersten Schritt entsteht ein Mapping zwischen LDS_{B_1} und LDS_{A_2}, deren Konfidenzwerte sich aus dem Maximum der Konfidenzwerte des Same-Mappings ergeben ($\vec{h} = Right$ und $\vec{v} = Max$). Anschließend wird dieses temporäre Mapping mit dem zweiten Assoziations-Mapping komponiert und mit Hilfe der Dice-Funktion werden Korrespondenzen, die über

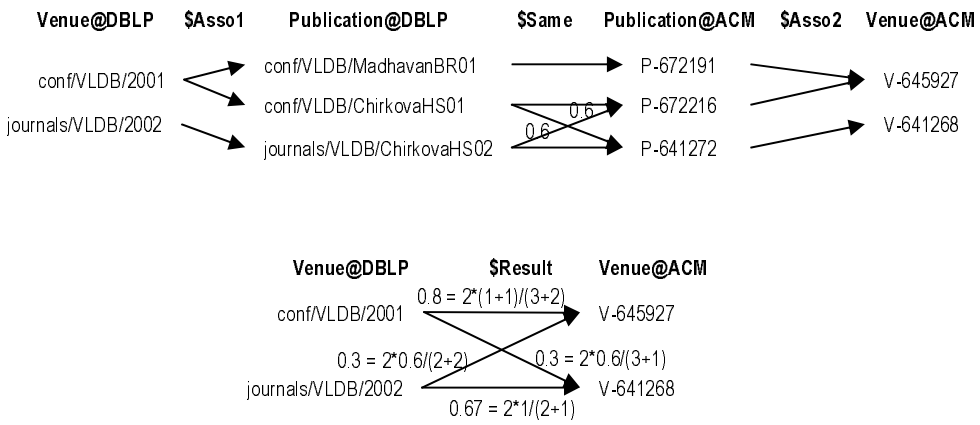


Abbildung 9.11: Beispiel für die Anwendung der Neighborhood-Strategie mit Assoziations-Mapping der Kardinalität $1:N$

mehrere Pfade erreicht werden, bevorzugt. Dies entspricht der Idee, dass eine große gemeinsame Nachbarschaft, d. h. eine große Menge gleicher assoziierter Objekte, zu einer hohen Ähnlichkeit zwischen den Objekten führt.

Abbildung 9.11 zeigt ein Beispiel, das ein Venue-Same-Mapping anhand eines Publikations-Same-Mappings mit folgendem Skript ermittelt.

```
$map1 := attrMatch ($DBLPPub, $ACMPub, Trigram,
          "[DBLP.title]", "[ACM.name]");
$map2 := attrMatch ($DBLPPub, $ACMPub, Trigram,
          "[DBLP.year]", "[ACM.year]");
$samePub := exec (MergeMatch, $map1, $map2);
$asso1 := map ($DBLPPub, DBLP.PubVenue);
$asso2 := map ($ACMPub, ACM.PubVenue);
$sameConf := exec (NeighborhoodMatch, $asso1, $samePub, $asso2);
```

Das verwendete Same-Mapping sei das Ergebnis der Kombination zweier Attribut-Matcher bzgl. Titel und Jahr der Publikationen. Daher ergeben sich im Beispiel für zwei Korrespondenzen Konfidenzwerte von 0.6, da die Jahreszahlen unterschiedlich sind. Für alle anderen Korrespondenzen sei eine Konfidenz von 1 angenommen, die aus Gründen der Übersichtlichkeit nicht in Abbildung 9.11 angegeben ist. Die Anwendung des Neighborhood-Matchers erzeugt nun vier Korrespondenzen, die beide DBLP- mit jeweils beiden ACM-Venues verknüpft, da es für jede Kombination mindestens einen Kompositionspfad gibt. Nach Anwendung des ersten `compose`-Operators ergibt sich das in Abbildung 9.3 (Seite 184) dargestellte Szenario für die zweite Ausführung von `compose`. Durch die dort angegebene Berechnung der Konfidenzen ergibt sich ein sehr gutes Ergebnis, da die beiden korrekten Korrespondenzen deutlich höhere Konfidenzwerte (0.8 bzw. 0.67) haben als die beiden anderen (jeweils 0.3).

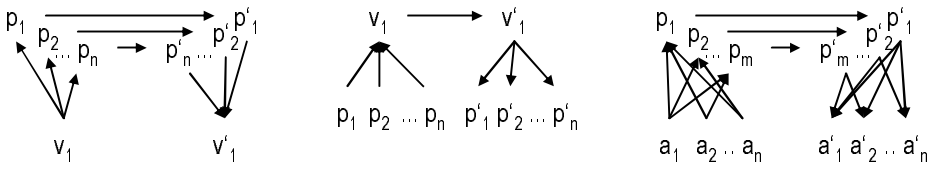
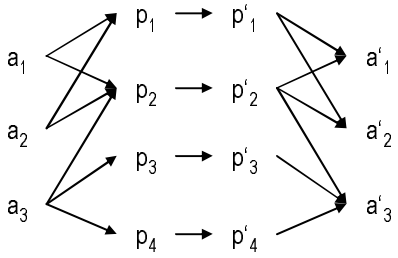


Abbildung 9.12: Einfluss der Kardinalität des Assoziations-Mappings bei Anwendung der Neighborhood-Strategie



Neighborhood		
A	A'	Conf
a_1	a'_1	1
a_1	a'_2	1
a_1	a'_3	0.5
a_2	a'_1	1
a_2	a'_2	1
a_2	a'_3	0.5
a_3	a'_1	0.3
a_3	a'_2	0.3
a_3	a'_3	1

Abbildung 9.13: Beispiel für die Anwendung der Neighborhood-Strategie mit Assoziations-Mapping der Kardinalität $N : M$

Der Grund liegt in der Berechnung mit Hilfe der Dice-Funktion, welche in diesem Fall für zwei Venues die relative Überlappung der gleichen Publikationen bestimmt. Durch eine anschließende Selektion mit einem Schwellwert (z. B. 0.5) kann also im Beispiel das korrekte Ergebnis erzielt werden.

Das im gegebenen Beispiel verwendete Assoziations-Mapping hat den semantischen Mapping-Typ „Publikation eines Venues“, dessen Kardinalität $1 : N$ ist, d. h. einem Venue sind N Publikationen und jeder Publikation genau ein Venue zugeordnet. In diesem Fall liefert der Neigborhood-Matcher ein sehr gutes Ergebnis. Die Abhängigkeit der Ergebnisqualität von der Kardinalität illustriert Abbildung 9.12. Der Fall $1 : N$ liefert sehr gute Ergebnisse insbesondere für große N, da auch bei fehlerhaften Korrespondenzen im verwendeten (Publikations-)Same-Mapping durch die relative Häufigkeit der Kompositionspfade die korrekten (Venue-)Korrespondenzen ermittelt werden. Der umgekehrte Fall $N : 1$ erzeugt hingegen Korrespondenzen zwischen zwei Publikationen, wenn sie zum gleichen Venue gehören. Da jede Publikation einem Venue zugeordnet ist und jedes Venue mehrere Publikation besitzt, liefert die Berechnung der Konfidenzwerte keine hinreichenden Ergebnisse. Dennoch ist das Ergebnis der Neighborhood-Strategie nützlich zur Weiterverarbeitung, da es die möglichen Korrespondenzen im „Endergebnis“ begrenzt. Wird das resultierende Mapping als Eingabe für einen attributbasierten Matcher verwendet, vergleicht dieser nur noch Publikationen des gleichen Venues. Dadurch wird sowohl die Komplexität des Attribut-Matchings reduziert, als auch die Precision erhöht.

```
PROCEDURE SelfSimMatch ($Asso1)
  $SelfSim := compose(inverse($Asso1), $Asso1, Max, Relative);
  $Result := diff ($SelfSim, map (range($Asso1)));
  RETURN $Result;
END
```

Abbildung 9.14: iFuice-Repräsentation der Match-Strategie Self-Similarity

Der letzte Fall $N : M$ dient ebenso wie $N : 1$ zur Reduktion der möglichen Korrespondenzen. Zusätzlich können die berechneten Konfidenzwerte bereits zur Identifikation der korrekten Korrespondenzen dienen. Abbildung 9.13 verdeutlicht dies am Beispiel der Assoziation „Autoren der Publikation“. Die Autoren a_1 und a_2 lassen sich durch die Neighborhood-Strategie nicht unterscheiden, da sie eine identische Nachbarschaft, d. h. die gleichen Publikationen, besitzen. Demgegenüber lässt sich (a_3, a_3') z. B. mit einer Best-1-Selektion identifizieren, obgleich a_3 auch gemeinsame Publikationen mit a_1 und a_2 hat.

Match-Strategie: Self-Similarity

Ein Spezialfall der Neighborhood-Strategie ist Self-Similarity, bei der nach Duplikaten innerhalb einer Datenquelle gesucht wird. Es genügt daher die Angabe eines Assoziations-Mappings vom Typ $LDS_A \times LDS_B$, um ein sogenanntes Self-Mapping, d. h. ein Same-Mapping zwischen Objekten derselben LDS, zu bestimmen.

Abbildung 9.14 zeigt die zugehörige Prozedur, welche nur einen Aufruf des `compose`-Operators benötigt. Im Gegensatz zu Neighborhood müssen jedoch triviale Duplikate, d. h. Korrespondenzen zwischen denselben Objekten, herausgefiltert werden. Dies wird mit Hilfe von `diff` realisiert, welches Korrespondenzen des Identitäts-Mappings vom berechneten Match-Ergebnis entfernt.

Das nachfolgende Beispiel zeigt die Verwendung der Strategie Self-Similarity für die Duplikaterkennung von Autoren innerhalb DBLP. Dabei wird als Assoziations-Mapping die Koautorschaft verwendet. Besitzen also zwei Autoren eine ähnliche Menge von Koautoren und haben sie zusätzlich einen ähnlichen Namen, besteht der Verdacht auf Duplikate. Obwohl DBLP eine sehr hohe Datenqualität auf Grund der manuellen Bearbeitung besitzt, treten solche Fälle durch Namensänderung (z. B. Alon Levy vs. Alon Halevy) oder durch verschiedene Schreibweisen (z. B. Agathoniki Trigoni vs. Niki Trigoni) auf.

```
$map1 := exec (SelfSimilarity, $DBLPCoAuth);
$map2 := attrMatch ($map1, Trigram, "[DBLP.name]", "[DBLP.name]");
$self := exec (MergeMatch, $map1, $map2);
```

9.5 Zusammenfassung

In diesem Kapitel wurde mit dem MOMA-Framework ein flexibles Mapping-basiertes System für das Object Matching vorgestellt. MOMA unterstützt die Ausführung sogenannter Match-Workflows, die neben der Verwendung von Matchern und anderen Workflows insbesondere Möglichkeiten zur Kombination von Mappings liefern. Dazu wurden entsprechende Kombinations- und Selektions-Operatoren vorgestellt. Ein wichtiges Element für die Flexibilität des MOMA-Ansatzes sind sogenannte Match-Strategien, die als parametrisierte Sub-Workflows in anderen Match-Workflows angewendet werden können und im folgenden Kapitel evaluiert werden.

10 Evaluation

In diesem Kapitel wird das MOMA-Framework am Beispiel realer bibliografischer Datenquellen evaluiert. Das Ziel der Evaluation ist es, die Eigenschaften der vorgestellten Match-Strategien zu charakterisieren. Es geht also nicht primär darum, das perfekte Match-Ergebnis durch Optimierung verschiedener Parameter (z. B. Schwellwert bei Selektion) zu erreichen. Vielmehr soll die Nutzbarkeit der Strategien nach verschiedenen Kriterien beurteilt werden. Nach der Vorstellung der verwendeten Daten sowie der Evaluationsmaße (Abschnitt 10.1) werden zunächst die unterschiedlichen Selektionsstrategien evaluiert (Abschnitt 10.2). Anschließend erfolgt die Evaluation der in Abschnitt 9.4 vorgestellten Match-Strategien (Abschnitte 10.3 und 10.4).

10.1 Aufbau der Evaluation

Die Evaluation umfasst die bereits in Kapitel 8 vorgestellten Datenquellen DBLP Bibliograpy, ACM Digital Library und Google Scholar. Sie stellen jeweils Informationen über Publikationen, Autoren und Venues bereit. Dabei variieren sie jedoch hinsichtlich Datenqualität und Vollständigkeit (siehe Abschnitt 8.1.2).

Für die Evaluation wurden die Konferenzen VLDB und SIGMOD sowie die Journals VLDB Journal, TODS und SIGMOD Record für den Zeitraum 1994-2003 betrachtet. Tabelle 10.1 zeigt die Anzahl der zugehörigen Venues, Autoren und Publikationen für die drei betrachteten Datenquellen. Die Tabelle zeigt, dass die Evaluation insbesondere bei Google Scholar eine deutlich größere Datenmenge verwendet als

	Venues	Publikationen	Autoren
DBLP	130	2.616	3.319
ACM Digital Library (ACM)	128	2.294	3.547
Google Scholar (GS)	–	64.263	(81.296)

Tabelle 10.1: Anzahl der in der Evalution betrachteten Objektinstanzen

standardisierte Evaluationsdaten wie z.B. Cora[62]. Der Cora-Datensatz enthält 1295 Referenzen, die manuell insgesamt 112 verschiedenen Publikationen zugeordnet wurden, und wurde z.B. [52] in verwendet.

Das Ziel der Evaluation ist die automatische Erstellung von Same-Mappings zwischen Instanzen aller drei Objekttypen. Auf Grund der unvollständigen und sehr uneinheitlichen Repräsentation von Venues wird für Google Scholar kein Same-Mapping für Venues gebildet. Des Weiteren repräsentiert Google Scholar Autorennamen nur durch die Anfangsbuchstaben der Vornamen und dem Familiennamen, wodurch mehrere Personen (z.B. Alan Smith, Adam Smith und Andrew Smith) innerhalb Google Scholar als gleich (im Beispiel: A-Smith) angesehen werden. Dadurch ist eine Evaluation bzgl. eines generierten Same-Mappings für Autoren nicht sinnvoll. Die Anzahl der Autoren ist dennoch in Tabelle 10.1 in Klammern angegeben, da Google-Scholar-Autoren bei der Neighborhood-Match-Strategie (siehe Abschnitt 10.4) verwendet werden.

Zur Evaluation werden die Standardmaße Precision, Recall und F-Measure – bezogen auf die zuvor manuell erstellten perfekten Mappings – verwendet. Bezeichnet map_E ein zu evaluierendes Mapping und map_P das zugehörige perfekte Mapping, so sind die drei Evaluationsmaße wie folgt definiert [180]:

Precision: $= |map_E \cap map_P|/|map_E|$ ist der Anteil der korrekt ermittelten Korrespondenzen im Vergleich zu allen ermittelten Korrespondenzen und bezeichnet die „Genauigkeit“ des Ergebnisses.

Recall: $= |map_E \cap map_P|/|map_P|$ ist der Anteil der korrekt ermittelten Korrespondenzen im Vergleich zu allen korrekten Korrespondenzen und bezeichnet die „Vollständigkeit“ des Ergebnisses.

F-Measure: $= 2 \cdot Precision \cdot Recall/(Precision + Recall)$ ist das harmonische Mittel von Precision und Recall. Es reflektiert in einem Wert die „Gesamtqualität“ des Ergebnisses.

[62] http://www.cs.umass.edu/~mccallum/data/cora-refs.tar.gz

	60%	70%	80%	90%	100%	Best-1	Best-1+Δ 0.25	
							50%	70%
Precision	55,6%	72,5%	81,1%	79,0%	52,2%	70,2%	66,1%	76,3%
Recall	97,5%	92,4%	81,6%	54,8%	12,2%	48,1%	92,2%	90,3%
F-Measure	70,8%	81,2%	81,3%	64,7%	23,6%	57,1%	77,1%	82,7%

Tabelle 10.2: Vergleich der Selektionsstrategien für Attribut-Matcher zwischen DBLP- und GS-Publikationen

10.2 Selektionsstrategien

Zunächst wird der Einfluss der Selektionsstrategien evaluiert. Dazu wurde mittels eines Attribut-Matchers (Ähnlichkeitsfunktion: Trigram) ein Same-Mapping zwischen DBLP- und GS-Publikationen auf Basis der Publikationstitel erstellt. Tabelle 10.2 zeigt die Mapping-Qualität nach Filterung durch verschiedene Selektionsstrategien.

Bei der schwellwertbasierten Strategie zeigt sich mit zunehmendem Schwellwert eine Steigerung der Precision auf Kosten des Recalls. Der maximale F-Measure wird dabei im Bereich um 80% erreicht. Interessanterweise ergibt sich bei Schwellwerten ab 90% wieder eine niedrigere Precision. Es werden bei höheren Schwellwerten zwar weniger falsche Korrespondenzen gefunden, aber auch bei 100% gibt es eine signifikante Zahl von Publikations-Korrespondenzen mit gleichem Titel, die nicht gleich sind, z. B. gleichnamige Journal-Rubriken aus verschiedenen Jahren. Dadurch steigt bei hohen Schwellwerten der relative Anteil der falschen Korrespondenzen, was zu einer verminderten Precision führt.

Das Ergebnis der Strategie Best-1 zeigt die Besonderheit einer „relativen“ Selektionsstrategie, die für jedes Domain-Objekt diejenige(n) Korrespondenz(en) mit der höchsten Konfidenz wählt. Diese Strategie liefert im Beispiel von Tabelle 10.2 einen schlechten Recall, da GS Duplikate enthält und somit für eine DBLP-Publikationen i. Allg. mehrere Korrespondenzen im perfekten Ergebnis enthalten sind. Des Weiteren unterstellt Best-1 das Vorhandensein eines gleichen Objektes in der Range-Datenquelle. Besitzt Google Scholar für einen DBLP-Publikation keinen Eintrag, so selektiert die Best-1-Strategie falsche Korrespondenzen. Dem ersten Problem tritt Best-Delta entgegen. Es selektiert nicht nur die beste Korrespondenz, sondern auch alle Korrespondenzen deren Ähnlichkeitswerte nicht mehr als um einen definierten Wert abweichen (im Beispiel 0,25). Zusätzlich wurde – um falschen Korrespondenzen vorzubeugen – die Best-1+Delta-Strategie mit einem Schwellwert (50% bzw. 70%) kombiniert, um bei fehlenden GS-Publikationen keine Korrespondenzen zu ermitteln. Dadurch können Recall und Precision deutlich gesteigert werden.

Sofern nicht anders angegeben, beziehen sich die folgenden Ergebnisse der Evaluation der Match-Strategien jeweils auf eine Selektion mit einem Schwellwert von 80%.

	Titel	Autoren	Jahr	Merge
Precision	86,7%	38,0%	0,4%	97,3%
Recall	97,7%	87,9%	100,0%	93,9%
F-Measure	91,9%	53,1%	0,8%	95,5%

Tabelle 10.3: Match-Ergebnis für Publikationen DBLP-ACM mit Hilfe mehrerer Attribut-Matcher sowie deren Kombination durch die Merge-Strategie

	Attribut	Link	Prefer
Precision	86,7%	97,7%	89,9%
Recall	81,7%	21,6%	81,3%
F-Measure	84,1%	35,3%	85,4%

Tabelle 10.4: Match-Ergebnis für Publikationen GS-ACM mit Hilfe eines Attribut-Matchers, Verwendung von Hyperlinks sowie deren Kombination durch die Prefer-Strategie

10.3 Strategien Merge, Prefer und Hub

10.3.1 Merge- und Prefer-Strategie

In einem Experiment wurde die Merge-Strategie für drei verschiedene Attribut-Matcher angewendet, um ein Same-Mapping zwischen ACM- und DBLP-Publikationen zu erstellen. Die ersten beiden Matcher ermitteln eine String-Ähnlichkeit zwischen den Titeln bzw. der Liste der Autoren (repräsentiert als kommaseparierter String). Der dritte Matcher vergleicht lediglich die Jahreszahl der Publikationen auf Gleichheit. Tabelle 10.3 zeigt Precision, Recall und F-Measure für alle drei einzelnen Matcher und für das Ergebnis der Merge-Strategie (mit Average als Konfidenzfunktion). Offenbar erzeugt der Titel-Matcher die besten Ergebnisse, welche jedoch durch die Kombination mit den anderen Match-Ergebnissen noch weiter verbessert werden können. Es zeigt sich daher, dass die Kombination mehrerer unabhängiger Match-Verfahren die Gesamtqualität des Match-Ergebnisses steigern kann.

Für die Strategie Prefer wurden zwei Mappings zwischen GS und ACM miteinander kombiniert. Für das erste Mapping wurden die von GS bereitgestellten Links zu ACM verwendet, die eine hohe Qualität aufweisen, aber leider nur für einen Teil der GS-Einträge verfügbar sind. Das zweite Mapping ist das Ergebnis eines Attribut-Matchers (Trigram) der Titel. Tabelle 10.4 zeigt die Ergebnisse der beiden Mappings sowie des kombinierten Ergebnisses. Im Vergleich zum Attribut-Match-Ergebnis zeigt sich, dass auf Grund einer deutlichen Steigerung der Precision bei fast gleichem Recall der F-Measure gesteigert werden konnte. Besonders in den Fäl-

Mapping	DBLP - GS			DBLP - ACM			GS - ACM		
Matcher	Direkt	ACM	Merge	Direkt	GS	Merge	Direkt	DBLP	Merge
Precision	81,1%	99,0%	81,0%	86,7%	94,9%	86,1%	97,7%	87,0%	86,6%
Recall	81,6%	20,4%	81,6%	97,7%	48,0%	97,8%	21,6%	80,9%	81,0%
F-Measure	81,3%	33,9%	81,3%	91,9%	63,7%	91,6%	35,3%	83,9%	83,7%

Tabelle 10.5: Match-Ergebnis für Publikations-Same-Mappings mit Hilfe verschiedener Kompositionspfade

len, bei denen der Attribut-Matcher für eine GS-Publikation mehrere ACM-Publikationen ermittelt, kann durch die Bevorzugung des Link-Mappings die Precision gesteigert werden. Analog zur Merge-Strategie zeigt sich auch hier, dass durch die Kombination mehrerer Mappings die Match-Qualität gesteigert werden kann.

10.3.2 Hub-Strategie

In diesem Experiment soll die Nützlichkeit der Hub-Strategie zur Bildung neuer Same-Mappings evaluiert werden. Die Wiederverwendung bestehender Same-Mappings durch Komposition ist besonders effektiv, da die Komposition effizient durch einen Join der Mapping-Tabellen bzgl. der Objekt-IDs realisiert werden kann. Ausgangspunkt für das Experiment sind drei Publikations-Same-Mappings zwischen allen drei Datenquellen.

Die Mappings DBLP-ACM und DBLP-GS wurden mit Hilfe eines Attribut-Matchers anhand des Titels bestimmt. Das Mapping GS-ACM wurde wie bei der Evaluation von Prefer (Abschnitt 10.3.1) von den GS-Ergebnisseiten extrahiert. Tabelle 10.5 zeigt die Ergebnisse der direkten Mappings sowie unter Verwendung der jeweils dritten Datenquelle als Hub. Zusätzlich wurden beide Mappings mittels der Match-Strategie Merge (Konfidenzfunktion: Average) kombiniert. In allen Fällen wurden nur Korrespondenzen mit einem Ähnlichkeitswert von mindestens 80% betrachtet.

Bei der ersten Beobachtung der Ergebnisse zeigt sich, dass im Fall GS-ACM die Komposition via DBLP bessere Ergebnisse erzielt als das direkte Mapping. Der Grund ist der schlechte Recall des direkten Mappings (21,6%), da Google Scholar nur für einen kleinen Teil seiner Ergebnisse Links zu ACM anbietet. Dadurch wird auch in den beiden anderen Fällen das Ergebnis der Kompositions-Mappings verschlechtert, so dass bei DBLP-ACM und DBLP-GS das direkte Mapping besser abschneidet. Zusätzlich beeinträchtigt die Tatsache, dass ACM keine VLDB-Publikationen der Jahrgänge 2002 und 2003 enthält, die Qualität des Kompositionspfades DBLP-ACM-GS (Abbildung 9.9 auf Seite 189 illustriert diesen Effekt). Die Ergebnisse zeigen daher, dass der Wahl der Hub-Datenquelle eine große Bedeutung zukommt. Sie sollte von großer Qualität hinsichtlich Vollständigkeit und Duplikatfreiheit sein.

Leider kennen Nutzer i. Allg. die Qualität der Datenquellen nicht, so dass die Wahl

der Hub-Datenquellen schwer fällt. Glücklicherweise kann das Risiko einer schlechten Wahl dahingehend reduziert werden, indem die direkten Mappings mit den Kompositions-Mappings kombiniert werden. Wie oben beschrieben, wurde in diesem Experiment die Match-Strategie Merge verwendet. Wie Tabelle 10.5 zeigt, erreicht das Merge-Mapping in allen drei Fällen die Qualität des besseren Mappings (direkt oder komponiert).

10.4 Strategien Neighborhood und Self-Similarity

10.4.1 Neighborhood-Strategie

In diesem Abschnitt wird die Match-Strategie Neighborhood für alle drei Kardinalitäten evaluiert. Dazu wird zwischen DBLP und ACM ein Venue-Same-Mapping mit Hilfe eines Publikations-Same-Mappings (1:N) bzw. ein Publikations-Same-Mapping mit Hilfe eines Venue-Same-Mappings (N:1) erstellt. Für den letzten Fall (N:M) werden ein Autoren-Same-Mapping zwischen DBLP und ACM mit Hilfe der Publikationen sowie ein Publikations-Same-Mappings zwischen DBLP und GS bzw. GS und ACM mit Hilfe der jeweiligen Autoren bestimmt.

Kardinalität 1:N am Beispiel Venue-Publikation

Ausgangspunkt dieses Experiments ist ein Publikations-Same-Mapping DBLP-ACM, welches durch einen Attribut-Matcher erstellt wurde. Zusätzlich wird in beiden Datenquellen das Assoziations-Mapping Venue-Publikation verwendet. Der Einsatz der Neighborhood-Strategie ist dabei insbesondere für Konferenzen sinnvoll, da sie eine große Heterogenität in ihren Attributwerten aufweisen, so dass z. B. die gleiche Konferenz sowohl als „VLDB 2002“ als auch als „28$^{\text{th}}$ International Conference on Very Large Databases“ bezeichnet werden kann.

Tabelle 10.6 zeigt die Effektivität der Strategie für diesen Fall. Ohne Kombination mit weiteren Matchern, d. h. insbesondere ohne Verwendung der Attribute, erreicht die Strategie einen durchschnittlichen F-Measure von bis zu 98,8%. Tabelle 10.6 unterscheidet das Match-Ergebnis zusätzlich noch für Konferenzen und Journals, da ein Konferenzband i. Allg. deutlich mehr Publikationen (ca. 60-120) als eine Journal-Ausgabe (2-26) besitzt. Dieser Umstand führt dazu, dass die schwellwertbasierte Selektion (80% und 50%) für Konferenzen ein perfektes Match-Ergebnis liefert, da durch die große Nachbarschaft kleine Fehler im verwendeten Publikations-Same-Mapping sogar bei einem geringen Schwellwert von 50% ausgeglichen werden. Auch die Best-1-Selektion liefert alle korrekten Korrespondenzen (Recall=100%), ermittelt jedoch für VLDB 2002 und 2003 falsche Korrespondenzen, da diese nicht in ACM auftreten. Durch fehlerhafte Korrespondenzen im Publikations-Mapping, z. B.

Matcher	Neighborhood (Publikation)		
Selektion	80%	50%	Best-1
	Konferenzen		
Precision	100%	100%	94,7%
Recall	100%	100%	100%
F-Measure	100%	100%	97,3%
	Journals		
Precision	100%	99,0%	98,2%
Recall	62,7%	86,4%	100%
F-Measure	77,1%	92,2%	99,1%
	Gesamt		
F-Measure	80,9%	93,4%	98,8%

Tabelle 10.6: Neighborhood-Match-Ergebnis für Venues DBLP-ACM auf Basis von Publikationen (1:N)

auf Grund gleichnamiger Publikationen in VLDB und VLDB Journal, ergeben sich dennoch ausgehend von den VLDB-Konferenzen bei DBLP Kompositionspfade zu (falschen) ACM-Venues. Auf der anderen Seite ergeben sich für die schwellwertbasierte Selektion deutlich schlechtere Ergebnisse für Journals, da diese nur zwischen 2-26 Publikationen pro Ausgabe besitzen und dadurch Fehler im zu Grunde liegenden Publikations-Same-Mapping einen größeren Einfluss haben, die insbesondere den Recall stark reduzieren. Demgegenüber schneidet die Best-1-Selektion sehr gut ab, da ACM alle betrachteten Journals enthält.

Kardinalität N:1 am Beispiel Publikation-Venue

In diesem Experiment wurde das im vorigen Abschnitt ermittelte Venue-Mapping mit 98,8% F-Measure verwendet, um ein Publikations-Mapping abzuleiten. Im Unterschied zum 1:N-Fall reicht die Anwendung der Neighborhood-Strategie nicht aus, da diese bereits eine Korrespondenz zwischen zwei Publikationen erstellt, wenn sie dem gleichen Venue angehören. Daher ist das Ziel, ein bestehendes Same-Mapping, das durch einen Attribut-Matcher anhand des Publikationstitels ermittelt wurde, durch Kombination (Merge-Strategie) zu verbessern.

Tabelle 10.7 zeigt die entsprechenden Ergebnisse, d. h. sowohl für den Attribut-Matcher, die Neighborhood-Strategie als auch für deren Kombination mittels Merge-Strategie. Wie erwartet liefert die Neighborhood-Strategie einen perfekten Recall – jedoch eine indiskutable Precision. Trotz des so erhaltenen kleinen F-Measures liefert die Kombination beider Mappings ein hervorragendes Ergebnis mit einem F-Measure von 98,6%. Auch hier zeigt Tabelle 10.7 die Ergebnisse getrennt für Konferenzen und Journals und illustriert, dass die Verbesserung durch die Mapping-

Matcher	Attribut (Titel)	Neighborhood (Venue)	Merge
	Publikationen von Konferenzen		
Precision	96,7%	1,2%	99,2%
Recall	99,8%	100%	98,8%
F-Measure	97,7%	2,4%	99,0%
	Publikationen von Journals		
Precision	72,8%	6,5%	99,7%
Recall	95,9%	100%	95,9%
F-Measure	82,8%	12,2%	97,8%
	Gesamt		
F-Measure	91,9%	3,36%	98,6%

Tabelle 10.7: Neighborhood-Match-Ergebnis für Publikationen DBLP-ACM auf Basis von Venues (N:1)

Kombination besonders groß für Journals ist, was auf zwei Gründe zurückzuführen ist. Zunächst haben insbesondere Journals wiederkehrende Rubriken (z. B. Editorials oder Interviews), was sich in ähnlichen oder gleichen Titel niederschlägt. Auf der anderen Seite ist (im Gegensatz zu 1:N) hier die kleine Nachbarschaft, d. h. die relativ wenigen Publikationen pro Ausgabe, von Vorteil, da sie die möglichen Korrespondenzen stark einschränkt. Dadurch können insbesondere die angesprochenen wiederkehrenden Rubriken differenziert werden.

Kardinalität N:M am Beispiel Autor-Publikation

Für den letzten Fall N:M wurde zunächst das Assoziations-Mapping zwischen Publikationen und Autoren bei DBLP und ACM verwendet, um ein Autoren-Same-Mapping zu bestimmen. Im Gegensatz zur Assoziation zwischen Publikation und Venue ist die Nachbarschaft relativ klein und variiert stark (im Durchschnitt ca. 3 Autoren mit Werten zwischen 1 und 27 Autoren pro Publikation).

Tabelle 10.8 (oben) zeigt das Ergebnis für den Attribut-Matcher (basierend auf dem Autorennamen), die Neighborhood-Strategie sowie deren Kombination (Merge mit Konfidenzfunktion *Avg-0*). Das Ergebnis des Attribut-Matchers wird im Wesentlichen durch die relativ kurzen Autorennamen beeinflusst, so dass kleine Variationen bereits stark ins Gewicht fallen (siehe auch Tabelle 10.9 im folgenden Abschnitt). Auch die Neighborhood-Strategie allein liefert kein zufriedenstellendes Ergebnis, was bereits in Abbildung 9.13 (Seite 192) illustriert wurde. Analog zu den bisherigen Experimenten verbessert auch hier die Kombination beider Mappings das Gesamtergebnis (F-Measure von 89,4% auf 96,9%).

Matcher (Autoren DBLP-ACM)	Attribut (Autorname)	Neighborhood (Publikation)	Merge
Precision	99,3%	24,8%	99,9%
Recall	81,3%	99,3%	94,0%
F-Measure	89,4%	39,7%	96,9%

Matcher (Publikationen DBLP-GS)	Attribut (Publikationstitel)	Neighborhood (Autor)	Merge
Precision	81,1%	15,2%	85,1%
Recall	81,6%	76,0%	92,9%
F-Measure	81,3%	25,4%	88,9%

Matcher (Publikationen GS-ACM)	Attribut (Publikationstitel)	Neighborhood (Autor)	Merge
Precision	86,7%	16,2%	84,6%
Recall	81,7%	75,6%	92,1%
F-Measure	84,1%	26,7%	88,2%

Tabelle 10.8: Neighborhood-Match-Ergebnis (N:M)

Zusätzlich wurden die Publikations-Same-Mappings zwischen DBLP und GS bzw. GS und ACM mit Hilfe der assoziierten Autoren verbessert. Das Autoren-Same-Mapping wurde wieder mit einem Attribut-Matcher erstellt, dessen Ergebnisse jedoch maßgeblich durch die starke Reduzierung der Namen bei GS (Familienname und der jeweils erste Buchstabe der Vornamen) beeinflusst wurde. Zusätzlich sind die Autorenlisten bei GS nicht immer vollständig, so dass (im Gegensatz zu DBLP und ACM) kein korrektes Assoziations-Mapping vorliegt. Daher wurde innerhalb der Neigborhood-Strategie die Gewichtsfunktion *Dice* durch *DiceRight* bei DBLP-GS (bzw. *DiceLeft* bei GS-ACM) ersetzt, um den Einfluss fehlender GS-Autoren zu eliminieren. Trotz der schlechten Datenqualität bzgl. der Autoren gelang es durch die Kombination mit der Neighborhood-Strategie, das Ergebnis des Attribut-Matchers deutlich auf einen F-Measure von 88,9% bzw. 88,2% zu steigern (siehe Tabelle 10.8 Mitte und unten). Dieser Anstieg geht primär auf eine Erhöhung des Recalls zurück. Offenbar nutzen die Autoren wenig, um z. B. gleichnamige Publikationen zu unterscheiden. Sie ermöglichen aber auf der anderen Seite das Finden von Korrespondenzen für GS-Publikationen mit fehlerhaften Titeln, die u. a. durch den automatischen Extraktionsprozess oder durch fehlerhafte Referenzierungen in anderen Publikationen entstanden sind.

Autor	Autor	Name	Koautor		Merge
Catalina Fan	Catalina Wie	64%	100%	(4)	82%
Amir M. Zarkesh	Amir Zarkesh	84%	75%	(3)	79%
M. Barczyk	M. Barczyc	75%	73%	(10)	74%
Agathoniki Trigoni	Niki Trigoni	75%	67%	(4)	71%
Joe Chun-Hung Yuen	Joe Yuen	62%	67%	(2)	65%

Tabelle 10.9: Liste der Top-5-Kandidaten für Autorenduplikate innerhalb DBLP

10.4.2 Self-Similarity-Strategie

Um die Verwendung von MOMA zur Identifikation von Duplikaten innerhalb einer Datenquelle zu untersuchen, wurden in einem letzten Experiment Autorenduplikate innerhalb DBLP ermittelt. Obwohl DBLP eine sehr gute Datenqualität besitzt, enthält es dennoch für einige Autoren mehrere Einträge. Die Anwendung der Match-Strategie Self-Similarity verwendet das Koautoren-Assoziations-Mapping, so dass zwei Autoren als umso ähnlicher angesehen werden, je ähnlicher ihre Koautorschaft ist.

Tabelle 10.9 zeigt die Liste der fünf ähnlichsten Autoren. Neben der Ähnlichkeit bzgl. der Koautorschaft wurde auch mittels eines Attribut-Matchers die Namensähnlichkeit bestimmt. Die Ergebnisse in Tabelle 10.9 sind entsprechend der kombinierten Ähnlichkeit (Match-Strategie Merge mit Average-Konfidenzfunktion) sortiert. Zusätzlich wird bei der Koautorschaftsähnlichkeit in Klammern die Anzahl der Kompositionspfade ausgegeben, d. h. wie viele Koautoren übereinstimmen.

Das Trigoni-Duplikat wird bereits innerhalb der DBLP-Website[63] dahingehend behandelt (Stand: Oktober 2007), dass beide Namen in einer Autorenseite zusammengefasst werden. Die dargestellten Publikationen beinhalten jedoch die unterschiedlichen Schreibweisen. Die Autoren Catalina Fan und Catalina Wei sind ein Beispiel für die Schwierigkeit des Problems der Duplikaterkennung. Beide besitzen eine identische Liste von Koautoren und haben einen ähnlichen Namen (gleicher Vorname, unterschiedlicher Familienname). Man könnte vermuten, dass es sich um dieselbe Person handelt, die z. B. durch Eheschließung einen anderen Familiennamen angenommen hat. Eine hundertprozentige Sicherheit ist jedoch nicht gegeben. Dennoch zeigt das Ergebnis die Nützlichkeit von MOMA, da durch die Match-Strategie automatisch entsprechende Kandidatenpaare ermittelt wurden, die durch weitere (evt. manuelle) Recherche überprüft werden können.

[63] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/t/Trigoni:Agathoniki.html

	Venues	Publikationen	Autoren
DBLP-ACM	98,6%	98,8%	96,9%
DBLP-GS	–	88,9%	–
GS-ACM	–	88,2%	–

Tabelle 10.10: Zusammenfassung der Match-Ergebnisse (F-Measure)

10.5 Zusammenfassung

In diesem Kapitel wurde das MOMA-Framework am Beispiel realer Daten aus dem Bereich wissenschaftlicher Publikationen evaluiert. Tabelle 10.10 zeigt eine kurze Zusammenfassung der Ergebnisse. Dabei konnten bei allen Objekttypen sehr gute Ergebnisse für den Fall DBLP-ACM erzielt werden. Die Evaluation zeigt, dass durch die Kombination von Mappings eine Qualitätsverbesserung erreicht werden kann. Gleichzeitig wurde der flexible Einsatz von Match-Strategien dargestellt. So wurde z. B. die Neighborhood-Strategie in verschiedenen Fällen angewendet. Insbesondere die Kombination des generischen Attribut-Matchers mit der Mapping-Kombination durch Match-Strategien liefert sehr gute Ergebnisse für das Matching zwischen DBLP und ACM. Die Match-Qualität für Publikations-Same-Mappings mit Google Scholar ist leider deutlich schlechter, was im Wesentlichen an der schlechten Datenqualität von Google Scholar liegt. Insbesondere die vielen Datenfehler bei Google Scholar, u. a. fehlende und falsche Attributwerte, z. B. Autornamen im Titel oder umgekehrt, erschweren ein exaktes Object Matching. Dennoch konnte auch hier gezeigt werden, dass eine Mapping-Kombination, insbesondere unter Verwendung des Neighborhood-Matchers, zu einer Verbesserung der Match-Qualität führt.

Das Hauptanliegen der Evaluation bestand jedoch nicht in der Analyse der Match-Ergebnisse für die ausgewählten Datenquellen einer speziellen Domäne. Das gewählte Evaluations-Setup war Mittel zum Nachweis der Nutzbarkeit der grundlegenden MOMA-Ideen. So konnte z. B. gezeigt werden, dass durch die Kombination unabhängiger Match-Ergebnisse eine Qualitätsverbesserung erzielt werden kann. Gleichzeitig können Same-Mappings unterschiedlichen Typs kombiniert werden, um neue Same-Mappings effizient durch Komposition zu bestimmen. Der Nutzen der generellen MOMA-Idee der Wiederverwendung von Mappings zeigt sich vor allem beim Neighborhood-Matcher, wobei durch die Einbindung von Assoziations-Mappings in den Match-Prozess sehr gute Match-Ergebnisse erzielt werden können.

Teil IV

Zusammenfassung und Ausblick

11 Zusammenfassung und Ausblick

11.1 Zusammenfassung

Die vorliegende Arbeit beschäftigte sich mit der automatischen Mapping-Verarbeitung auf Webdaten. Im ersten Teil wurden dazu Recommendations betrachtet, d. h. Webseitenempfehlungen, die Nutzern Hinweise auf potenziell interessante andere Seiten geben. Der zweite Teil thematisiert eine flexible, Workflow-basierte Mapping-Verarbeitung zur Datenintegration.

11.1.1 Adaptive Webseitenempfehlungen

Es wurde mit AWESOME eine Architektur präsentiert, welche für beliebige Websites adaptive Recommendations ermöglicht, wobei insbesondere mit dem Data Warehouse und der Recommender-Bibliothek die Kernkomponenten näher vorgestellt wurden. Das Data Warehouse ermöglicht dabei die Integration verschiedener, für die Website relevanter Datenquellen und bildet dadurch eine einheitliche Plattform für alle Recommender. Diese stehen innerhalb einer erweiterbaren Recommender-Bibliothek zur Verfügung, so dass verschiedene Verfahren zur Bestimmung der Recommendations verwendet werden können.

Ein entscheidendes Merkmal von AWESOME ist das automatische Aufzeichnen des Nutzer-Feedbacks, d. h. ob Nutzer die angezeigten Recommendations angeklickt haben oder nicht, und dessen Integration in das Data Warehouse. Mit Hilfe von Evaluationsmetriken kann dadurch eine Data-Warehouse-gestützte OLAP-Analyse durch-

geführt werden, welche die Evaluation der Qualität präsentierter Recommendations sowohl bezüglich verschiedener Nutzergruppen als auch für einzelne Recommender bzw. Recommender-Klassen ermöglicht.

Zusätzlich wurde ein flexibler, regelbasierter Mechanismus zur dynamischen Auswahl von Recommendern zur Bestimmung adaptiver Recommendations eingeführt. Der einfache Aufbau der Selektionsregeln erlaubt sowohl deren manuelle als auch automatische Erstellung. Dazu wurden insbesondere zwei automatische Verfahren vorgestellt, die das aufgezeichnete Nutzer-Feedback in Selektionsregeln umwandeln und dadurch eine automatische Closed-Loop-Optimierung der Recommendations ermöglichen. Der AWESOME-Ansatz wurde innerhalb einer Website prototypisch implementiert und die Recommender sowie die dynamische Recommender-Selektion evaluiert. Dabei konnte gezeigt werden, dass die automatische Adaption ähnliche Ergebnisse erzielt wie eine aufwändige, manuelle Definition der Recommender-Selektion.

11.1.2 Mapping-basierte Datenintegration

Der neuartige Datenintegrationsansatz iFuice wurde im zweiten Teil der Arbeit vorgestellt. iFuice erlaubt eine einfache Verknüpfung von Datenquellen innerhalb einer P2P-artigen Struktur durch die Bildung von Instanz-Mappings, d. h. Korrespondenzen zwischen Objekten der einzelnen Datenquellen. Durch eine einfache und generische Modellierung der Datenquellen können neben strukturierten Datenquellen, z. B. Datenbanken, auch semistrukturierte Datenquellen, z. B. Webseiten, integriert werden. Zusätzlich unterstützt iFuice durch die dynamische Erstellung von Anfragen auf Basis von Attributwerten von Objektinstanzen die Anbindung von Datenquellen, auf die nur über Anfragen zugegriffen werden kann (z. B. Suchmaschinen). Mit Hilfe einer Skriptsprache steuert ein Mediator den nutzerdefinierten Prozess der Datenintegration. Die im Source-Mapping-Modell registrierten Datenquellen und Mappings können durch den Nutzer mit Hilfe verschiedener High-Level-Operatoren innerhalb von Skripten verarbeitet werden. Sämtliche Ergebnisse werden in einem generischen Warehouse abgespeichert und stehen dadurch zur weiteren Verwendung in anderen Skripten zur Verfügung.

Als Einsatzszenario wurde die prototypische Implementation mit ausgewählten Datenquellen aus dem Bereich bibliografischer Daten für eine Zitierungsanalyse verwendet. Dabei wurde die Flexibilität des iFuice-Ansatzes demonstriert, indem unterschiedliche Arten von Datenquellen verknüpft wurden (Datenbank, Website, Suchmaschine, lokale Datei). Die präsentierten Beispiel-Workflows illustrierten das nahtlose Zusammenspiel von Mapping-Erstellung und -Kombination sowie deren Verwendung zur Informationsfusion. Zusätzlich wurden zwei auf iFuice basierende Applikationen vorgestellt, die die Verwendung der Integrationsplattform in konkreten Anwendungen zeigen.

Der letzte Teil der vorliegenden Arbeit beschäftigte sich mit dem auf iFuice aufsetzenden Object-Matching-Framework MOMA. Es unterstützt die flexible Definition sogenannter Match-Workflows zur Identifikation gleicher Objekte der realen Welt in (verschiedenen) Datenquellen. Innerhalb der Workflows können Matcher einer erweiterbaren Matcher-Bibliothek ausgeführt werden, wobei MOMA bereits Matcher auf Basis der syntaktischen Ähnlichkeit von Attributwerten bereitstellt. Eine Besonderheit von MOMA ist die Mapping-Kombination, d. h. die gezielte Verknüpfung von Mappings zur Verbesserung der Match-Qualität. Neben der Wiederverwendung bestehender Same-Mappings wird dabei auch die nahtlose Integration von Assoziations- und Same-Mapings unterstützt. Die für ein konkretes Match-Problem eingesetzten Workflows können sich sogenannter Match-Strategien bedienen, welche als Sub-Workflows definiert sind und typische, in verschiedenen Match-Szenarien einsetzbare Kombinations- und Selektionsstrategien abbilden. Die Evaluation der verschiedenen Match-Strategien erfolgte ebenfalls mit Hilfe verschiedener Datenquellen aus dem Bereich bibliografischer Daten.

11.2 Ausblick

Die in der vorliegenden Arbeit vorgestellten Ansätze zur Mapping-Verarbeitung zeichnen sich insbesondere durch ihre Flexibilität aus, die u. a. aus der generischen Repräsentation von Instanzkorrespondenzen resultiert. Daher ergeben sich vielfältige Möglichkeiten zum Einsatz sowie zur Erweiterung der vorgestellten Ansätze. Fünf mögliche Richtungen für weitere Arbeiten werden im Folgenden kurz skizziert.

Website-Portal: Während der AWESOME-Ansatz auf optimierte Webseitenempfehlungen innerhalb einer Website abzielt, können mittels iFuice/MOMA Korrespondenzen zwischen Websites erstellt und verarbeitet werden. Eine Kombination der Ansätze ist z. B. für ein Webportal im E-Commerce-Bereich denkbar, welches mehrere E-Commerce-Websites im Rahmen einer Partnerschaft bündelt. Neben der Identifikation gleicher Produkte in den verschiedenen E-Shops kann die Analyse des Nutzerverhaltens auf mehrere Websites ausgedehnt werden. Dadurch können Recommendations zu Produkten eines anderen E-Shops gegeben werden, was zu besseren Empfehlungen insbesondere dann führt, wenn das Produktsortiment der einzelnen E-Shops begrenzt ist.

Ontology Matching: Die uniforme Datenrepräsentation mittels Objektinstanzen und Mappings in iFuice ermöglicht auch eine Darstellung von Ontologien [183]. Beziehungen sowohl zwischen Konzepten und Instanzen als auch zwischen Konzepten selbst lassen sich durch Assoziations-Mappings darstellen. Ontology Matching reduziert sich dadurch auf die Bestimmung eines Same-Mappings zwischen den Konzepten zweier Ontologien.

Erste Untersuchungen für den Spezialfall von Hierarchien wurden mit Hilfe realer Daten aus dem E-Commerce-Bereich [190] (sowie für die Bioinformatik [103]) bereits begonnen. Dabei wurde ein Mapping zwischen den hierarchisch angeordneten Produktkategorien zweier E-Commerce-Shops anhand der assoziierten Produkte bestimmt. Das Same-Mapping zwischen den Produkten ließ sich dabei einfach über die eindeutige EAN[64] bestimmen. Durch die Verknüpfung des Produkt-Same-Mappings mit den jeweiligen Assoziations-Mappings zwischen Produkten und Kategorien gelingt es, semantisch gleiche Kategorien mit unterschiedlichen Namen einander zuzuordnen, z. B. wenn beiden Kategorien die gleichen Produkte zugeordnet sind. Weiterhin können die bei iFuice zur Verfügung stehenden aggregierten Objekte verwendet werden, um Korrespondenzen zwischen Mengen von Kategorien zu bestimmen. Dies ist zum einen dann nötig, wenn die Kategorienbäume eine unterschiedliche Granularität aufweisen, so dass z. B. eine Kategorie zur Vereinigung mehrerer Kategorien im anderen Kategorienbaum zugeordnet werden soll. Andererseits können Korrespondenzen bei orthogonalen Kategorisierungen besser durch Korrespondenzen zwischen Kategorienmengen erfasst werden, z. B. wenn ein Kategorienbaum anhand der Produkthersteller und ein anderer nach dem Typ des Produkts erstellt wurde.

Erstellung optimierter Match-Workflows: Das MOMA-Framework ermöglicht eine flexible Definition von Match-Workflows durch iFuice-Skripte. Die verwendeten Operatoren werden u. a. durch Parameter gesteuert, wie z. B. der Konfidenzfunktion bei `union` oder `compose` oder einem Schwellwert bei der Selektion von Korrespondenzen. Eine automatische Optimierung solcher Parameter für einen gegebenen Match-Workflow ist daher ein vielversprechender Ansatz zur Steigerung der Match-Qualität. Die Optimierung kann dabei z. B. mit Hilfe von manuell definierten Test- und Trainingsdaten realisiert werden. Zusätzlich kann auch die Generierung der Match-Workflows automatisiert werden. Die generischen Datenstrukturen sowie die mengenorientierten Operatoren ermöglichen z. B. den Einsatz genetischer Algorithmen [70] zur Generierung von Match-Workflows.

Anfragestrategien zur Integration von Suchmaschinen: Der Integration von Suchmaschinen kommt eine besondere Bedeutung zu, da sie i. Allg. einen sehr großen Datenbestand vorhalten und selbst z. T. Datenintegration betreiben. Im Gegensatz zu Datenbanken oder Dateien kann der komplette Datenbestand nicht vorab ermittelt werden, da lediglich über Anfragen zugegriffen werden kann. Daher muss z. B. das Object Matching online durchgeführt werden, d. h. innerhalb eines MOMA-Match-Workflows kann nicht immer auf bestehende Same-Mappings zurückgegriffen werden. Vielmehr muss für Objektinstanzen, für die ein Same-Mapping mit den Objektinstanzen der Suchmaschine ermit-

[64] European Article Number

telt werden soll, anhand der Attributwerte eine Menge von Anfragen bestimmt werden. Die Zahl der resultierenden Anfragen ist dabei aus Performanzgründen sowie Zugriffsrestriktionen, z. B. einer Maximalzahl von Anfragen für einen definierten Zeitraum, zu minimieren. Ziel ist es, mit möglichst wenig Anfragen möglichst alle relevanten Suchergebnisse zu ermitteln. Es muss somit ein Mechanismus entwickelt werden, welcher die flexible Generierung solcher Anfragen im Hinblick auf die durch die einzelnen Suchmaschinen angebotene Query-Funktionalität ermöglicht. Für die Suchmaschine Google Scholar wurde im Rahmen der OCS-Webapplikation [188] ein initiales Vorgehen mit mehreren Anfragestrategien realisiert (siehe Abschnitt 8.5.2).

Kollaborative Datenintegration: Unter dem Schlagwort „Web 2.0“ sind in den vergangenen Jahren Websites entstanden, die sich neben einer benutzerfreundlichen Bedienung auch durch die aktive Mitwirkung der Website-Nutzer an den Inhalten auszeichnen. Beispiele sind u. a. Flickr[65] und Delicious[66], bei denen Nutzer Bilder bzw. Bookmarks beisteuern und annotieren. Auf der anderen Seite entstehen immer mehr Webservices, mit deren Hilfe auf Datenbestände oder Funktionalitäten zugegriffen werden kann. Mit Hilfe von iFuice kann eine Website realisiert werden, welche die aktive Mitwirkung von Nutzern und das Angebot von Webservices kombiniert. Nutzer können dabei iFuice-Skripte erstellen und innerhalb der Website ausführen, um so den Datenbestand zu erweitern oder zu verändern. Die Webservices werden dabei im Source-Mapping-Modell registriert und können innerhalb der Skripte ausgeführt werden.

[65] http://www.flickr.com/
[66] http://del.icio.us/

Teil V

Anhang

A

iFuice-Skriptsprache

iFuice ermöglicht mit Hilfe von Skripten und Prozeduren die Definition komplexer Workflows zur Mapping-basierten Datenintegration. In diesem Kapitel wird zunächst die Syntax von iFuice-Skripten und -Prozeduren mittels Syntaxdiagrammen und der erweiterten Backus-Naur-Form definiert (Abschnitt A.1). Anschließend werden die grundlegenden iFuice-Datenstrukturen vorgestellt, deren Instanzen mittels Variablen gespeichert werden können (Abschnitt A.2). Abschließend erfolgt eine vollständige Definition aller iFuice-Operatoren inklusive Aufrufparametern und Beispielen (Abschnitte A.3 bis A.9).

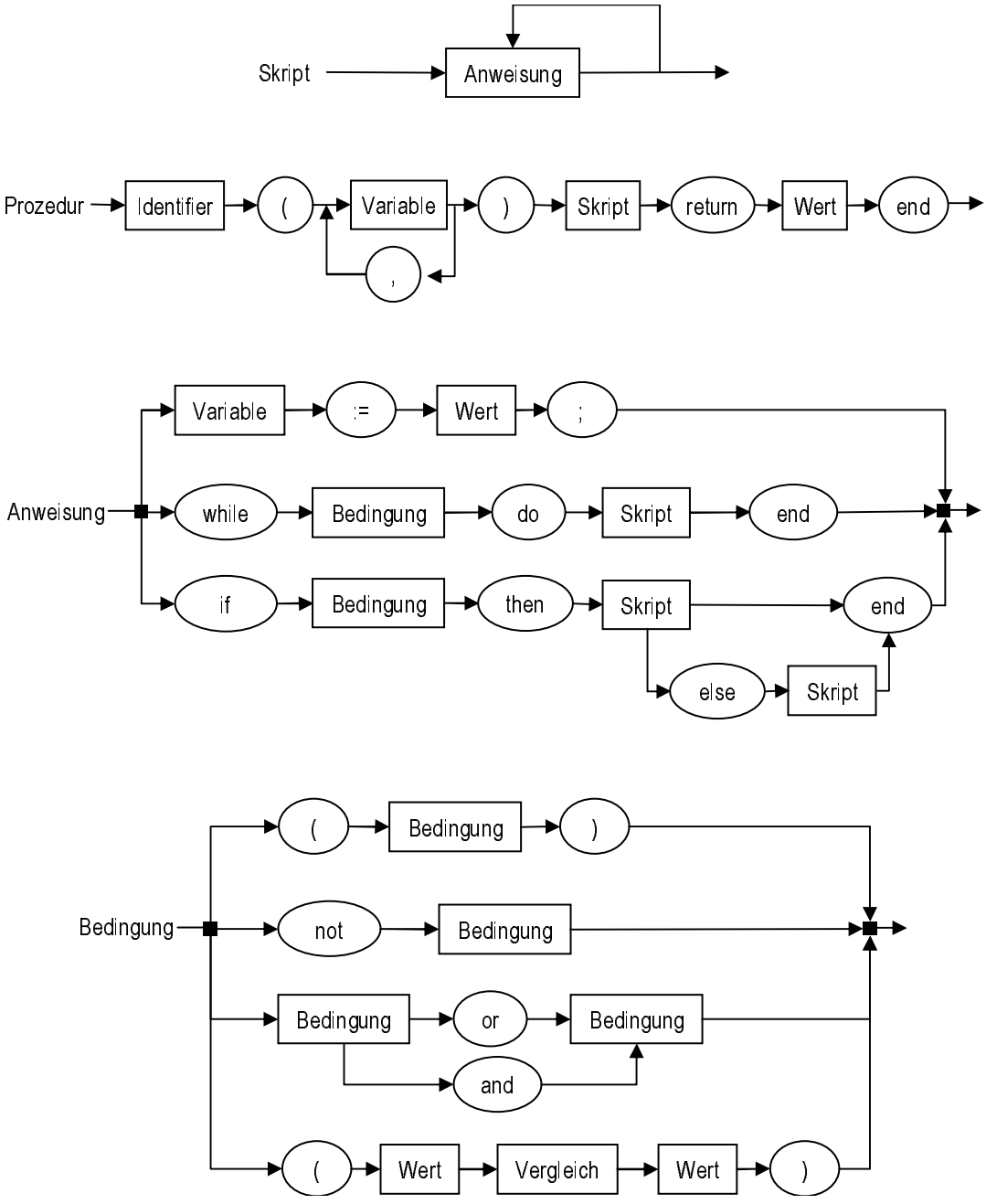
A.1 Aufbau von iFuice-Skripten

Die iFuice-Programmiersprache ist eine prozedurale Programmiersprache, welche die Erstellung von zwei Arten von Programmen ermöglicht.

Skripte sind direkt ausführbare Anweisungsfolgen. Sie greifen auf existierende Variablen zu, können deren Inhalt verändern sowie neue Variablen erstellen. Nach Beendigung einer Skriptausführung stehen die Werte der durch das Skript verarbeiteten Variablen als Ergebnis zur Verfügung. Skripte sind nicht parametrisiert, so dass sie einfach (z. B. durch Angabe des Namens einer Datei, welche das Skript beinhaltet) ausgeführt werden können. Die Möglichkeit der Verwendung bestehender Variablen innerhalb von Skripten erlaubt es dennoch, die Skriptausführung durch entsprechende Startbelegungen von Variablen zu steuern.

Prozeduren sind parametrisierte Anweisungsfolgen, welche in Skripten aufgerufen werden können. Eine Prozedur definiert einen Rückgabewert, welcher nach Beendigung der Prozedurausführung im aufrufenden Skript zur Verfügung steht. Die weiteren in Prozeduren benutzten Variablen sind lokal und daher nicht im aufrufenden Skript verfügbar.

Die nachfolgenden Syntaxdiagramme definieren den Aufbau von Skripten und Prozeduren.



Die weiteren Teile der iFuice-Skriptsprache werden nun mittels der erweiterten Backus-Naur-Form (EBNF) angegeben.

```
Wert      ::= <Variable>|<Konstante>|<Zeichenkette>|<Zahl>|
              <Operator> "(" <Parameter> ("," <Parameter>)* ")"
Parameter ::= <Wert>|<Identifier>

Variable          ::= "$" <Identifier>
Konstante         ::= "%" <Identifier>

Identifier    ::= <Buchstabe> (<Ziffer>|<Buchstabe>|"_"|"."|"@")*
Zeichenkette  ::= (<Buchstabe> | <Ziffer> | <Sonderzeichen>)*
Zahl          ::= ("+" | "-" )?  <Ziffer>+  ( "."  <Ziffer>+)?

Ziffer        ::= "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
Buchstabe     ::= "a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|
                  "l"|"m"|"n"|"o"|"p"|"q"|"r"|"s"|"t"|"u"|"v"|
                  "w"|"x"|"y"|"z"|"A"|"B"|"C"|"D"|"E"|"F"|"G"|
                  "H"|"I"|"J"|"K"|"L"|"M"|"N"|"O"|"P"|"Q"|"R"|
                  "S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"
Sonderzeichen ::= "."|":"|","|";"|"-"|"_"|"<"|">"|"#"|"'"|"+"|
                  "*"|"@"|"!"|"\"|"$"|"%"|"/"|"&"|"("|")"|"["|
                  "]"|"="|"?"|" "
Vergleich     ::= "<"|"<="|"=="|">="|">"
```

<Operator> definiert sich aus der Liste der zur Verfügung stehenden Operatoren, die ab Abschnitt A.3 vorgestellt werden. Die Darstellung orientiert sich an den in Kapitel 7.4 präsentierten sieben Klassen von Operatoren, die in Tabelle 7.2 (ab Seite 143) mit allen zugehörigen Operatoren aufgelistet sind.

A.2 Datenstrukturen und Notationen

iFuice unterstützt vier komplexe Datentypen.

Object Instances (O) ist eine Menge von Objektinstanzen einer LDS. Sie besitzen daher alle denselben Objekttyp und stammen aus derselben PDS. Weiterhin definiert die LDS ein ID-Attribut, für welches jedes Objekt einen nichtleeren Wert besitzt. Zusätzlich hat jedes Objekt eine (möglicherweise leere) Liste von weiteren Attributen.

Mapping Result (MR) ist eine Menge von Korrespondenzen zwischen Objekten. Dabei stammen sowohl alle Domain-Objekte als auch alle Range-Objekte jeweils aus der gleichen LDS. Jede Korrespondenz besitzt mit der Konfidenz ein „Korrespondenzattribut“, das die Güte der Korrespondenz charakterisiert. Die Konfidenz ist eine reelle Zahl im Intervall [0,1], welche die Sicherheit, Glaubwürdigkeit oder Zuversicht der Korrespondenz definiert. Dabei bedeutet der Wert 1 (0) vollständige (Un-)Sicherheit.

Aggregated Objects (AO) ist eine Menge aggregierter Objekte, die wiederum jeweils eine nicht-leere Menge von Objekten des gleichen Objekttyps (nicht notwendigerweise der gleichen LDS) darstellen. Zusätzlich zu den Attributen der Objekte kann jedes aggregierte Objekt noch sogenannte fusionierte Attribute besitzen, welche keiner Objektinstanz sondern ausschließlich dem aggregierten Objekt zugeordnet sind.

Aggregated Mapping Result (AMR) ist eine Menge von Korrespondenzen zwischen aggregierten Objekten. Analog zu MR besitzen alle Domain- und alle Range-Objekte jeweils denselben Objekttyp. Alle Korrespondenzen werden ebenfalls mit einer Konfidenz annotiert.

Des Weiteren gibt es drei einfache Datenstrukturen: **String (S)** ist eine Zeichenkette, **Integer (I)** ist eine ganze Zahl und **Float (F)** ist eine reelle Zahl.

A.2.1 Notationen

Bei der Definition der Operatoren werden insbesondere die Parameter sowie der Rückgabewert definiert. Dabei gibt es zwischen den Parametern und Rückgabewerten Abhängigkeiten, z. B. dass bei der Vereinigung von Objektinstanzen (Operator `union`) nur Objektinstanzen des gleichen Objekttyps als Eingabe verwendet werden dürfen, wobei der Rückgabewert vom selben Typ ist. Dazu werden bei der Definition Indizes mit folgender Bedeutung verwendet.

LDS: LDS_A bezeichnet eine logische Datenquelle A.

O: O_A bezeichnet eine Menge von Objektinstanzen der LDS A. Die Instanzen werden mit a_1, a_2, usw. bezeichnet.

MR: MR_{AB} bezeichnet ein Mapping Result des Typs $A \times B$, d. h. die Domain-LDS ist A und die Range-LDS ist B. Die Korrespondenzen werden mit (a_i, b_k, c) bezeichnet, wobei $c \in [0, 1]$ die Konfidenz angibt und a_i und b_k das Domain- bzw. Range-Objekt definieren.

AO: AO_S bezeichnet eine Menge aggregierter Objekte des Objekttyps S. Die Instanzen werden mit $\hat{s}_i$ bezeichnet, wobei jedes $\hat{s}_i$ selbst wieder eine Menge von Objektinstanzen ist.

AMR: AMR_{ST} bezeichnet ein aggregiertes Mapping Result des Typs $S \times T$, d. h. der Domain-Objekttyp ist S und der Range-Objekttyp ist T. Die Korrespondenzen sind $(\hat{s_i}, \hat{t_k}, c)$, wobei $c \in [0, 1]$ die Konfidenz angibt und $\hat{s_i}$ und $\hat{t_k}$ die aggregierten Domain- bzw. Range-Objekte definieren.

Zum besseren Verständnis für die Funktionsweise der Operatoren werden an vielen Stellen Beispiele gegeben. Dazu wird von den im Folgenden beschriebenen Darstellungen der Datenstrukturen Gebrauch gemacht.

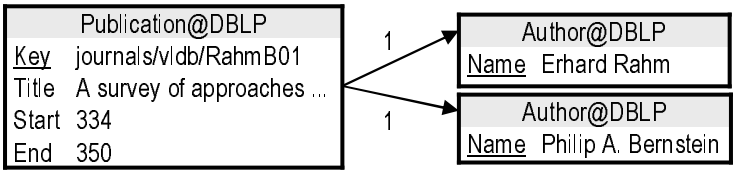
Grafische Darstellung

Zur Beschreibung der Wirkungsweise von Operatoren werden deren Einfluss auf die Datenstrukturen durch Abbildungen von Beispielen illustriert. Bei der grafischen Darstellung werden Objektinstanzen, aggregierte Objekte, Attribute und Korrespondenzen charakterisiert.

Die nachfolgende Abbildung zeigt die Darstellung einer Objektinstanz inklusive aller Attribute sowie der (grau hinterlegten) LDS, aus der die Objektinstanz stammt. Die Attribute sind zeilenweise als Name-Wert-Paare angegeben, wobei das ID-Attribut unterstrichen ist.

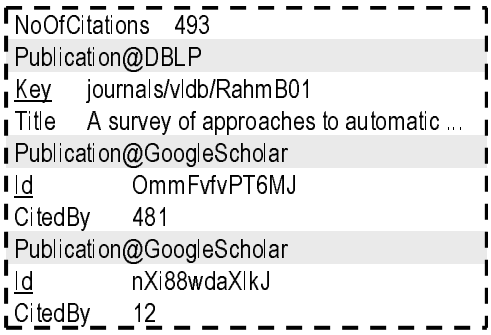
Publication@DBLP	
Key	journals/vldb/RahmB01
Title	A survey of approaches to automatic schema matching
Start	334
End	350

Die Repräsentation eines MR erfolgt durch die grafische Darstellung der Korrespondenzen zwischen den beteiligten Objektinstanzen. Die nachfolgende Abbildung gibt ein Beispiel für ein MR des Typs Publication@DBLP × Author@DBLP mit zwei Korrespondenzen. Jede Korrespondenz wird durch einen Pfeil zwischen den Objektinstanzen dargestellt, wobei am Pfeil die Konfidenz der Korrespondenz notiert ist. Aus Platzgründen bzw. für eine übersichtliche Darstellung werden lange Attributwerte ggf. durch „. . .“ abgekürzt.

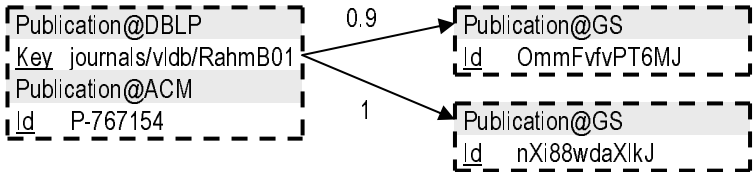


Jedes aggregierte Objekt besteht aus einer Menge von Objektinstanzen des gleichen Objekttyps. Die Zusammenfassung mehrerer Objektinstanzen zu einem aggregierten

Objekt erfolgt in der grafischen Darstellung durch einen gestrichelten Rahmen. Das Beispiel der nachfolgenden Abbildung zeigt ein aggregiertes Objekt mit drei Objektinstanzen. Der Objekttyp des aggregierten Objekts ergibt sich dabei aus dem Objekttyp der Instanzen. Zusätzlich zu den Attributen der Objektinstanzen können aggregierte Objekte sogenannte fusionierte Attribute besitzen. Diese sind keiner Objektinstanz zugeordnet und werden daher „über“ der ersten Instanz aufgeführt.



Die Repräsentation eines AMR ist analog zur MR-Repräsentation, d. h. Pfeile mit angegebener Konfidenz als Darstellung der Korrespondenzen zwischen den aggregierten Objekten. Aus Gründen der Übersichtlichkeit werden nicht immer alle Attribute dargestellt, so dass z. B. in der nachfolgenden Abbildung lediglich die (notwendigen und immer dargestellten) ID-Attribute aufgeführt sind.



Tabellarische Darstellung

Die beschriebene grafische Darstellung wird i. Allg. dann gewählt, wenn die Attribute und insbesondere deren Werte im Mittelpunkt des Interesses stehen. Für eine kompaktere Darstellung der Datenstrukturen wird die tabellarische Form gewählt. Hauptunterscheidungsmerkmal zur grafischen Darstellung ist die Repräsentation einer Objektinstanz durch einen einzigen Wert.

Die nachfolgende Abbildung zeigt ein Beispiel für eine Menge von Objektinstanzen. Es zeigt vier Instanzen der LDS Venue@DBLP, die wie bei der grafischen Darstellung grau hinterlegt wird. Die Instanzen sind in der einspaltigen Tabelle jeweils durch ihren ID-Wert definiert.

Venue@DBLP
conf/vldb/2000
conf/vldb/2001
conf/vldb/2002
conf/vldb/2003

Die Repräsentation eines MR entspricht einer dreispaltigen Tabelle, wobei jede Zeile einer Korrespondenz zwischen Domain- (erste Spalte) und Range-Objekt (zweite Spalte) entspricht. In der dritten Spalte wird die Konfidenz vermerkt. Die Spaltenüberschriften benennen die Domain- und Range-LDS sowie mit „Conf“ die Konfidenz. Die Abbildung zeigt ein Beispiel eines MR des Typs Publication@DBLP × Publication@ACM. Hervorzuheben ist die Tatsache, dass die Objekte der Übersichtlichkeit in diesem Beispiel *nicht* durch ihr ID-Attribut definiert sind, sondern ein anderes „charakteristisches“ Attribut (im Beispiel der Publikationstitel) gewählt wurde. Diese Variante dient einem einfacheren Verständnis für den Leser und wird verwendet, solange die Instanzen innerhalb des Beispiels weiterhin eindeutig beschrieben sind.

Publication@DBLP	Publication@ACM	Conf
iFuice – Information ...	iFuice – Information ...	1
A Survey of Approaches ...	A Survey of Approaches ...	1

Aggregierte Objekte werden analog zu Objektinstanzen in einer einspaltigen Tabelle beschrieben. Zur Definition jedes aggregierten Objekts wird die Menge der zugehörigen ID-Attribute der Instanzen in eckige Klammern gesetzt. Die nachstehende Abbildung gibt ein Beispiel für aggregierte Objekte des Typs Venue, die jeweils aus einer oder zwei Objektinstanzen bestehen.

Venue
[conf/vldb/2000, V-645926]
[conf/vldb/2001, V-645927]
[conf/vldb/2002]
[conf/vldb/2003]

Die tabellarische Repräsentation der AMR kombiniert die dreispaltige Mapping-Tabelle mit der Angabe aggregierter Objekte durch die Menge der ID-Attribute. Im Unterschied zu den bisherigen „konkreten“ Beispielen wird im Folgenden ein „abstraktes“ Beispiel mit den Objekttypen S und T gegeben. Die Objektinstanzen werden durch Bezeichner (z. B. a_1) charakterisiert und entsprechend zu aggregierten Objekten zusammengefasst (z. B. $[a_1,a'_1]$).

S	T	Conf
$[a_1, a'_1]$	$[b_1, b'_1]$	0.8
$[a_1, a'_1]$	$[b_2, b'_2]$	0.7
$[a_2]$	$[b'_2]$	1

A.3 Query-Operatoren

Mit Hilfe der Query-Operatoren `queryInstances`, `queryMap` und `join` können durch Angabe von Bedingungen Mengen von (aggregierten) Objekten und (aggregierten) Korrespondenzen definiert werden. Die Bedingungen beschreiben einfache Filterausdrücke auf Basis der Attribute und können sowohl mit Hilfe eines Query-Mappings als auch innerhalb des Mediators ausgeführt werden.

A.3.1 queryInstances

Der Operator `queryInstances` dient zur Bestimmung von Objektinstanzen bzw. aggregierten Objekten anhand einer bestimmten Bedingung. Bei Angabe einer LDS wird die Query an die Datenquelle gesendet, d. h. das entsprechende Query-Mapping wird ausgeführt. Alternativ kann `queryInstances` auch auf eine Variable vom Typ O oder AO angewendet werden, wobei in diesem Fall die Query-Funktionalität des Mediators verwendet wird.

Aufruf O_A `= queryInstances (`LDS_A`,` *Bedingung*`)`

Beschreibung Es werden alle Objekte der Datenquelle LDS_A, welche die angegebene Bedingung erfüllen, durch Ausführung des entsprechenden Query-Mappings ermittelt. Die Syntax der Bedingung richtet sich nach der vom Query-Mapping unterstützten Funktionalität.

Beispiel
```
$Result := queryInstances (Venue@DBLP, "[series] =
           'VLDB' and [year]>1999 and [year]<=2003")
```
Es werden alle DBLP-Venues der Konferenzserie „VLDB“ nach 1999 bis einschließlich 2003 selektiert.

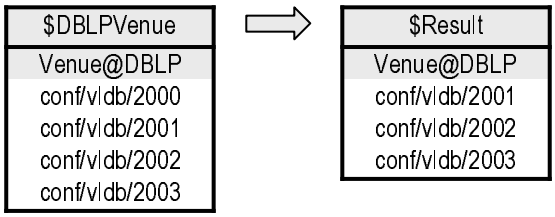
$Result
Venue@DBLP
conf/vldb/2000
conf/vldb/2001
conf/vldb/2002
conf/vldb/2003

Aufruf O_A `= queryInstances (`O_A`,` *Bedingung*`)`

Beschreibung Alle Objekte aus O_A, welche die angegebene Bedingung erfüllen, werden selektiert und zurückgegeben. Die Bedingungsprüfung wird dabei innerhalb des Mediators ausgeführt.

Beispiel
```
$Result := queryInstances ($DBLPVenue, "[year]>=2001")
```

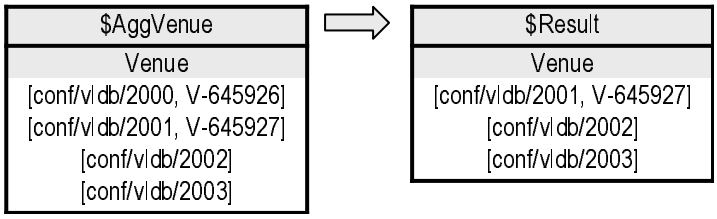
Es werden alle Venues aus `$DBLPVenue` ab dem Jahr 2001 selektiert.



Aufruf AO_S `= queryInstances (`AO_S`,` *Bedingung*`)`

Beschreibung Es werden alle aggregierten Objekte aus AO_S, welche die angegebene Bedingung erfüllen, ermittelt. Bei der Angabe der Bedingung kann auf die Attribute der einzelnen Objektinstanzen eines aggregierten Objekts zugegriffen werden. Da in einem aggregierten Objekt Objektinstanzen aus mehreren PDS enthalten sein können, muss dem Attributnamen der Name der PDS vorangestellt werden. Da weiterhin mehrere Instanzen aus einer PDS in einem aggregierten Objekt enthalten sein können, muss zusätzlich für jedes Attribut eine Aggregatfunktion (analog zur Verwendung von `GROUP BY` bei SQL-Anfragen) angegeben werden. Dazu stehen `min`, `max` und `count` sowie zusätzlich für numerische Werte `avg` und `sum` zur Verfügung.

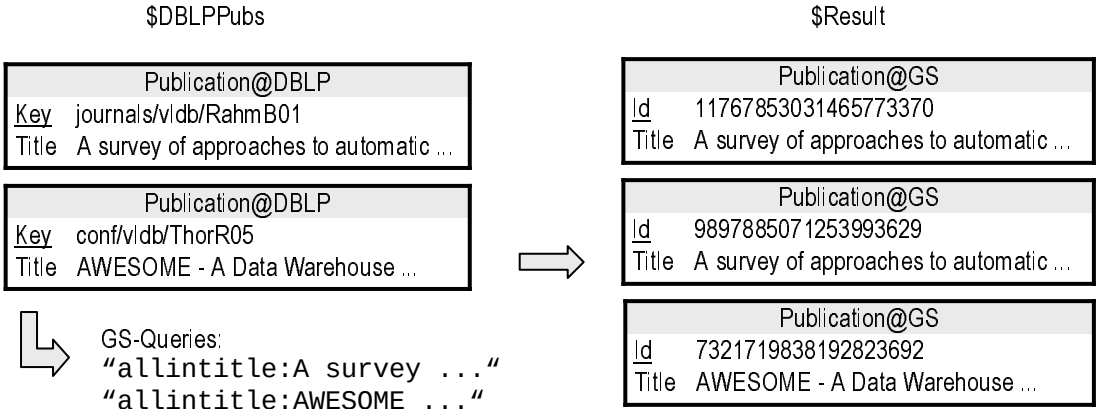
Beispiel
```
$Result := queryInstances ($AggVenue,
           "min([DBLP.year])>=2001")
```
Es werden alle aggregierten Venues ab dem Jahr 2001, jeweils bestehend aus dem DBLP-Venue und – sofern in der Datenquelle ACM vorhanden – aus dem zugehörigen ACM-Venue, bestimmt. Die Jahresangabe wird von den DBLP-Instanzen verwendet. Da in diesem Beispiel jeweils nur eine DBLP-Instanz pro aggregiertem Objekt auftritt, liefern die Aggregierungsfunktionen `min` und `max` in diesem Fall die gleichen Ergebnisse.



Aufruf O_B `= queryInstances (`LDS_B`,` *Bedingungsmuster*`,` O_A`)`

Beschreibung Bei den bisher dargestellten Varianten von `queryInstances` kann jeweils nur eine Bedingung angegeben werden. Die Kombination mehrerer Bedingung durch eine logische Oder-Verknpüfung ist bei der Anwendung von `queryInstances` auf O oder AO möglich. Dies ist für den Fall einer LDS nicht immer möglich, da hier die Syntax einer Bedingung vom zugehörigen Query-Mapping abhängig ist. Daher ermöglicht `queryInstances` die Definition und Ausführung einer Menge von Bedingungen. Neben der LDS, deren Query-Mapping verwendet wird, wird ein Bedingungsmuster sowie eine Menge von „Query-Objekten“ O_A spezifiziert. Das Muster ist ein parametrisierter Ausdruck, dessen Parameter durch die Attributwerte der Query-Objekte belegt werden. Für jede Objektinstanz aus O_A entsteht dadurch eine Bedingung für LDS_B. Das Ergebnis ist die Menge aller Objektinstanzen aus LDS_B, die mindestens eine der gebildeten Bedingungen erfüllen.

Beispiel
```
$Result := queryInstances (Publication@GS,
              "allintitle:[[title]]", $DBLPPubs)
```
Für eine Menge von DBLP-Publikationen wird je eine Query zur Bestimmung von GS-Publikationen erstellt und ausgeführt. Die resultierenden Bedingungen bestehen aus dem DBLP-Publikationstitel und dem (GS-spezifischen) Präfix „allintitle:“ zur Einschränkung der Suche auf den GS-Titel.



A.3.2 queryMap

Korrespondenzen zwischen (aggregierten) Objekten werden durch den Operator `queryMap` selektiert. Analog zu `queryInstances` wird ein Bedingungsausdruck definiert, welcher sowohl auf Objekt- als auch auf Korrespondenzattribute zugreift.

Aufruf MR_{AB} `=` `queryMap` `(`MR_{AB}`,` *Bedingung*`)`

Beschreibung Der Aufruf liefert alle Korrespondenzen des übergebenen MR, welche die angegebene Bedingung erfüllen. Hierbei kann sowohl auf die Objektattribute der Domain- und Range-Objekte als auch auf die Konfidenz zugegriffen werden.

Beispiel `$Result := queryMap ($DBLP2ACM, "[_confidence]>0.9")`
Es werden alle Korrespondenzen, deren Konfidenz größer 0.9 ist, ermittelt.

$DBLP2ACM		
Publication@DBLP	Publication@ACM	Conf
iFuice – Information ...	Information Fusion ...	0.8
A Survey of Approaches ...	A Survey of Approaches ...	1

⇨

$Result		
Publication@DBLP	Publication@ACM	Conf
A Survey of Approaches ...	A Survey of Approaches ...	1

Aufruf AMR_{ST} `= queryMap (`AMR_{ST}`,` *Bedingung*`)`

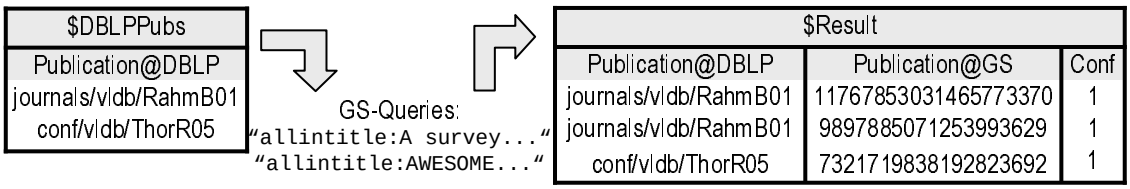
Beschreibung Es werden alle Korrespondenzen des übergebenen AMR, welche die angegebene Bedingung erfüllen, bestimmt. Hierbei können die Attribute der Objektinstanzen (analog zu `queryInstances` mittels Angabe der PDS sowie einer Aggregatfunktion) und die fusionierten Attribute der Domain- und Range-Objekte verwendet werden. Zusätzlich kann die Konfidenz innerhalb der Bedingung genutzt werden.

Aufruf MR_{AB} `= queryMap (`LDS_B`,` *Bedingungsmuster*`,` O_A`)`

Beschreibung Die Ausführung von Query-Mappings mittels des `queryInstances`-Operators liefert Objektinstanzen. Der Operator `queryMap` ermöglicht zusätzlich die Verwendung von Query-Mappings zur Erstellung eines MR. Dazu werden analog zu `queryInstances (`LDS_B`,` *Bedingungsmuster*`,` O_A`)` mehrere Bedingungen erstellt und durch das Query-Mapping ausgeführt. Der Unterschied zum Operator `queryInstances` besteht darin, dass nicht nur die Menge der resultierenden Objektinstanzen zurückgeliefert wird, sondern Korrespondenzen zwischen den Query-Objekten aus O_A und den durch Ausführung der Query entstehenden Objektinstanzen von LDS_B.

Beispiel
```
$Result := queryMap (Publication@GS,
                     "allintitle:[[title]]", $DBLPPubs)
```
Es wird ein Mappings zwischen DBLP- und GS-Publikationen

durch mehrere Anfragen erstellt. Jeder DBLP-Publikation werden durch Korrespondenzen diejenigen GS-Publikationen zugeordnet, die bei der Ausführung des GS-Query-Mappings mit dem jeweiligen DBLP-Publikationstitel zurückgeliefert wurden.



A.3.3 join

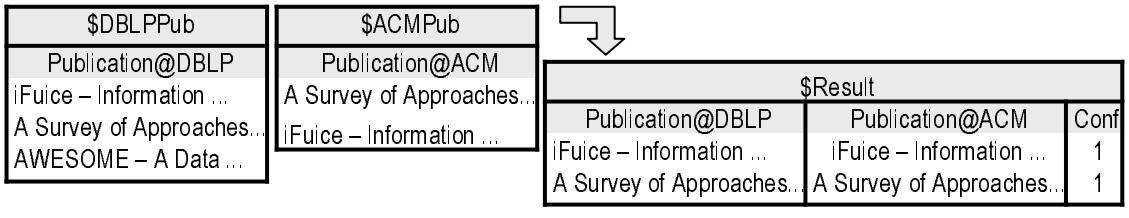
Der Operator `join` erzeugt durch eine Join-Bedingung aus zwei Mengen (aggregierter) Objekte eine Menge von (aggregierten) Korrespondenzen.

Aufruf MR_{AB} `= join (`O_A`,` O_B`,` *Bedingung*`)`

Definition `= queryMap (`$O_A \times O_B$`,` *Bedingung*`)`

Beschreibung Die beiden Mengen von Objektinstanzen werden mit Hilfe einer Bedingung verknüpft, d. h. es werden Korrespondenzen zwischen denjenigen Objektinstanzen gebildet, für die die Bedingung erfüllt ist. Dies entspricht der Ausführung von `queryMap` auf dem „Kreuz-Mapping“ der beiden O, d. h. dem Mapping bestehend aus allen möglichen Korrespondenzen zwischen Objektinstanzen aus O_A und O_B. Zur Unterscheidung zwischen den Attributen der Domain- und Range-Objekte, werden bei der Angabe der Bedingung den Attributnamen die Schlüsselworte `domain` bzw. `range` vorangestellt.

Beispiel
```
$Result := join ($DBLPPub, $ACMPub,
                "[domain.title]=[range.name]")
```
Es wird ein Publikations-Mappings zwischen DBLP und ACM auf Basis gleicher Titel erzeugt. Sollen Korrespondenzen nicht nur bei Gleichheit sondern auch bei hinreichend großer Ähnlichkeit der Attributwerte erstellt werden („fuzzy join“), kann der Operator `attrMatch` verwendet werden (siehe Abschnitt A.5.4).



Aufruf AMR_{ST} = `join` (AO_S, AO_T, *Bedingung*)

Definition = `queryMap` ($AO_S \times AO_T$, *Bedingung*)

Beschreibung Analog zu obigem `join` werden Korrespondenzen zwischen aggregierten Objekten gebildet.

A.4 Attribut-Operatoren

Mit Hilfe der Attribut-Operatoren `getInstances`, `setAttr` und `delAttr` können sowohl Attribute von Objektinstanzen als auch fusionierte Attribute erstellt, verändert und gelöscht werden. Zusätzlich können die Konfidenzen bei MR und AMR manipuliert werden.

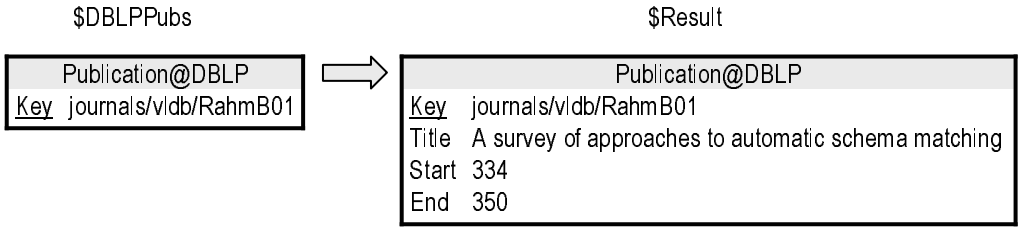
A.4.1 getInstances

Der Operator `getInstances` ermittelt für die übergebenen Objektinstanzen alle – in der zugehörigen Datenquelle – verfügbaren Attribute. Dabei können sowohl neue Attribute erstellt, als auch die Werte bestehender Attribute aktualisiert werden. Die Realisierung erfolgt durch den Aufruf der GetInstance-Funktion der betreffenden LDS.

Aufruf O_A = `getInstances` (O_A)

Beschreibung Es werden alle Attribute der übergebenen Objektinstanzen aktualisiert.

Beispiel `$Result := getInstances ($DBLPPubs)`
Es werden alle Attribute der übergebenen DBLP-Publikationen aktualisiert. Das Beispiel illustriert den „Extremfall“, d. h. die Objektinstanzen besitzen anfangs nur das ID-Attribut.



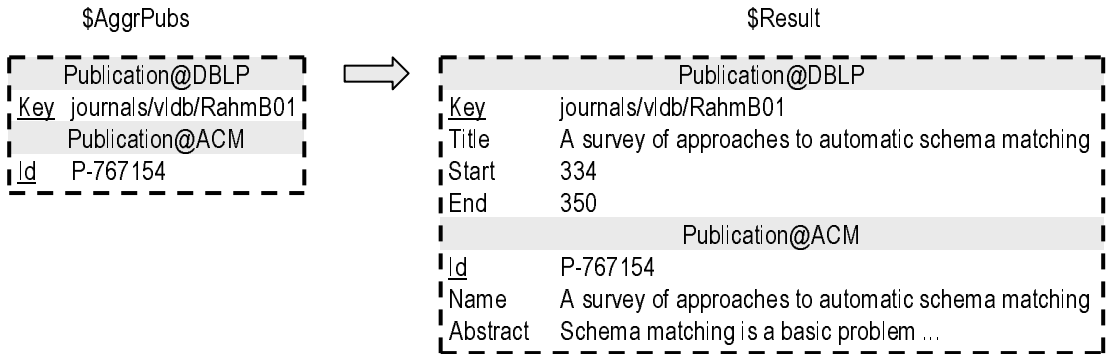
Aufruf MR_{AB} = `getInstances` (MR_{AB})

Beschreibung Es wird `getInstances` sowohl für alle Domain- als auch für alle Range-Objekte ausgeführt.

Aufruf AO_S `= getInstances (`AO_S`)`

Beschreibung Es wird `getInstances` für alle in den aggregierten Objekten enthaltenen Objektinstanzen ausgeführt.

Beispiel `$Result := getInstances ($AggrPubs)`
Es werden die aktuellen Attribute der aggregierten DBLP- und ACM-Publikationen bestimmt.



Aufruf AMR_{ST} `= getInstances (`AMR_{ST}`)`

Beschreibung Es wird `getInstances` sowohl für alle aggregierten Domain- als auch für alle aggregierten Range-Objekte ausgeführt.

A.4.2 setAttr

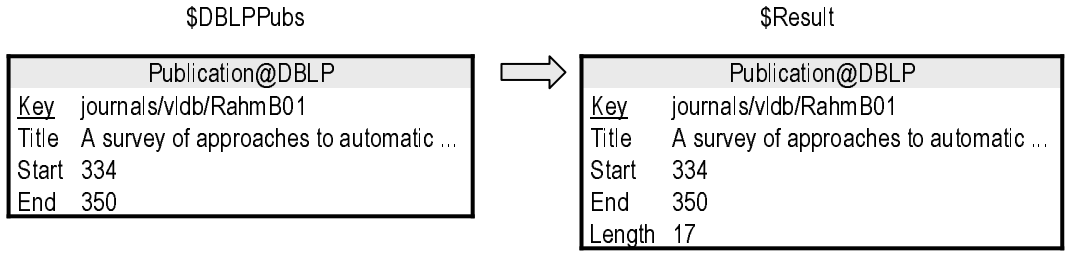
Mit Hilfe des Operators `setAttr` können nutzerdefinierte Attribute erstellt und geändert werden. Dabei können sowohl Attribute von Objektinstanzen als auch fusionierte Attribute, d. h. Attribute aggregierter Objekte, verarbeitet werden. Bei MR und AMR können zusätzlich die Konfidenzwerte definiert werden.

Aufruf O_A `= setAttr (`O_A`, AttrName, AttrValue)`

Beschreibung Den Objektinstanzen wird ein Attribut mit dem Namen `AttrName` hinzugefügt oder – falls es schon existiert – aktualisiert. Der Wert des Attributs wird durch `AttrValue` bestimmt. Dies kann sowohl ein statischer Wert sein als auch ein Ausdruck unter Verwendung der bestehenden Attributwerte des Objekts. Da sämtliche Attribute generisch als String behandelt werden, müssen sie vor arithmetischen Berechnungen entsprechend umgewandelt werden. Die

Syntax der Konvertierung orientiert sich an der mySQL-Syntax[67].

Beispiel `$Result := setAttr ($DBLPPubs, "Length", "1 + cast( [End] as signed)-cast([Start] as signed)")`
Für die übergebenen DBLP-Publikationen wird das Attribut `Length` als Seitenanzahl definiert, d. h. als 1 plus die Differenz von End- und Startseite. Die Attribute `Start` und `End` werden vor der Berechnung in Zahlen konvertiert.



Aufruf MR_{AB} `= setAttr (`MR_{AB}`, "_confidence", AttrValue)`

Beschreibung Es wird der Konfidenzwert mittels `AttrValue` definiert.

Aufruf MR_{AB} `= setAttr (`MR_{AB}`, AttrName, AttrValue)`

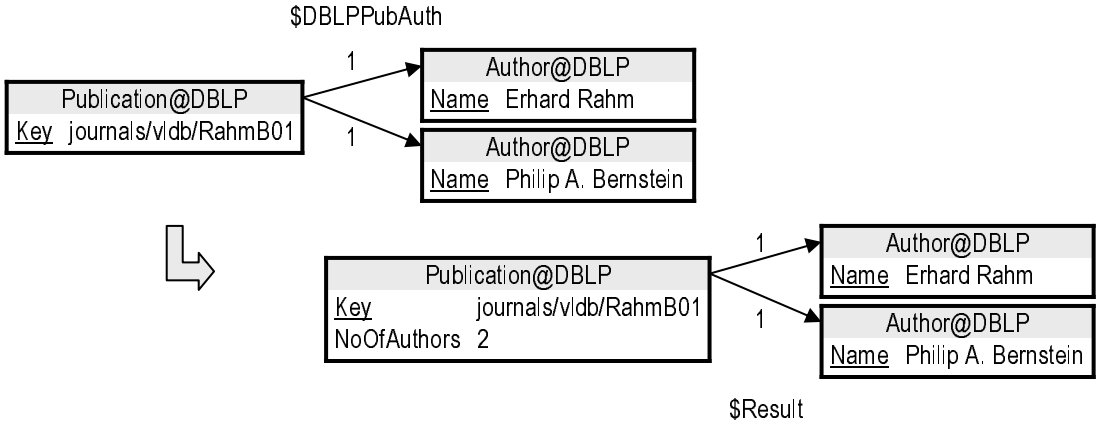
Beschreibung Bei der Anwendung von `setAttr` auf Objektinstanzen können nur (eigene) Attribute der Instanz verwendet werden, um ein neues Attribut zu erstellen oder den Wert eines bestehende Attributs zu aktualisieren. Durch die Verwendung eines MR ist es mit `setAttr` auch möglich, bei der Wertdefinition eines Attributs die Attribute korrespondierender Objektinstanzen zu verwenden. Es werden dabei die Objektattribute der Domain-Objekte verändert.[68] `AttrValue` definiert entweder einen statischen Wert oder greift auf die Attribute der korrespondierenden Range-Objekte zu. Da es zu einem Domain-Objekt Korrespondenzen zu mehreren Range-Objekten geben kann, muss eine Aggregatfunktion verwendet werden. Analog dem Zugriff auf die Attribute der Objektinstanzen aggregierter Objekte (siehe Operator `queryInstances`) stehen die Aggregatfunktionen `min`, `max` und `count` sowie zusätzlich für numerische Werte `avg` und `sum` zur Verfügung.

Beispiel `$Result := setAttr ($DBLPPubAuth, "NoOfAuthors", "count([Name])")`

[67] http://dev.mysql.com/doc/refman/5.1/en/cast-functions.html

[68] Sollen die Range-Objekte verändert werden, muss das Mapping vorher mit dem Operator `inverse` (siehe Abschnitt A.8.2) umgekehrt werden.

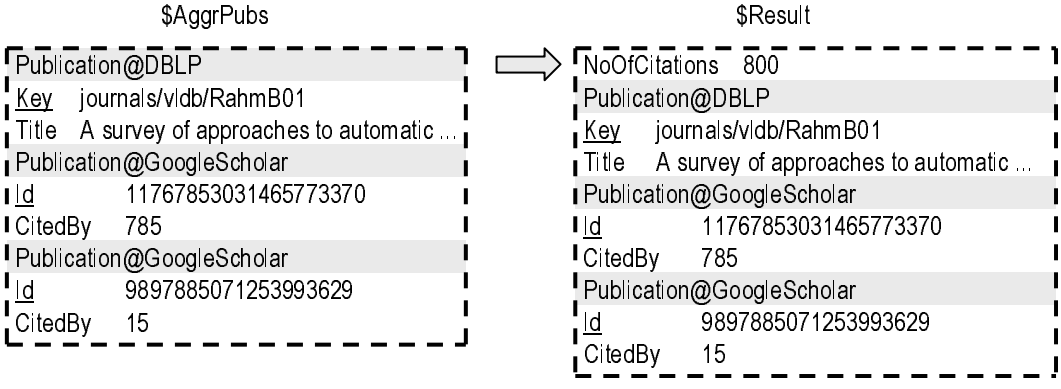
Für DBLP-Publikationen wird die Anzahl der Autoren als Attribut gespeichert, die sich aus der Anzahl der Korrespondenzen jeder Publikation im übergebenen Assoziations-Mapping zwischen Publikationen und Autoren bestimmt.



Aufruf AO_S `= setAttr (`AO_S`, AttrName, AttrValue)`

Beschreibung Es werden fusionierte Attribute manipuliert. Bei der Wertdefinition kann auf die Attribute der einzelnen Objekte des aggregierten Objekts zugegriffen werden. Für die Eindeutigkeit des Attributnamens muss die PDS vorangestellt werden, da Objekte verschiedener PDS Attribute des gleichen Namens aufweisen können. Da es mehrere Objekte einer PDS pro aggregiertem Objekt geben kann, muss zusätzlich eine Aggregatfunktion (siehe oben) angegeben werden.

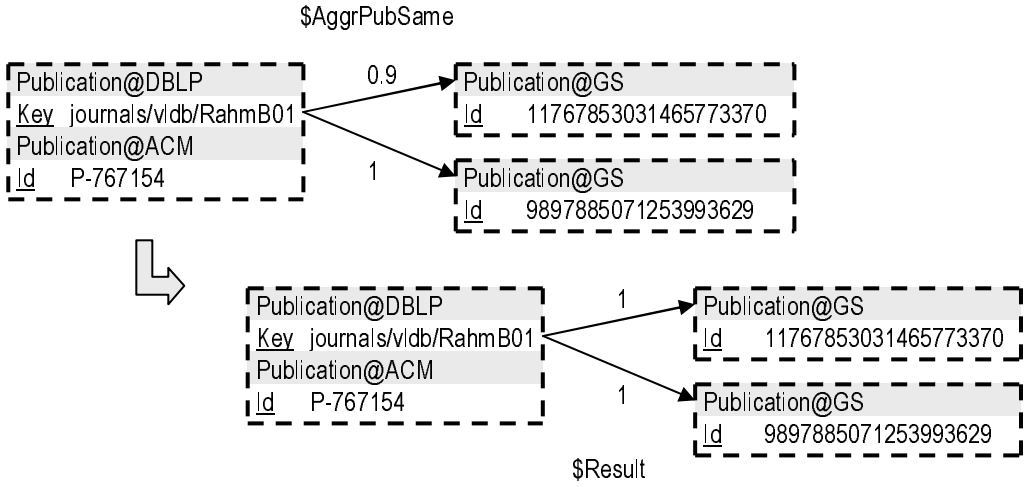
Beispiel
```
$Result := setAttr ($AggrPubs, "NoOfCitations",
                    "sum(cast([GS.Citations] as signed))")
```
Für die aggregierten Publikationen soll eine Gesamtzitierungszahl bestimmt werden. Diese ergibt sich als Summe der von Google Scholar gelieferten Zitierungszahlen.



Aufruf AMR_{ST} `= setAttr (`AMR_{ST}`, "_confidence", AttrValue)`

Beschreibung Es wird der Konfidenzwert mittels `AttrValue` definiert.

Beispiel `$Result := setAttr ($AggrPubSame, "_confidence", "1")`
Der Konfidenzwert aller Korrespondenzen wird auf 1 gesetzt, z. B. weil das verwendete Mapping manuell überprüft wurde und die Korrespondenzen alle als korrekt angesehen wurden.



Aufruf AMR_{ST} `= setAttr (`AMR_{ST}`, AttrName, AttrValue)`

Beschreibung Die Funktionsweise ist analog zu MR. Bei der Wertdefinition kann auf die fusionierten Attribute der aggregierten Range-Objekte zugegriffen werden.

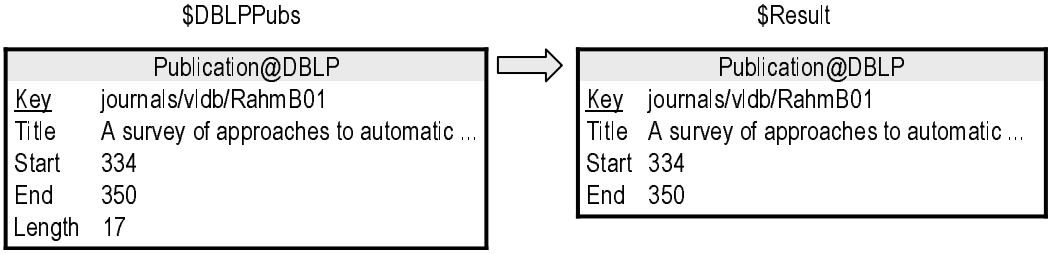
A.4.3 delAttr

Der Operator `delAttr` dient zum Löschen eines oder mehrerer Attribute. Dabei können sowohl Attribute von Objektinstanzen als auch fusionierte Attribute entfernt werden; nicht jedoch ID-Attribute sowie Konfidenzwerte.

Aufruf O_A `= delAttr (`O_A`, AttrName)`

Beschreibung Es wird das angegebene Attribut in allen übergebenen Objekten gelöscht.

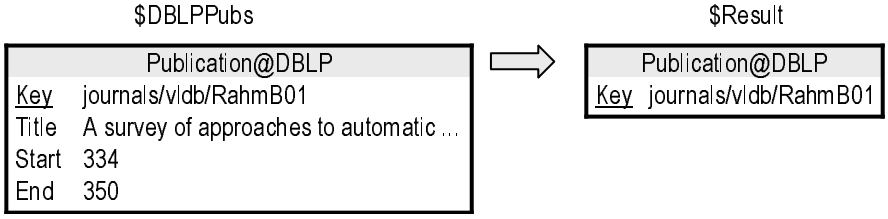
Beispiel `$Result := delAttr ($DBLPPubs, "Length")`
Das Attribut `Length` wird bei allen übergebenen DBLP-Publikationen gelöscht.



Aufruf O_A `= delAttr (`O_A`)`

Beschreibung Alle Attribute – mit Ausnahme des ID-Attributs – werden in allen übergebenen Objekten gelöscht.

Beispiel `$Result := delAttr ($DBLPPubs)`
Alle Attribute – bis auf das ID-Attribut `Key` – werden bei den DBLP-Publikationen gelöscht.



Aufruf MR_{AB} `= delAttr (`MR_{AB}`, AttrName)`
MR_{AB} `= delAttr (`MR_{AB}`)`

Beschreibung Das angegebene Attribut bzw. alle Nicht-ID-Attribute werden sowohl in den Domain- als auch Range-Objekten gelöscht.

Aufruf AO_S `= delAttr (`AO_S`, AttrName)`
AO_S `= delAttr (`AO_S`)`

Beschreibung Das angegebene fusionierte Attribut bzw. alle fusionierten Attribute der übergebenen aggregierten Objekte werden gelöscht.

Aufruf AMR_{ST} `= delAttr (`AMR_{ST}`, AttrName)`
AMR_{ST} `= delAttr (`AMR_{ST}`)`

Beschreibung Das angegebene fusionierte Attribut bzw. alle fusionierten Attribute werden sowohl in den Domain- als auch Range-Objekten gelöscht.

A.5 Operatoren zur Mapping-Ausführung

Mit den Operatoren `traverse` und `map` lassen sich traversierbare Mappings ausführen. Mappings können dabei sowohl durch ihren – im Source-Mapping-Modell definierten – Namen referenziert werden, um die dem Mediator bekannten (externen) Mappings auszuführen, als auch durch Mapping Results definiert werden. Der Operator `match` ermöglicht die Ausführung von Match-Mappings. Als Spezialfall eines Match-Mappings lässt sich mittels `attrMatch` ein „eingebauter" Attribut-Matcher verwenden, der Korrespondenzen auf Grund von Ähnlichkeiten beliebiger Objektattribute mittels verschiedener Funktionen ermittelt.

A.5.1 traverse

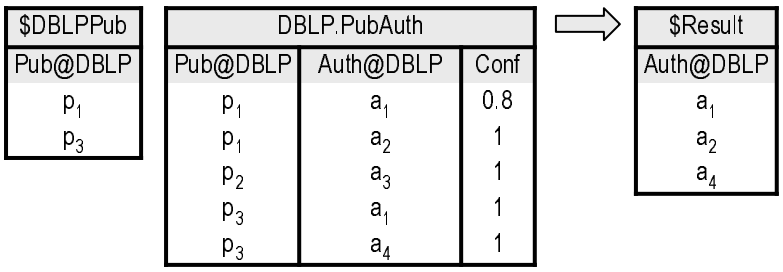
Mit Hilfe von `traverse` wird ein traversierbares Mapping traversiert, d. h. zu einer Menge von Domain-Objekten wird die Menge aller korrespondierenden Range-Objekte bestimmt. Im Folgenden werden die Mappings jeweils durch ein Mapping Result definiert. Sie können aber genauso durch ihren Namen im Source-Mapping-Modell beschrieben werden.

Aufruf O_B `= traverse (`O_A`,` MR_{AB}`)`

Definition $= \{b | a \in O_A \wedge (a, b, c) \in MR_{AB}\}$
`= range(map(`O_A`,` MR_{AB}`))`

Beschreibung Ausgehend von den Input-Objekten wird das Mapping traversiert, d. h. es wird die Menge aller korrespondierenden Objekte bestimmt.

Beispiel `$Result := traverse ($DBLPPub, DBLP.PubAuth)`
Für die übergebenen DBLP-Publikationen wird die Menge der assoziierten Autoren durch Traversierung des Mapping DBLP.PubAuth ermittelt.

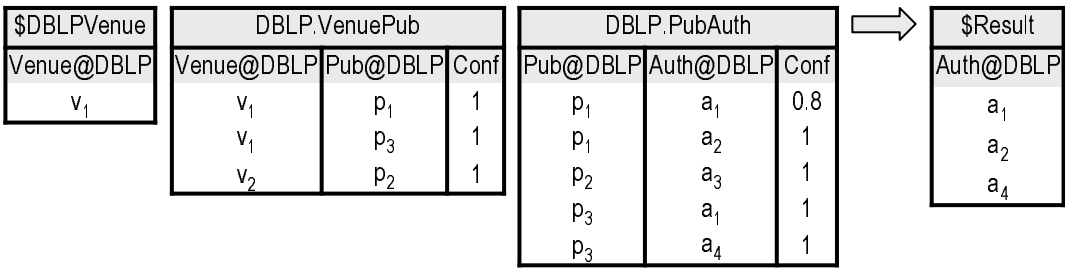


Aufruf O_{A_n} `= traverse (`O_{A_1}`,` $MR_{A_1A_2}$`,` $MR_{A_2A_3}$`, ...,` $MR_{A_{n-1}A_n}$`)`

Definition = traverse ((...traverse (traverse (O_A, $MR_{A_1A_2}$), $MR_{A_2A_3}$) ...), $MR_{A_{n-1}A_n}$)
= range(map(O_{A_1}, $MR_{A_1A_2}$, $MR_{A_2A_3}$, ..., $MR_{A_{n-1}A_n}$))

Beschreibung Statt eines Mappings wird ein Mapping-Pfad traversiert, was einer sequenziellen Ausführung der einzelnen Mappings entspricht.

Beispiel
```
$Result := traverse ($DBLPVenue, DBLP.VenuePub,
                     DBLP.PubAuth)
```
Zu den übergebenen DBLP-Venues wird die Menge derjenigen Autoren bestimmt, die mindestens einmal als Autor einer Publikation eines Venues auftreten. Dazu werden die Mappings DBLP.VenuePub und DBLP.PubAuth sequenziell traversiert, d. h. zuerst wird die Menge der Publikationen bestimmt, aus denen anschließend die Menge der Autoren ermittelt wird.



Aufruf O_{A_n} = traverse (O_{A_1}, PDS_n)

Definition $= \{b | \exists MR_{A_1A_2} \dots MR_{A_{n-1}A_n}$:
$b \in$ traverse (O_{A_1}, $MR_{A_1A_2}$, ..., $MR_{A_{n-1}A_n}$) $\}$

Beschreibung Es werden alle zyklenfreie Same-Mapping-Pfade zwischen der Startdatenquelle $LDS_{A_1} \supseteq O_{A_1}$ und der Zieldatenquelle LDS_{A_n} bestimmt. Diese Pfade werden traversiert und die resultierenden Mengen von Objektinstanzen vereinigt.

Anmerkung Da Same-Mappings nur zwischen LDS des gleichen Objekttyps auftreten, ist der Objekttyp der Zieldatenquelle LDS_{A_n} bereits durch den Objekttyp von LDS_{A_1} definiert. Daher genügt zur Beschreibung der Zieldatenquelle die Angabe der PDS.

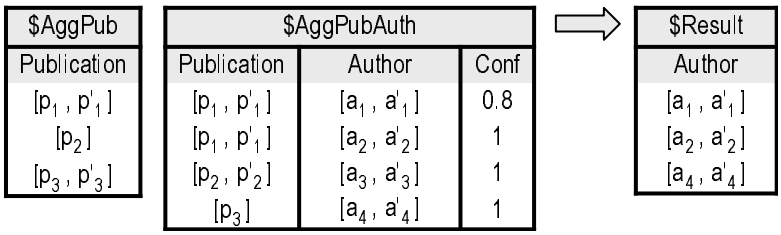
Aufruf AO_T = traverse (AO_S, AMR_{ST})
AO_T = traverse (AO_S, MapType)

Anmerkung Wird ein Mapping-Typ angegeben, so steht dieser für die Vereinigung aller aggregierten Mappings des angegebenen Typs.

Definition $= \{\hat{t} | \hat{s} \in AO_S \wedge \exists \hat{s'} : \hat{s'} \subseteq \hat{s} \wedge (\hat{s'}, \hat{t}, c) \in AMR_{ST}\}$
`= range(map(`AO_S`,` AMR_{ST}`))`

Beschreibung Ausgehend von aggregierten Input-Objekten wird ein AMR traversiert, d. h. es wird die Menge aller korrespondierenden aggregierten Objekte bestimmt. Die Zuordnung, für welches aggregierte Objekt $\hat{s} \in AO_S$ Korrespondenzen in AMR_{ST} existieren, erfolgt mit Hilfe der Teilmengenbeziehung. Das bedeutet, dass alle Objektinstanzen des aggregierten Domain-Objekts $\hat{s'}$ in $\hat{s}$ vorkommen müssen, damit das zugehörige Range-Objekt $\hat{t}$ im Ergebnis von `traverse` erscheint.

Beispiel `$Result := traverse ($AggPub, AggPubAuth)`
Für aggregierte Publikationen `$AggPub` werden die zugehörigen aggregierten Autoren bestimmt.



Aufruf AO_{S_n} `= traverse(`$AO_{S_1}, AMR_{S_1S_2}, AMR_{S_2S_3}, \ldots, AMR_{S_{n-1}S_n}$`)`

Definition `= traverse (( ...traverse (traverse (`AO_{S_1}`,` $AMR_{S_1S_2}$`),`
$AMR_{S_2S_3}$`) ...),` $AMR_{S_{n-1}S_n}$`)`
`= range(map(`$AO_{S_1}, AMR_{S_1S_2}, AMR_{S_2S_3}, \ldots, AMR_{S_{n-1}S_n}$`))`

Beschreibung Es wird – analog zu oben – eine Folge aggregierter Mappings traversiert.

A.5.2 map

Der Operator `map` wendet – analog zu `traverse` – ein oder mehrere traversierbare Mappings auf eine Menge von (aggregierten) Objekten an. Während `traverse` lediglich die korrespondierenden (aggregierten) Objekte zurückliefert, ermittelt `map` die Korrespondenzen. Wie auch bei `traverse` werden zur Vereinfachung der Darstellung Mappings durch ein Mapping Result definiert. Die Angabe eines Mapping-Namens aus dem Source-Mapping-Modell ist ebenfalls möglich.

Aufruf MR_{AA} `= map (`O_A`)`

Definition $= \{(a, a, 1) | a \in O_A\}$

Beschreibung Dieser Spezialfall von `map` ohne Angabe eines Mappings erzeugt ein Identitäts-Mapping, d. h. für jede Objektinstanz wird eine Korrespondenz zwischen der gleichen Instanz gebildet. Es dient i. Allg. als ein „Hilfs-Mapping“, z. B. im Rahmen einer Duplikaterkennung zur Entfernung trivialer Korrespondenzen zwischen gleichen Objekten bei einem erzeugten Self-Mapping, d. h. einem Same-Mapping zwischen der gleichen LDS, mittels `diff ($SelfMap, $IdentMap)`.

Beispiel `$Result := map ($DBLPPub)`
Es wird ein Mapping erzeugt, das für jede übergebene DBLP-Publikation eine Korrespondenz mit sich selbst besitzt.

$DBLPPub
Pub@DBLP
p_1
p_2
p_3

⇒

$Result		
Pub@DBLP	Pub@DBLP	Conf
p_1	p_1	1
p_2	p_2	1
p_3	p_3	1

Aufruf MR_{AB} `= map (`O_A`,` MR_{AB}`)`

Definition $= \{(a, b, c) \in MR_{AB} | a \in O_A\}$

Beschreibung Es werden diejenigen Korrespondenzen des Mappings ermittelt, deren Domain-Objekte aus der übergebenen Menge der Eingabeobjekte stammen.

Beispiel `$Result := map ($DBLPPub, DBLP.PubAuth)`
Der Aufruf führt das Mapping DBLP.PubAuth für die übergebenen DBLP-Publikationen aus. Das Ergebnis sind alle Korrespondenzen zwischen Publikationen und Autoren, die im Mapping für die übergebenen DBLP-Publikationen vorliegen.

$DBLPPub
Pub@DBLP
p_1
p_3

DBLP.PubAuth		
Pub@DBLP	Auth@DBLP	Conf
p_1	a_1	0.8
p_1	a_2	1
p_2	a_3	1
p_3	a_1	1
p_3	a_4	1

⇒

$Result		
Pub@DBLP	Auth@DBLP	Conf
p_1	a_1	0.8
p_1	a_2	1
p_3	a_1	1
p_3	a_4	1

Aufruf $MR_{A_1A_n}$ `= map (`O_{A_1}`,` $MR_{A_1A_2}$`,` $MR_{A_2A_3}$`, ...,` $MR_{A_{n-1}A_n}$`)`

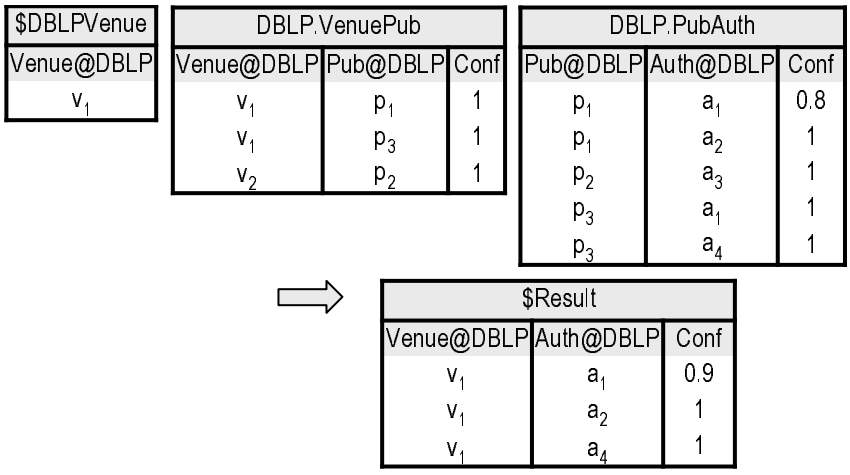
Definition = `compose( ...(compose(map(`O_{A_1}`,` $MR_{A_1A_2}$`),` $MR_{A_2A_3}$`),` `...),` $MR_{A_{n-1}A_n}$`)`

Beschreibung Die sequenzielle Ausführung von Mappings entspricht der Komposition der Mappings. Der `compose`-Operator definiert dabei auch die Bestimmung der Konfidenz, da resultierende Korrespondenzen über mehrere Kompositionspfade der Eingangs-Mappings entstehen können. Wie in der Definition angegeben, verwendet `map` den `compose`-Operator ohne weitere Parameter außer den Mappings, so dass die für `compose` definierten Standardfunktionen zur Berechnung der Konfidenz zur Anwendung kommen. Sollen spezielle Konfidenzfunktionen verwendet werden, muss statt `map` explizit `compose` mit der gewünschten Parametrisierung angegeben werden.

Beispiel

```
$Result := map ($DBLPVenue, DBLP.VenuePub, DBLP.PubAuth)
```

Es werden für die übergebenen DBLP-Venues die assoziierten Autoren durch sequenzielle Mapping-Ausführung von DBLP.VenuePub und DBLP.PubAuth ermittelt. Eine resultierende Korrespondenz zwischen Venue und Autor bedeutet, dass es mindestens eine bei dem Venue erschienene Publikation gibt, an welcher der Autor beteiligt war.



Aufruf $MR_{A_1A_n}$ `= map (`O_{A_1}`,` PDS_n`)`

Definition $= \{(a,b,c) | \exists MR_{A_1A_2} \dots MR_{A_{n-1}A_n} :$
$(a,b,c) \in$ `map (`O_{A_1}`,` $MR_{A_1A_2}$`,` `...,` $MR_{A_{n-1}A_n}$`) }`

Beschreibung Analog zu `traverse` werden alle zyklenfreie Same-Mapping-Pfade zwischen der Startdatenquelle $LDS_{A_1} \supseteq O_{A_1}$ und der Zieldaten-

quelle LDS_{A_n} bestimmt. Diese Pfade werden – im Gegensatz zu `traverse` – durch Komposition verbunden und die resultierenden Korrespondenzen zurückgeliefert.

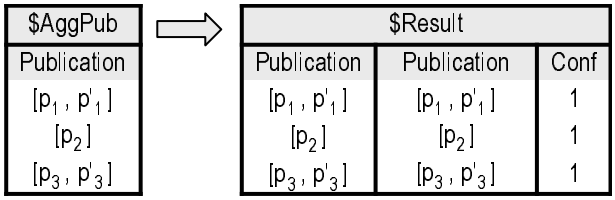
Anmerkung Da Same-Mappings nur zwischen LDS des gleichen Objekttyps auftreten, ist der Objekttyp der Zieldatenquelle LDS_{A_n} bereits durch LDS_{A_1} definiert. Daher genügt zur Beschreibung der Zieldatenquelle die Angabe der PDS.

Aufruf AMR_{SS} `= map (`AO_S`)`

Definition $= \{(\hat{s}, \hat{s}, 1) | \hat{s} \in AO_S\}$

Beschreibung Es wird ein Identitäts-Mapping für aggregierte Objekte erzeugt. Das Identitäts-Mapping wird insbesondere bei den folgenden Definitionen des `map`-Operators verwendet.

Beispiel `$Result := map ($AggPub)`
Für jedes übergebene aggregierte Objekt wird eine Korrespondenz (Konfidenzwert = 1) zwischen den gleichen Objekten gebildet.



Aufruf AMR_{ST} `= map(`AO_S`,`AMR_{ST}`)`
AMR_{ST} `= map(`AO_S`, MapType)`

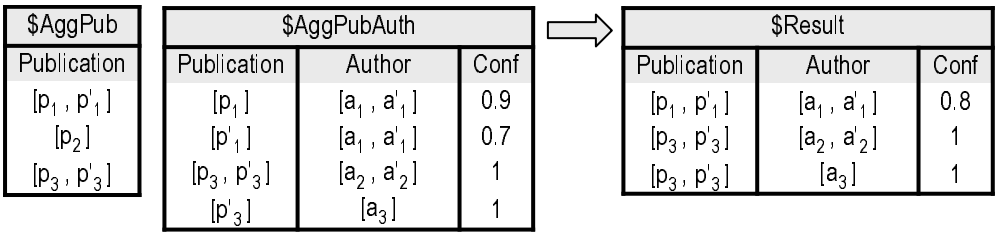
Anmerkung Wird ein Mapping-Typ angegeben, so steht dieser – wie bei `traverse` – für die Vereinigung aller aggregierten Mappings des angegebenen Typs.

Definition `= compose(map(`AO_S`),`AMR_{ST}`,`$Right$`,`Avg`)`

Anmerkung Bei aggregierten Objekten können bereits bei der Anwendung von `map` mit nur einem AMR Korrespondenzen zwischen gleichen aggregierten Objekten mehrfach entstehen. Daher verwendet die Definition von `map` das Identitäts-Mapping und den `compose`-Operator. Als Konfidenzfunktionen $\vec{h}$ (siehe Definition von `compose`) wird $Right$ verwendet, so dass die Konfidenzwerte des Eingabe-Mappings AMR_{ST} übernommen werden. Sollte eine resultierende Korrespondenz durch mehrere Pfade entstehen, wird für den Konfidenzwert

der Korrespondenz der Durchschnitt der Konfidenzwerte der Kompositionspfade gebildet ($\vec{v} = Avg$).

Beispiel `$Result := map ($AggPub, $AggPubAuth)`
Es wird das aggregierte Mapping `$AggPubAuth` für die übergebene Menge aggregierter Publikationen ausgeführt. Die Korrespondenz zwischen $[p_1, p'_1]$ und $[a_1, a'_1]$ wird über zwei Kompositionspfade erreicht, welche als Konfidenz 0.9 bzw. 0.7 besitzen. Daher hat die resultierende Korrespondenz eine Konfidenz von 0.8 (= (0.9+0.7)/2).



Aufruf AO_{S_n} `= map (`AO_{S_1}`,` $AMR_{S_1S_2}$`,` $AMR_{S_1S_2}$`, ...,` $AMR_{S_{n-1}S_n}$`)`

Definition `= compose ( ...(compose (map (`AO_{S_1}`,` $AMR_{S_1S_2}$`) ,` $AMR_{S_2S_3}$`), ...,` $AMR_{S_{n-1}S_n}$`)`

Beschreibung Analog zum Fall für MR entspricht die Ausführung von `map` mit mehreren AMR der Mapping-Komposition.

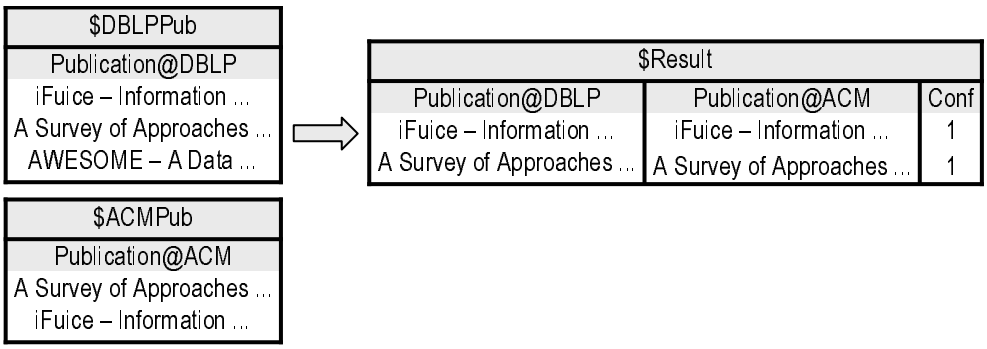
A.5.3 match

Der Operator `match` dient zum Ausführen eines Match-Mappings. Dies wird i. Allg. auch als *Matching* und das Mapping selbst als *Matcher* bezeichnet. Im Gegensatz zu traversierbaren Mappings müssen sowohl die Domain- als auch die Range-Objekte übergeben werden.

Aufruf MR_{AB} `= match(` O_A`,` O_B`, MatchName)`
MR_{AB} `= match(` MR_{AB}`, MatchName)`

Beschreibung Der Matcher `MatchName` wird für die übergebenen Objektinstanzen ausgeführt. Im ersten Fall werden zwei Objektmengen übergeben, so dass jede Objektinstanz aus O_A mit jeder aus O_B durch den Matcher verglichen wird. Im zweiten Fall wird bereits ein MR übergeben, so dass lediglich die in MR_{AB} korrespondierenden Objektinstanzen verglichen werden. In beiden Fällen liefert der Matcher ein MR zurück.

Beispiel `$Result := match($DBLPPub, $ACMPub, MatchDBLPACM)`
Es wird für die übergebenen DBLP- und ACM-Publikationen der Matcher MatchDBLPACM aufgerufen. Die Bestimmung der Korrespondenzen wird in der Implementation des Matchers festgelegt. Im Beispiel werden Korrespondenzen z. B. auf Grund der Ähnlichkeit des Publikationstitels gebildet.



A.5.4 attrMatch

Ein generischer Attribut-Matcher steht bereits in iFuice zur Verfügung. Dieser wird mittels `attrMatch` aufgerufen. Dabei können die zu vergleichenden Attribute und das anzuwendende Stringähnlichkeitsmaß angegeben werden. Um die Anzahl der resultierenden Korrespondenzen zu begrenzen, kann optional eine minimale Konfidenz angegeben werden.

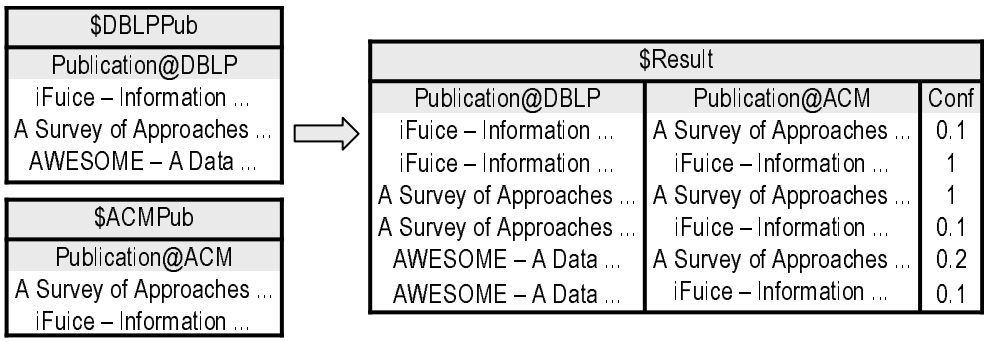
Aufruf
MR_{AB} = `attrMatch` $(O_A, O_B, Sim, Attr_A, Attr_B)$
MR_{AB} = `attrMatch` $(O_A, O_B, Sim, Attr_A, Attr_B, MinConf)$
MR_{AB} = `attrMatch` $(MR_{AB}, Sim, Attr_A, Attr_B)$
MR_{AB} = `attrMatch` $(MR_{AB}, Sim, Attr_A, Attr_B, MinConf)$
AMR_{ST} = `attrMatch` $(AO_S, AO_T, Sim, Attr_S, Attr_T)$
AMR_{ST} = `attrMatch` $(AO_S, AO_T, Sim, Attr_S, Attr_T, MinConf)$
AMR_{ST} = `attrMatch` $(AMR_{ST}, Sim, Attr_S, Attr_T)$
AMR_{ST} = `attrMatch` $(AMR_{ST}, Sim, Attr_S, Attr_T, MinConf)$

Anmerkung In der derzeitigen Implementation stehen als Ähnlichkeitsfunktionen für Sim die Maße `Trigram`, `TFIDF` und `EditDistance` zur Verfügung. Weitere Ähnlichkeitsmaße können einfach hinzugefügt werden, ohne die Signatur des Operators zu verändern.

Beschreibung Es wird der eingebauten Attribut-Matcher ausgeführt. Dabei werden die Attributwerte $Attr_A$ und $Attr_B$ bei Objektinstanzen bzw. die fusionierten Attribute $Attr_S$ und $Attr_T$ bei aggregierten Objekten mittels der Ähnlichkeitsfunktion Sim verglichen und der er-

rechnete Ähnlichkeitswert als Konfidenz für die resultierende Korrespondenz verwendet wird. Bei Angabe von $MinConf$ werden lediglich alle Korrespondenzen mit einer Konfidenz $c \geq MinConf$ zurückgegeben. Der `attrMatch`-Operator realisiert einen „fuzzy join“, bei dem Korrespondenzen auf Grund von Attributwertähnlichkeiten entstehen. Im Gegensatz dazu steht der `join`-Operator, der Korrespondenzen auf Basis einer zu prüfenden Bedingung (z. B. Gleichheit von Attributwerten) ermittelt.

Beispiel
```
$Result := attrMatch($DBLPPub, $ACMPub, Trigram,
                     "[DBLP.title]", "[ACM.name]")
```
Für die übergebenen DBLP- und ACM-Publikationen wird die Trigram-Ähnlichkeit der Publikationstitel bestimmt. Da keine minimale Konfidenz angegeben wurde, enthält das resultierende Mapping alle möglichen 6 (= 3 Domain-Objekte · 2 Range-Objekte) Korrespondenzen zwischen den Eingabeobjekten. Der Konfidenzwert entspricht der Trigram-Ähnlichkeit. (In der Abbildung wurden aus Gründen der Übersichtlichkeit approximierte Ähnlichkeitswerte angegeben.)



A.6 Operatoren zur Aggregation

Die Bildung aggregierter Objekte ermöglicht die Zusammenfassung von als *gleich* angesehenen Objektinstanzen zu einer Einheit, um eine gemeinsame Weiterverarbeitung, z. B. bei der Mapping-Ausführung, zu gewährleisten. Aggregierte Objekte werden mittels `aggregate` erstellt. Die Umkehrung, d. h. die Extraktion von Instanzen aus aggregierten Objekten, wird durch `disAgg` realisiert. Entsprechend werden MR und AMR ineinander überführt. Zentrales Element bei der Aggregation ist dabei die Definition, welche Objektinstanzen zu einem aggregierten Objekt zusammengefasst werden. Diese Zuordnung wird durch Korrespondenzen von Same-Mappings definiert, so dass Objektinstanzen des gleichen Objekttyps (z. B. Publikation) zusammengefasst werden, wenn sie als gleich erkannt wurden.

A.6.1 aggregate

Der Operator `aggregate` konvertiert Objektinstanzen zu aggregierten Objekten.

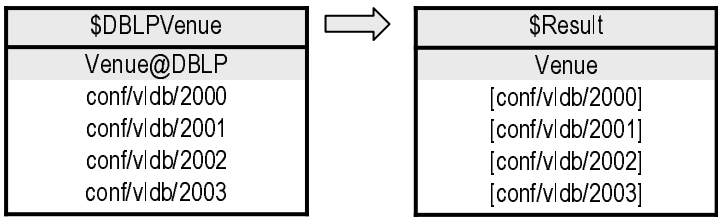
Aufruf AO_S `= aggregate(`O_A`)`

Anmerkung $LDS_A \supseteq O_A$ hat den Objekttyp S.

Definition $= \{\{a\} \,|a \in O_A\}$

Beschreibung Jede Objektinstanz wird in ein aggregiertes Objekt konvertiert, das genau aus einer Instanz besteht. Dieser Spezialfall aggregierter Objekte wird u.a. zur späteren Definition von `aggregate` benötigt.

Beispiel `$Result := aggregate($DBLPVenue)`
Die übergebenen DBLP-Venues werden in aggregierte Objekte konvertiert.



$DBLPVenue
Venue@DBLP
conf/vldb/2000
conf/vldb/2001
conf/vldb/2002
conf/vldb/2003

$Result
Venue
[conf/vldb/2000]
[conf/vldb/2001]
[conf/vldb/2002]
[conf/vldb/2003]

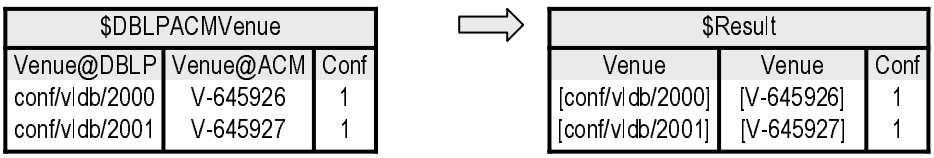
Aufruf AMR_{ST} `= aggregate(`MR_{AB}`)`

Anmerkung LDS_A hat den Objekttyp S. LDS_B hat den Objekttyp T.

Definition $= \{(\{a\}, \{b\}, c) \,|\, (a, b, c) \in MR_{AB}\}$

Beschreibung Die jeweiligen Objektinstanzen werden in aggregierte Objekte umgewandelt. Auch dieser Spezialfall dient zu späteren Definitionszwecken.

Beispiel `$Result := aggregate($DBLPACMVenue)`
Die Korrespondenzen des übergebenen Mappings werden in Korrespondenzen zwischen aggregierten Objekten umgewandelt.



$DBLPACMVenue		
Venue@DBLP	Venue@ACM	Conf
conf/vldb/2000	V-645926	1
conf/vldb/2001	V-645927	1

$Result		
Venue	Venue	Conf
[conf/vldb/2000]	[V-645926]	1
[conf/vldb/2001]	[V-645927]	1

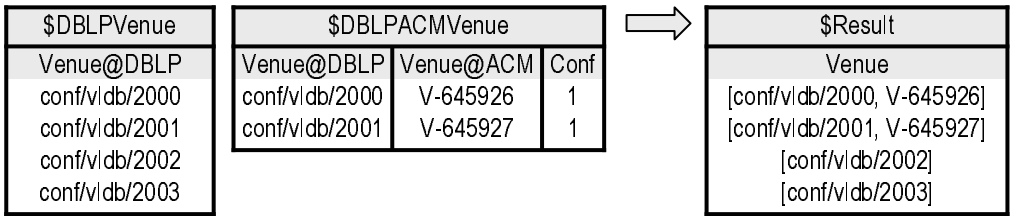
Aufruf AO_S = `aggregate` (O_A,MR_{AB})

Anmerkung $LDS_A \supseteq O_A$ und LDS_B haben jeweils den Objekttyp S.

Definition = `aggregate(` `aggregate`(O_A), MR_{AB})

Beschreibung Das übergebene Mapping MR_{AB} ist ein Same-Mapping zwischen LDS_A und LDS_B, die beide den Objekttyp S besitzen. Durch `aggregate` werden die im Same-Mapping durch Korrespondenzen verbundenen Objektinstanzen zu jeweils einem aggregierten Objekt zusammengefasst. Dazu werden zunächst die Objektinstanzen O_A in aggregierte Objekte umgewandelt. Anschließend werden jedem aggregierten Objekt diejenigen Objektinstanzen aus LDS_B hinzugefügt, die eine Korrespondenz in MR_{AB} zur Objektinstanz aus LDS_A besitzen.

Beispiel `$Result := aggregate($DBLPVenue, $DBLPACMVenue)`
Für die übergebenen DBLP-Venues werden mit Hilfe eines Venue-Same-Mappings zwischen DBLP und ACM aggregierte Objekte gebildet. Da nicht für alle DBLP-Venues eine Korrespondenz im Same-Mapping vorliegt, ergeben sich neben aggregierten Objekten bestehend aus je einem DBLP- und ACM-Venue auch aggregierte Objekte, die nur aus einem DBLP-Venue bestehen.



Aufruf AO_S = `aggregate`(AO_S,MR_{AB})

Anmerkung Die Domain-LDS LDS_A sowie die Range-LDS LDS_B des übergebenen Same-Mappings besitzen jeweils den Objekttyp S.

Definition $= \{\{\hat{s} \cup \{b_1\} \cup \ldots \cup \{b_n\}\} \,|\hat{s} \in AO_S$
$\wedge \forall 1 \leq k \leq n \exists a_i \in \hat{s}, c_l \in [0,1] : (a_i, b_k, c_l) \in MR_{AB}\}$

Beschreibung Den übergebenen aggregierten Objekten AO_S werden Objektinstanzen auf Grund des Same-Mappings MR_{AB} hinzugefügt. Eine Objektinstanz $b_k \in LDS_B$ wird genau dann einem der aggregierten Objekte $\hat{s}$ hinzugefügt, wenn es mindestens ein $a_i \in \hat{s}$ gibt, das eine Korrespondenz zu b_k im Mapping besitzt.

Aufruf AMR_{ST} = `aggregate` (MR_{AB}, MR_{AC})

Anmerkung LDS_A und LDS_C besitzen den Objekttyp S und MR_{AC} ist ein Same-Mapping. LDS_B hat den Objekttyp T.

Definition = `aggregate (aggregate` (MR_{AB}), MR_{AC})

Beschreibung Aus dem übergebenen MR_{AB} wird ein AMR gebildet. Zusätzlich werden den aggregierten Domain-Objekten mit Hilfe des Same-Mappings MR_{AC} *gleiche* Objektinstanzen hinzugefügt.

Aufruf AMR_{ST} = `aggregate` (AMR_{ST}, MR_{AB})

Anmerkung Die Domain-LDS LDS_A sowie die Range-LDS LDS_B des übergebenen Same-Mappings besitzen jeweils den Objekttyp S.

Definition = `map (aggregate (domain(`AMR_{ST}`),` MR_{AB}`),` AMR_{ST})

Beschreibung Bei der Aggregation eines aggregierten Mappings werden lediglich die Domain-Objekte aggregiert.[69] Durch die Aggregation der Domain-Objekte kann es vorkommen, dass gleiche Korrespondenzen entstehen, die zusammengefasst werden. Die Bestimmung der resultierenden Konfidenzwerte wird durch den Operator `map` realisiert.

Beispiel `$Result := aggregate ($PubVenue, $DBLPACMPub)`
Das aggregierte Mapping `$PubVenue` wird durch das Same-Mapping `$DBLPACM` „erweitert“, indem gleiche Publikationen zu aggregierten Objekten zusammengefasst werden. Durch die Aggregation fallen je zwei Korrespondenzen aus `$PubVenue` zusammen, so dass im resultierenden AMR nur noch 3 statt 5 Korrespondenzen auftreten

$PubVenue		
Publication	Venue	Conf
[conf/vldb/StohrMR00]	[conf/vldb/2000, V-645926]	1
[P-671843]	[conf/vldb/2000, V-645926]	1
[conf/vldb/StohrR01]	[conf/vldb/2001, V-645927]	1
[conf/vldb/MadhavanBR01]	[conf/vldb/2001, V-645927]	1
[P-672191]	[conf/vldb/2001, V-645927]	1

$DBLPACMPub		
Publications@DBLP	Publication@ACM	Conf
conf/vldb/StohrMR00	P-671843	1
conf/vldb/MadhavanBR01	P-672191	1

⇨

$Result		
Publication	Venue	Conf
[conf/vldb/StohrMR00, P-671843]	[conf/vldb/2000, V-645926]	1
[conf/vldb/StohrR01]	[conf/vldb/2001, V-645927]	1
[conf/vldb/MadhavanBR01, P-672191]	[conf/vldb/2001, V-645927]	1

[69] Sollen auch die Range-Objekte aggregiert werden, so muss `aggregate` erneut mit dem inversen Mapping (siehe Operator `inverse`) ausgeführt werden.

A.6.2 disAgg

Der Operator `disAgg` wandelt aggregierte Objekte wieder in Objektinstanzen um.

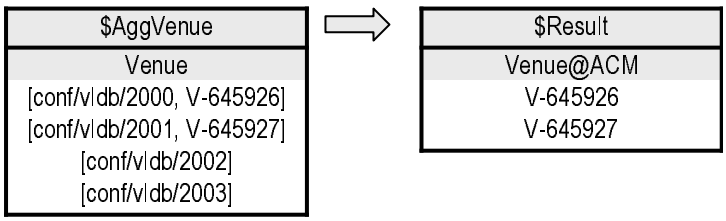
Aufruf O_A = `disAgg(`AO_S`,` PDS_A`)`

Anmerkung LDS_A hat den Objekttyp S.

Definition $= \{a | a \in PDS_A \wedge a \in \hat{s} \wedge \hat{s} \in AO_S\}$

Beschreibung Aus der Menge der übergebenen aggregierten Objekte werden diejenigen Objektinstanzen extrahiert, welche aus der angegebenen Datenquelle stammen. Die Angabe der physischen Datenquelle ist ausreichend, da alle aggregierten Objekte den gleichen Objekttyp haben.

Beispiel `$Result := disAgg($AggVenue, ACM)`
Aus den aggregierten Objekten, die DBLP- und ACM-Venues beinhalten, werden die ACM-Instanzen extrahiert.



Aufruf MR_{AB} = `disAgg(`AMR_{ST}`,` PDS_A`,` PDS_B`)`
MR_{AB} = `disAgg(`AMR_{ST}`,` PDS_A`,` PDS_B`,` $\vec{v}$`)`

Anmerkung LDS_A hat den Objekttyp S. LDS_B hat den Objekttyp T.

Definition $= \{(a, b, \vec{v}\{c_i\}) | a \in PDS_A \wedge a \in \hat{s_i} \wedge b \in PDS_B$
$\wedge b \in \hat{t_i} \wedge (\hat{s_i}, \hat{t_i}, c_i) \in AMR_{ST}\}$

Beschreibung Aus der Menge der übergebenen Korrespondenzen zwischen aggregierten Objekte werden diejenigen Korrespondenzen zwischen Objektinstanzen extrahiert, bei denen die korrespondierenden Objekte aus den angegebenen Datenquellen stammen. Auch hier ist die Angabe der physischen Datenquelle ausreichend. Da eine Korrespondenz zwischen zwei Objektinstanzen in mehreren Korrespondenzen aggregierter Objekte vorkommen kann, muss eine Konfidenzfunktion $\vec{v}$ angewendet werden (mögliche Funktionen: Min, Avg (Default), Max), die aus den Konfidenzen der betreffenden

aggregierten Korrespondenzen die Konfidenz für die resultierende Korrespondenz bestimmt.

Beispiel `$Result := disAgg($AggPubConf, DBLP, ACM)`
Es werden Korrespondenzen zwischen DBLP-Publikationen und ACM-Venues erstellt. Eine Korrespondenz zwischen einer DBLP-Publikation p und einem ACM-Venue v entsteht genau dann, wenn p und v jeweils in einem aggregierten Objekt auftreten, die in `$AggPubConf` eine Korrespondenz besitzen.

$AggPubVenue		
Publication	Venue	Conf
[conf/vldb/StohrMR00, P-671843]	[conf/vldb/2000, V-645926]	1
[conf/vldb/StohrR01]	[conf/vldb/2001, V-645927]	1
[conf/vldb/MadhavanBR01, P-672191]	[conf/vldb/2001, V-645927]	1

⇨

$Result		
Publication@DBLP	Venue@ACM	Conf
conf/vldb/StohrMR00	V-645926	1
conf/vldb/StohrR01	V-645927	1
conf/vldb/MadhavanBR01	V-645927	1

A.7 Kombinationsoperatoren

Mit Hilfe der Operatoren `diff`, `intersect`, `union` und `compose` können Objekte und Korrespondenzen kombiniert werden. Bei der Kombination von MR und AMR muss – außer bei `diff` – für die resultierenden Korrespondenzen definiert werden, wie deren Konfidenzwert aus den Konfidenzen der Ursprungskorrespondenzen bestimmt wird. Dazu werden Konfidenzfunktionen als optionale (Default: Avg) Parameter übergeben, die in Kapitel 9.3 ausführlich vorgestellt wurden.

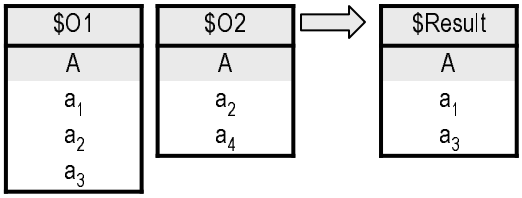
A.7.1 diff

Der `diff`-Operator erzeugt die Differenzmenge von Objekten und Korrespondenzen.

Aufruf O_A `= diff(`O_{1_A}`,` O_{2_A}`, ...,` O_{n_A}`)`

Definition $= \{a | a \in O_{1_A} \wedge \forall 2 \leq i \leq n : a \notin O_{i_A}\}$

Beispiel `$Result := diff($O1, $O2)`



Aufruf MR_{AB} `= diff(`$MR_{1_{AB}}$`,` $MR_{2_{AB}}$`, ...,` $MR_{n_{AB}}$`)`

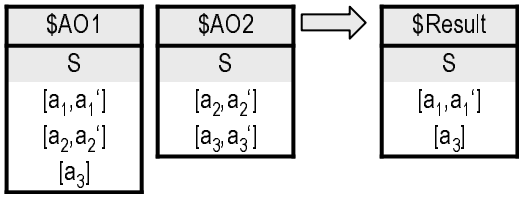
Definition $= \{(a,b,c) \,|\, (a,b,c) \in MR_{1_{AB}} \wedge \forall 2 \leq i \leq n : \exists c' : (a,b,c') \in MR_{i_{AB}}\}$

Beispiel `$Result := diff($MR1, $MR2)`

Aufruf AO_S `= diff(`AO_{1_S}`,` AO_{2_S}`, ...,` AO_{n_S}`)`

Definition $= \{\hat{s} | \hat{s} \in AO_{1_S} \wedge \forall 2 \leq i \leq n : \hat{s} \notin AO_{i_S}\}$

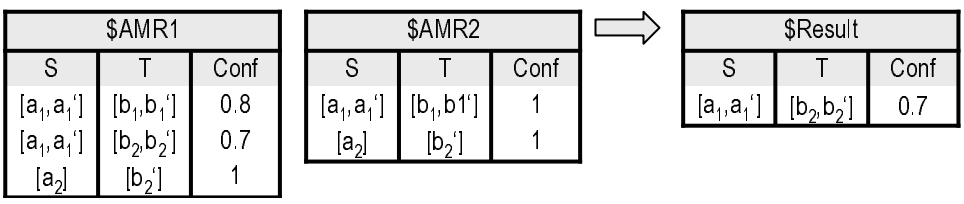
Beispiel `$Result := diff($AO1, $AO2)`



Aufruf AMR_{ST} `= diff(`$AMR_{1_{ST}}$`,` $AMR_{2_{ST}}$`, ...,` $AMR_{n_{ST}}$`)`

Definition $= \{(s,t,c) \,|\, (s,t,c) \in AMR_{1_{ST}} \wedge \forall 2 \leq i \leq n : \neg\exists c' : (s,t,c') \in AMR_{i_{ST}}\}$

Beispiel `$Result := diff($AMR1, $AMR2)`



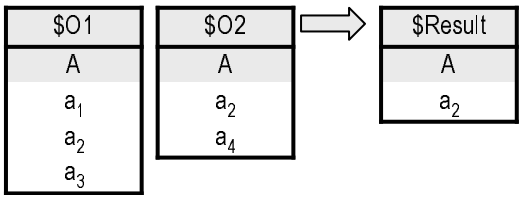
A.7.2 intersect

Der Operator `intersect` bestimmt den Durchschnitt von Objekten bzw. Korrespondenzen, d. h. alle Objekte bzw. Korrespondenzen, die in allen Eingabewerten auftreten.

Aufruf O_A `= intersect(`$O_{1_A}, \ldots, O_{n_A}$`)`

Definition $= \{a | \forall 1 \leq i \leq n : a \in O_{i_A}\}$

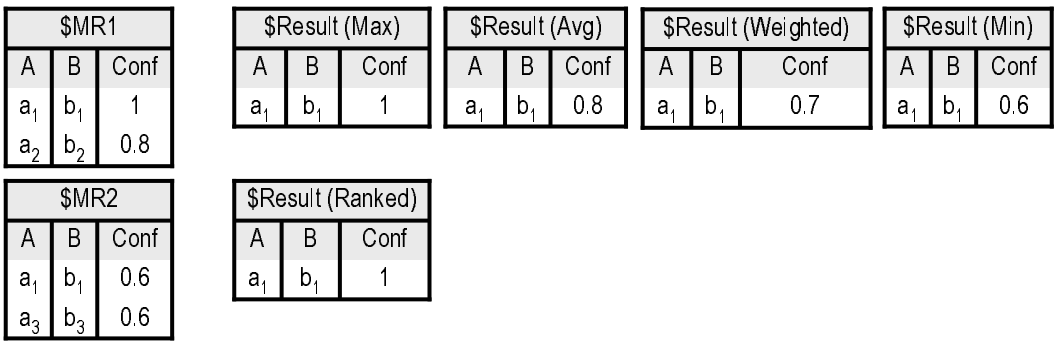
Beispiel `$Result := intersect($O1, $O2)`



Aufruf MR_{AB} `= intersect(`$MR_{1_{AB}}, \ldots, MR_{n_{AB}}$`)`
MR_{AB} `= intersect(`$MR_{1_{AB}}, \ldots, MR_{n_{AB}}, \vec{h}$`)`

Definition $= \left\{ \left(a, b, \vec{h}\left(\{c_i\}\right) \right) | \forall 1 \leq i \leq n : (a, b, c_i) \in MR_{i_{AB}} \right\}$

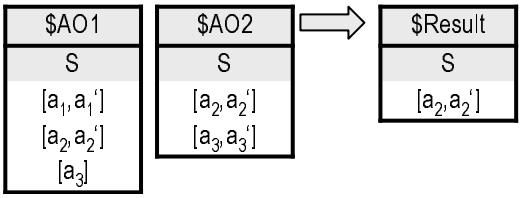
Beispiel `$Result := intersect($MR1, $MR2,` $\vec{v}$`)`
Das Bild zeigt das Ergebnis für alle verschiedenen Konfidenzfunktionen. Bei $\vec{h} = \textit{Weighted}$ wurden die Gewichte 1 und 3 verwendet. Die korrespondierenden Objektinstanzen sind in allen Fällen gleich, da die Konfidenzfunktion nur den Wert der Konfidenz beeinflusst.



Aufruf AO_S `= intersect(`$AO_{1_S}, \ldots, AO_{n_S}$`)`

Definition $= \{\hat{s} | \forall 1 \leq i \leq n : \hat{s} \in AO_{i_S}\}$

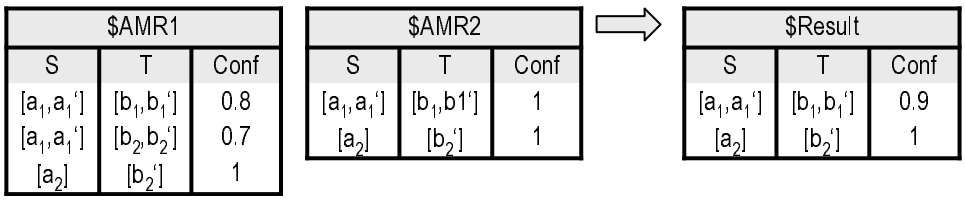
Beispiel `$Result := intersect($AO1, $AO2)`



Aufruf AMR_{ST} `= intersect(`$AMR_{1_{ST}}, \ldots, AMR_{n_{ST}}$`)`
AMR_{ST} `= intersect(`$AMR_{1_{ST}}, \ldots, AMR_{n_{ST}}, \vec{h}$`)`

Definition $= \left\{ \left(\hat{s}, \hat{t}, \vec{h}\left(\{c_i\}\right) \right) | \forall 1 \leq i \leq n : \left(\hat{s}, \hat{t}, c_i\right) \in AMR_{i_{ST}} \right\}$

Beispiel `$Result := intersect($AMR1, $AMR2)`
Im Beispiel wird keine Konfidenzfunktion angegeben, so dass standardmäßig Avg verwendet wird.



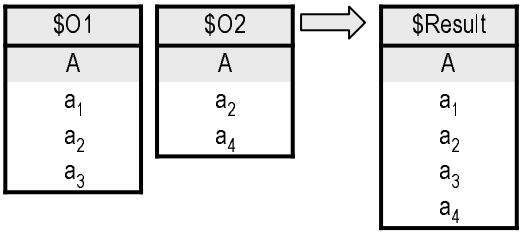
A.7.3 union

Die Vereinigung von Objekten bzw. Korrespondenzen ermittelt der Operator `union`.

Aufruf O_A `= union(`$O_{1_A}, \ldots, O_{n_A}$`)`

Definition $= \{a | \exists 1 \leq i \leq n : a \in O_{i_A}\}$

Beispiel `$Result := union($O1, $O2)`



Aufruf MR_{AB} `= union(`$MR_{1_{AB}}, \ldots, MR_{n_{AB}}$`)`
MR_{AB} `= union(`$MR_{1_{AB}}, \ldots, MR_{n_{AB}}, \vec{h}$`)`

Definition $= \left\{ \left(a, b, \vec{h}\left(\{c_i\}\right) \right) | \exists 1 \leq i \leq n : (a, b, c_i) \in MR_{i_{AB}} \right\}$

Beispiel `$Result := union($MR1, $MR2)`
Das Beispiel zeigt das Ergebnis von `union` mit allen zur Verfügung stehenden Konfidenzfunktionen. Bei der Konfidenzfunktion *Weighted* wurden, analog zu obigem `intersect`-Beispiel, ebenfalls die Gewichte 1 und 3 verwendet.

$MR1		
A	B	Conf
a_1	b_1	1
a_2	b_2	0.8

$Result (Max)		
A	B	Conf
a_1	b_1	1
a_2	b_2	0.8
a_3	b_3	0.6

$Result (Avg)		
A	B	Conf
a_1	b_1	0.8
a_2	b_2	0.8
a_3	b_3	0.6

$Result (Weighted)		
A	B	Conf
a_1	b_1	0.7
a_2	b_2	0.8
a_3	b_3	0.6

$Result (Min)		
A	B	Conf
a_1	b_1	0.6
a_2	b_2	0.8
a_3	b_3	0.6

$MR2		
A	B	Conf
a_1	b_1	0.6
a_3	b_3	0.6

$Result (Ranked)		
A	B	Conf
a_1	b_1	1
a_2	b_2	0.8
a_3	b_3	0.6

$Result (Avg-0)		
A	B	Conf
a_1	b_1	0.8
a_2	b_2	0.4
a_3	b_3	0.3

$Result (Weighted-0)		
A	B	Conf
a_1	b_1	0.7
a_2	b_2	0.2
a_3	b_3	0.45

$Result (Min-0)		
A	B	Conf
a_1	b_1	0.6
a_2	b_2	0
a_3	b_3	0

Aufruf AO_S `= union(`$AO_{1_S}, \ldots, AO_{n_S}$`)`

Definition $= \{\hat{s} | \exists 1 \leq i \leq n : \hat{s} \in AO_{i_S}\}$

Beispiel `$Result := union($AO1, $AO2)`

$AO1
S
[a_1,a_1']
[a_2,a_2']
[a_3]

$AO2
S
[a_2,a_2']
[a_3,a_3']

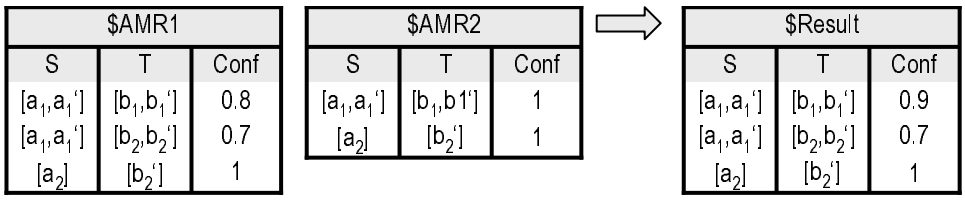
⇨

$Result
S
[a_1,a_1']
[a_2,a_2']
[a_3]
[a_3,a_3']

Aufruf AMR_{ST} `= union(`$AMR_{1_{ST}}, \ldots, AMR_{n_{ST}}$`)`
AMR_{ST} `= union(`$AMR_{1_{ST}}, \ldots, AMR_{n_{ST}}, \vec{h}$`)`

Definition $= \left\{ \left(\hat{s}, \hat{t}, \vec{h}\left(\{c_i\}\right) \right) | \exists 1 \leq i \leq n : \left(\hat{s}, \hat{t}, c_i\right) \in AMR_{i_{ST}} \right\}$

Beispiel `$Result := union($AMR1, $AMR2)`
Die Ausführung von `union` ohne Angabe einer Konfidenzfunktion verwendet die Standardfunktion *Avg*.



$AMR1		
S	T	Conf
[a_1,a_1']	[b_1,b_1']	0.8
[a_1,a_1']	[b_2,b_2']	0.7
[a_2]	[b_2']	1

$AMR2		
S	T	Conf
[a_1,a_1']	[b_1,b1']	1
[a_2]	[b_2']	1

$Result		
S	T	Conf
[a_1,a_1']	[b_1,b_1']	0.9
[a_1,a_1']	[b_2,b_2']	0.7
[a_2]	[b_2']	1

A.7.4 compose

Der `compose`-Operator bildet die Komposition zweier Mappings. Die Eingabe sind ein Mapping MR_{AX} von LDS_A nach LDS_X sowie ein zweites Mapping MR_{XB} von LDS_X nach LDS_B. Die Komposition bildet nun ein Mapping MR_{AB} von LDS_A nach LDS_B, welches Korrespondenzen zwischen $a \in LDS_A$ und $b \in LDS_B$ enthält, für die es mindestens ein $x \in LDS_X$ gibt, so dass $(a, x, c_1) \in MR_{AX}$ und $(x, b, c_2) \in MR_{XB}$.

Der Kompositionspfad p sei für zwei Mappings MR_{AX} und MR_{XB} unter Verwendung einer horizontalen Konfidenzfunktion $\vec{h}$ wie folgt definiert:

$$p = \left(a, x, b, \vec{h}\,(c_1, c_2)\right)$$

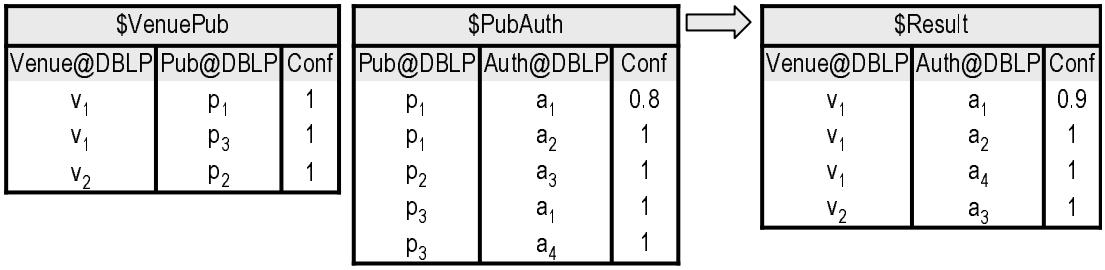
wobei $(a, x, c_1) \in MR_{AX} \wedge (x, b, c_2) \in MR_{XB}$ gilt. $P\,(MR_{AX}, MR_{XB})$ bezeichne die Menge aller Kompositionspfade p für zwei gegebene Mappings.

Aufruf MR_{AB} `= compose(`MR_{AX}, MR_{XB}`)`
MR_{AB} `= compose(`$MR_{AX}, MR_{XB}, \vec{h}, \vec{v}$`)`

Definition $= \{(a, b, \vec{v}\,(\{c_k\}))\,|\forall k \exists x_k : (a, x_k, b, c_k) \in P\,(MR_{AX}, MR_{XB})\}$

Beschreibung Es werden alle Kompositionspfade gebildet, deren Konfidenz sich durch die Funktion $\vec{h}$ – angewendet auf die beiden Konfidenzwerte der beteiligten Korrespondenzen – ergibt. Alle Pfade mit dem gleichen Domain-Objekt a und dem gleichen Range-Objekt b werden zu einer Korrespondenz zusammengefasst. Die resultierende Konfidenz wird mit Hilfe von $\vec{v}$ bestimmt. Die verfügbaren Konfidenzfunktionen $\vec{h}$ und $\vec{v}$ werden in Abschnitt 9.3.2 detailliert vorgestellt.

Beispiel `$Result := compose ($VenuePub, $PubAuth)`
Die Komposition der Assoziations-Mappings zwischen Venue und Publikation (`$VenuePub`) sowie zwischen Publikation und Autor (`$PubAuth`) erzeugt ein Mapping zwischen Venue und Autoren. Es enthält eine Korrespondenz zwischen einem Venue und einem Autor genau dann, wenn es mindestens eine Publikation gibt, die in den beiden Eingabe-Mappings jeweils eine Korrespondenz zum Venue bzw. Autor hat.



$VenuePub		
Venue@DBLP	Pub@DBLP	Conf
v_1	p_1	1
v_1	p_3	1
v_2	p_2	1

$PubAuth		
Pub@DBLP	Auth@DBLP	Conf
p_1	a_1	0.8
p_1	a_2	1
p_2	a_3	1
p_3	a_1	1
p_3	a_4	1

$Result		
Venue@DBLP	Auth@DBLP	Conf
v_1	a_1	0.9
v_1	a_2	1
v_1	a_4	1
v_2	a_3	1

Ein Compose-Pfad $\hat{p}$ sei für zwei aggregierte Mappings AMR_{SZ} und AMR_{ZT} wie folgt definiert:

$$\hat{p} = \left(\hat{s}, \hat{z}_1, \hat{z}_2, \hat{t}, \vec{h}\,(c_1, c_2)\right)$$

wobei $(\hat{s}, \hat{z}_1, c_1) \in AMR_{SZ} \wedge \left(\hat{z}_2, \hat{t}, c_2\right) \in AMR_{ZT} \wedge \hat{z}_1 \supseteq \hat{z}_2$ gilt.

Analog bezeichne $\hat{P}\,(AMR_{S_1S_2}, AMR_{S_2S_3})$ die Menge aller Compose-Pfade $\hat{p}$.

Aufruf AMR_{ST} `= compose(`AMR_{SZ}, AMR_{ZT}`)`
AMR_{ST} `= compose(`$AMR_{SZ}, AMR_{ZT}, \vec{h}, \vec{v}$`)`

Definition $= \{\left(\hat{s}, \hat{t}, \vec{v}\,(\{c_k\})\right) | \forall k \exists\,(\hat{z}_{1_k}, \hat{z}_{2_k}) :$
$\left(\hat{s}, \hat{z}_{1_k}, \hat{z}_{2_k}, \hat{t}, c_k\right) \in \hat{P}\,(AMR_{SZ}, AMR_{ZT})\}$

Beschreibung Die Komposition berechnet sich analog zu den nicht aggregierten Mappings. Der einzige Unterschied besteht in der verwendeten Teilmengensemantik, um die Komponierbarkeit von Korrespondenzen zu definieren. Dadurch beinhalten Kompositionspfade aggregierter Mappings vier aggregierte Objekte, von denen $\hat{z}_2$ eine Teilmenge von $\hat{z}_1$ ist, d. h. alle in $\hat{z}_2$ zusammengefassten Objektinstanzen treten auch in $\hat{z}_1$ auf.

A.8 Hilfsoperatoren

Hilfsoperatoren bieten einfache Basisfunktionalitäten und vereinfachen die Erstellung von Skripten. Die Operatoren `domain` und `range` extrahieren alle Domain- bzw. Range-Objekte aus einem gegebenen Mapping. Die Bestimmung des inversen Mappings durch Vertauschen von Domain- und Range-Objekt bei jeder Korrespondenz wird durch den Operator `inverse` ermöglicht. Der Operator `exec` dient der Ausführung von iFuice-Prozeduren.

A.8.1 domain

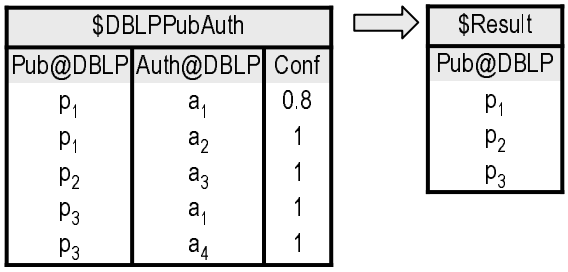
Der Operator `domain` ermittelt alle Domain-Objekte eines übergebenen (aggregierten) Mappings.

Aufruf O_A `= domain (`MR_{AB}`)`

Definition $= \{a|(a,b,c) \in MR_{AB}\}$

Beschreibung Es werden alle Domain-Objekte eines MR ermittelt.

Beispiel `$Result := domain ($DBLPPubAuth)`
Alle DBLP-Publikationen des Mappings `$DBLPPubAuth` werden bestimmt.



$DBLPPubAuth		
Pub@DBLP	Auth@DBLP	Conf
p_1	a_1	0.8
p_1	a_2	1
p_2	a_3	1
p_3	a_1	1
p_3	a_4	1

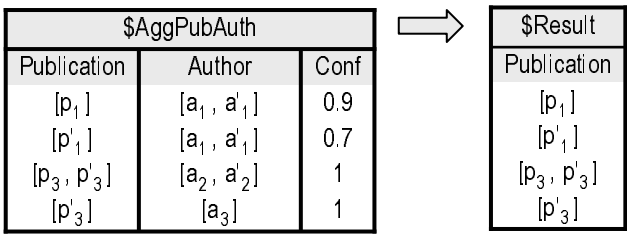
$Result
Pub@DBLP
p_1
p_2
p_3

Aufruf AO_S `= domain (`AMR_{ST}`)`

Definition $= \{\hat{s}|(\hat{s},\hat{t},c) \in AMR_{ST}\}$

Beschreibung Es werden alle aggregierten Domain-Objekte eines AMR ermittelt.

Beispiel `$Result := domain ($AggPubAuth)`
Für das Eingabe-Mapping `$AggPubAuth` werden alle aggregierte Publikationen bestimmt.



$AggPubAuth		
Publication	Author	Conf
$[p_1]$	$[a_1, a'_1]$	0.9
$[p'_1]$	$[a_1, a'_1]$	0.7
$[p_3, p'_3]$	$[a_2, a'_2]$	1
$[p'_3]$	$[a_3]$	1

$Result
Publication
$[p_1]$
$[p'_1]$
$[p_3, p'_3]$
$[p'_3]$

A.8.2 range

Der Operator `range` ermittelt alle Range-Objekte eines übergebenen (aggregierten) Mappings.

Aufruf O_B `= range (`MR_{AB}`)`

Definition $= \{b|(a,b,c) \in MR_{AB}\}$

Beschreibung Es werden alle Range-Objekte eines MR ermittelt.

Beispiel `$Result := range ($DBLPPubAuth)`
Alle DBLP-Autoren des Eingabe-Mappings `$DBLPPubAuth` werden ermittelt.

$DBLPPubAuth		
Pub@DBLP	Auth@DBLP	Conf
p_1	a_1	0.8
p_1	a_2	1
p_2	a_3	1
p_3	a_1	1
p_3	a_4	1

⇨

$Result
Auth@DBLP
a_1
a_2
a_3
a_4

Aufruf AO_T `= range (`AMR_{ST}`)`

Definition $= \{\hat{t}|(\hat{s},\hat{t},c) \in AMR_{ST}\}$

Beschreibung Alle aggregierten Range-Objekte eines AMR werden ermittelt.

Beispiel `$Result := range($AggPubAuth)`
Für das Eingabe-Mapping `$AggPubAuth` werden alle aggregierten Autoren ermittelt.

$AggPubAuth		
Publication	Author	Conf
$[p_1]$	$[a_1, a'_1]$	0.9
$[p'_1]$	$[a_1, a'_1]$	0.7
$[p_3, p'_3]$	$[a_2, a'_2]$	1
$[p'_3]$	$[a_3]$	1

⇨

$Result
Author
$[a_1, a'_1]$
$[a_2, a'_2]$
$[a_3]$

A.8.3 inverse

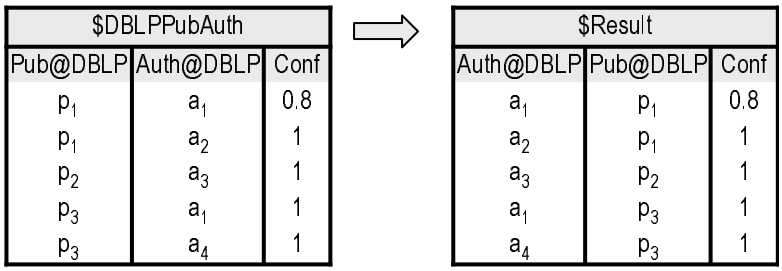
Der Operator `inverse` bildet die Umkehrung eines Mappings, indem in jeder Korrespondenz Domain- und Range-Objekt vertauscht werden.

Aufruf MR_{BA} `= inverse (`MR_{AB}`)`

Definition $= \{(b,a,c)|(a,b,c) \in MR_{AB}\}$

Beschreibung Es wird ein MR invertiert, d. h. Domain- und Range-Objekte werden in jedem Tupel vertauscht.

Beispiel `$Result := inverse ($DBLPPubAuth)`
Aus dem Eingabe-Mapping `$DBLPPubAuth` wird ein Mapping mit Korrespondenzen zwischen DBLP-Autoren und DBLP-Publikationen erstellt.



$DBLPPubAuth		
Pub@DBLP	Auth@DBLP	Conf
p_1	a_1	0.8
p_1	a_2	1
p_2	a_3	1
p_3	a_1	1
p_3	a_4	1

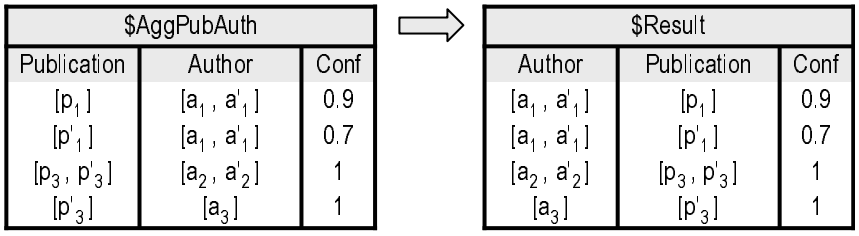
$Result		
Auth@DBLP	Pub@DBLP	Conf
a_1	p_1	0.8
a_2	p_1	1
a_3	p_2	1
a_1	p_3	1
a_4	p_3	1

Aufruf AMR_{TS} `= inverse (`AMR_{ST}`)`

Definition $= \{(\hat{t}, \hat{s}, c) | (\hat{s}, \hat{t}, c) \in AMR_{ST}\}$

Beschreibung Durch Vertauschen der aggregierten Domain- und Range-Objekte wird ein AMR invertiert.

Beispiel `$Result := inverse($AggPubAuth)`
Ein AMR mit Korrespondenzen zwischen Autoren und Publikationen wird aus dem Eingabe-Mapping `$AggPubAuth` erstellt.



$AggPubAuth		
Publication	Author	Conf
$[p_1]$	$[a_1, a'_1]$	0.9
$[p'_1]$	$[a_1, a'_1]$	0.7
$[p_3, p'_3]$	$[a_2, a'_2]$	1
$[p'_3]$	$[a_3]$	1

$Result		
Author	Publication	Conf
$[a_1, a'_1]$	$[p_1]$	0.9
$[a_1, a'_1]$	$[p'_1]$	0.7
$[a_2, a'_2]$	$[p_3, p'_3]$	1
$[a_3]$	$[p'_3]$	1

A.8.4 exec

Mit dem Operator `exec` werden iFuice-Prozeduren ausgeführt.

Aufruf X `= exec (Prozedurname,` $Param_1$`,` $Param_2$`, ...`$Param_n$`)`

Beschreibung Es wird die iFuice-Prozedur mit dem angegebenen Namen ausgeführt. Die weiteren Parameter $Param_i$ richten sich nach den Parametern der Prozedur und werden in der gegebenen Reihenfolge an

die Prozedur übergeben. Das Ergebnis des `exec`-Operators ist das Ergebnis der Prozedur und entspricht dem bei `return` übergebenen Wert.

Beispiel `$Result := exec(MergeMatch, $PubDBLPACM1, $PubDBLPACM2)`

Anmerkung Führt die als iFuice-Prozedur definierte Match-Strategie MergeMatch aus (siehe Abschnitt 9.4.1).

A.9 Skalare Funktionen

Die Operatoren `count` und `print` liefern Ergebnisse vom Typ Integer oder String zurück. Der `count`-Operator zählt die Anzahl der Objekte bzw. Korrespondenzen. Eine String-Serialisierung der Datenstrukturen liefert der `print`-Operator.

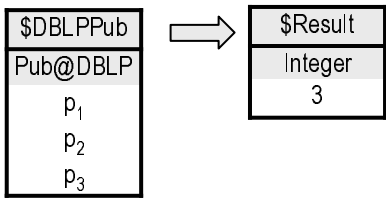
A.9.1 count

Der Operator `count` ermittelt die Größe einer Datenstruktur, d. h. die Anzahl der Objekte bzw. Korrespondenzen. Diese Informationen können z. B. zur Evaluation von Same-Mappings innerhalb von Object-Matching-Workflows verwendet werden.

Aufruf I `= count(`O`)`

Beschreibung Es wird die Anzahl der Objektinstanzen in O bestimmt.

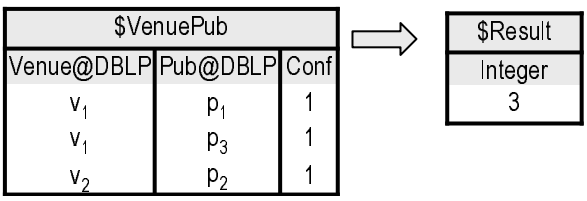
Beispiel `$Result := count($DBLPPub);`
Die Anzahl der in der Variable `$DBLPPub` gespeicherten DBLP-Publikationen wird bestimmt.



Aufruf I `= count(`MR`)`

Beschreibung Es wird die Anzahl der Korrespondenzen in MR bestimmt.

Beispiel `$Result := count($DBLPVenuePub);`
Die Anzahl der Korrespondenzen zwischen DBLP-Venues und DBLP-Publikationen wird bestimmt



Aufruf I = `count(`AO`)`
I = `count(`AMR`)`

Beschreibung Es wird die Anzahl der aggregierten Objekte (AO) bzw. Korrespondenzen (AMR) bestimmt.

A.9.2 print

Mit Hilfe von `print` werden Variablen jeden Typs in eine String-Repräsentation konvertiert. Dies dient u.a. zur Ausgabe von Informationen während einer Skriptausführung, z.B. indem die Größe einer Variable (vorher mittels `count` bestimmt) ausgegeben wird.

Aufruf S = `print(`O`)`
S = `print(`MR`)`
S = `print(`AO`)`
S = `print(`AMR`)`
S = `print(`S`)`
S = `print(`I`)`
S = `print(`F`)`

Beschreibung Eine String-Serialisierung des übergebenen Werts wird erzeugt und zurückgeliefert.

Literaturverzeichnis

[1] ACM Digital Library. http://portal.acm.org.

[2] DBLP Informatik Bibliography. http://dblp.uni-trier.de.

[3] Google Scholar. http://scholar.google.com.

[4] Adomavicius, G., Tuzhilin, A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering 17*, 6 (2005).

[5] Ambite, J. L., Ashish, N., Barish, G., Knoblock, C. A., Minton, S., Modi, P. J., Muslea, I., Philpot, A., Tejada, S. ARIADNE: A System for Constructing Mediators for Internet Sources. In *Proc. of the International Conference on Management of Data (SIGMOD)* (1998).

[6] Amin, M., Mabe, M. Impact Factors: Use and Abuse. *Perspectives in Publishing 1*, 1 (2000).

[7] Ananthakrishna, R., Chaudhuri, S., Ganti, V. Eliminating Fuzzy Duplicates in Data Warehouses. In *Proc. of 28th International Conference on Very Large Data Bases (VLDB)* (2002).

[8] Arenas, M., Kantere, V., Kementsietsidis, A., Kiringa, I., Miller, R. J., Mylopoulos, J. The Hyperion Project: From Data Integration to Data Coordination. *SIGMOD Record 32*, 3 (2003).

[9] Aumueller, D., Do, H. H., Massmann, S., Rahm, E. Schema and Ontology Matching with COMA++. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2005).

[10] Batini, C., Scannapieco, M. *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.

[11] Baxter, R., Christen, P., Churches, T. A Comparison of fast Blocking Methods for Record Linkage. In *Proc. of the SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Identification* (2003).

[12] Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C. Minerva: Collaborative p2p search. In *Proc. of the 31st International Conference on Very Large Data Bases (VLDB)* (2005).

[13] Berners-Lee, T., Hendler, J., Lassila, O. The semantic Web. *Scientific American 284*, 5 (2001).

[14] Bernstein, P. A. Generic Model Management: A Database Infrastructure for Schema Manipulation. In *Proc. of the 9th International Conference on Cooperative Information Systems (CoopIS)* (2001).

[15] Bernstein, P. A., DeWitt, D. J., Heuer, A., Ives, Z. G., Jensen, C. S., Meyer, H., Özsu, M. T., Snodgrass, R. T., Whang, K.-Y., Widom, J. Database Publication Practices. In *Proc. of the 31st International Conference on Very Large Data Bases (VLDB)* (2005).

[16] Bernstein, P. A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu, I. Data Management for Peer-to-Peer Computing : A Vision. In *Proc. of the 5th International Workshop on the Web and Databases (WebDB)* (2002).

[17] Bhattacharya, I., Getoor, L. Iterative Record Linkage for Cleaning and Integration. In *Proc. of the 9th Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)* (2004).

[18] Bilenko, M., Mooney, R. J. Adaptive Duplicate Detection using learnable String Similarity Measures. In *Proc. of the 9th International Conference on Knowledge Discovery and Data Mining (SIGKDD)* (2003).

[19] Bilenko, M., Mooney, R. J., Cohen, W. W., Ravikumar, P., Fienberg, S. E. Adaptive Name Matching in Information Integration. *IEEE Intelligent Systems 18*, 5 (2003).

[20] Bilke, A., Bleiholder, J., Böhm, C., Draba, K., Naumann, F., Weis, M. Automatic Data Fusion with HumMer. In *Proc. of the 31st International Conference on Very Large Data Bases (VLDB)* (2005).

[21] Bitton, D., DeWitt, D. J. Duplicate Record Elimination in Large Data Files. *ACM Transactions on Database Systems 8*, 2 (1983).

[22] Bleiholder, J., Naumann, F. Declarative data fusion - syntax, semantics, and implementation. In *Proc. of the 9th East European Conference on Advances in Databases and Information Systems (ADBIS)* (2005).

[23] Brin, S., Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks 30*, 1-7 (1998).

[24] Broder, A. A Taxonomy of Web Search. *SIGIR Forum 36*, 2 (2002).

[25] Burke, R. D. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction 12*, 4 (2002).

[26] Burke, R. D., Mobasher, B., Williams, C., Bhaumik, R. Classification Features for Attack Detection in Collaborative Recommender Systems. In *Proc. of the 12th International Conference on Knowledge Discovery and Data*

Mining (KDD) (2006).

[27] Cai, Y., Dong, X. L., Halevy, A. Y., Liu, J. M., Madhavan, J. Personal Information Management with SEMEX. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2005).

[28] Carey, M. J., Haas, L. M., Schwarz, P. M., Arya, M., Cody, W. F., Fagin, R., Flickner, M., Luniewski, A., Niblack, W., Petkovic, D., II, J. T., Williams, J. H., Wimmers, E. L. Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. In *Proc. of the 5th International Workshop on Research Issues in Data Engineering - Distributed Object Management (RIDE-DOM)* (1995).

[29] Carvalho, J. C. P., da Silva, A. S. Finding Similar Identities among Objects from Multiple Web Sources. In *Proc. of the 5th International Workshop on Web Information and Data Management (WIDM)* (2003).

[30] Chaudhuri, S., Dayal, U. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record 26*, 1 (1997).

[31] Chaudhuri, S., Ganjam, K., Ganti, V., Kapoor, R., Narasayya, V. R., Vassilakis, T. Data Cleaning in Microsoft SQL Server 2005. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2005).

[32] Chaudhuri, S., Ganti, V., Motwani, R. Robust Identification of Fuzzy Duplicates. In *Proc. of the 21st International Conference on Data Engineering (ICDE)* (2005).

[33] Chawathe, S. S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. D., Widom, J. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *Proc. of the 10th Meeting of the Information Processing Society of Japan (IPSJ)* (1994).

[34] Chen, Z., Kalashnikov, D. V., Mehrotra, S. Exploiting Relationships for Object Consolidation. In *Proc. of the International Workshop on Information Quality in Information Systems (IQIS)* (2005).

[35] Chi, E. H., Rosien, A., Heer, J. LumberJack: Intelligent Discovery and Analysis of Web User Traffic Composition. In *Proc. of the 4th International WebKDD Workshop* (2002).

[36] Chirkova, R., Halevy, A. Y., Suciu, D. A Formal Perspective on the View Selection Problem. In *Proc. of 27th International Conference on Very Large Data Bases (VLDB)* (2001).

[37] Chirkova, R., Halevy, A. Y., Suciu, D. A Formal Perspective on the View Selection Problem. *The VLDB Journal 11*, 3 (2002).

[38] Christen, P., Churches, T., Hegland, M. Febrl - A Parallel Open Source Data Linkage System. In *Proc. of the 8th Pacific-Asia Conference on Advances*

in Knowledge Discovery and Data Mining (PAKDD) (2004).

[39] Cohen, W. W., Ravikumar, P., Fienberg, S. E. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proc. of IJCAI Workshop on Information Integration on the Web (IIWeb)* (2003).

[40] Cooley, R., Mobasher, B., Srivastava, J. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems 1*, 1 (1999).

[41] Cosley, D., Lawrence, S., Pennock, D. M. REFEREE: An Open Framework for Practical Testing of Recommender Systems using ResearchIndex. In *Proc. of 28th International Conference on Very Large Data Bases (VLDB)* (2002).

[42] Critchlow, T., Fidelis, K., Ganesh, M., Musick, R., Slezak, T. DataFoundry: Information Management for Scientific Data. *IEEE Transactions on Information Technology in Biomedicine 4*, 1 (2000).

[43] Culotta, A., McCallum, A. Joint Deduplication of Multiple Record Types in Relational Data. In *Proc. of the International Conference on Information and Knowledge Management (CIKM)* (2005).

[44] Decker, S., Melnik, S., Harmelen, F. V., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., Horrocks, I. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing 15*, 3 (2000).

[45] Dey, D., Sarkar, S., De, P. A Distance-Based Approach to Entity Reconciliation in Heterogeneous Databases. *IEEE Transactions Knowledge and Data Engineering 14*, 3 (2002).

[46] Do, H. H., Kirsten, T., Rahm, E. Comparative Evaluation of Microarray-based Gene Expression Databases. In *Proc. of the 10th Conference on Database Systems for Business, Technology, and Web (BTW)* (2003).

[47] Do, H. H., Rahm, E. COMA - A System for Flexible Combination of Schema Matching Approaches. In *Proc. of 28th International Conference on Very Large Data Bases (VLDB)* (2002).

[48] Do, H. H., Rahm, E. Flexible Integration of Molecular-Biological Annotation Data: The GenMapper Approach. In *Proc. of the 9th International Conference on Extending Database Technology (EDBT)* (2004).

[49] Doan, A., Lu, Y., Lee, Y., Han, J. Object Matching for Information Integration: A Profiler-Based Approach. In *Proc. of the Workshop on Information Integration on the Web (IIWeb)* (2003).

[50] Domenig, R., Dittrich, K. R. An Overview and Classification of Mediated Query Systems. *SIGMOD Record 28*, 3 (1999).

[51] Dong, X., Halevy, A. Y. A Platform for Personal Information Management

and Integration. In *Proc. of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR)* (2005).

[52] Dong, X., Halevy, A. Y., Madhavan, J. Reference Reconciliation in Complex Information Spaces. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2005).

[53] Dumais, S., Chen, H. Hierarchical Classification of Web content. In *Proc. of the 23rd International Conference on Research and Development in Information Retrieval (SIGIR)* (2000).

[54] Edgar F. Codd, S. B. Codd, C. T. S. Providing OLAP to User-Analysts: An IT Mandate. Tech. rep., Codd Associates, Ann Arbor/Michigan, 1993.

[55] Egghe, L. Theory and Practise of the g-Index. *Scientometrics 69*, 1 (2006).

[56] Elfeky, M. G., Elmagarmid, A. K., Verykios, V. S. TAILOR: A Record Linkage Tool Box. In *Proc. of the 18th International Conference on Data Engineering (ICDE)* (2002).

[57] Elmagarmid, A. K., Ipeirotis, P. G., Verykios, V. S. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering 19*, 1 (2007).

[58] Etzold, T., Ulyanov, A., Argos, P. SRS: information retrieval system for molecular biology data banks. *Methods Enzymology 266* (1996).

[59] Felligi, I. P., Sunter, A. B. A Theory of Record Linkage. *Journal of the American Statistical Association 64* (1969).

[60] Franklin, M. J., Halevy, A. Y., Maier, D. From Databases to Dataspaces: A new Abstraction for Information Management. *SIGMOD Record 34*, 4 (2005).

[61] Friedman, M., Levy, A. Y., Millstein, T. D. Navigational Plans For Data Integration. In *Proc. of the 16th National Conference on Artificial Intelligence and 11th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)* (1999).

[62] Fujibuchi, W., Goto, S., Migimatsu, H., Uchiyama, I., Ogiwara, A., Akiyama, Y., Kanehisa, M. DBGET/LinkDB: An Integrated Database Retrieval System. In *Proc. of Pacific Symposium on Biocomputing (PSB)* (1998).

[63] Galhardas, H., Florescu, D., Shasha, D., Simon, E. AJAX: An Extensible Data Cleaning Tool. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2000).

[64] Galhardas, H., Florescu, D., Shasha, D., Simon, E., Saita, C.-A. Declarative Data Cleaning: Language, Model, and Algorithms. In *Proc. of 27th International Conference on Very Large Data Bases (VDLB)* (2001).

[65] Galperin, M. Y. The Molecular Biology Database Collection: 2006 update. *Nucleic Acids Res 34* (2006).

[66] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J. D., Vassalos, V., Widom, J. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal Intelligent Information Systems 8*, 2 (1997).

[67] Geyer-Schulz, A., Hahsler, M. Evaluation of Recommender Algorithms for an Internet Information Broker based on Simple Association Rules and on the Repeat-Buying Theory. In *Proc. of the 4th WebKDD Workshop on Web Mining for Usage Patterns & User Profiles* (2002).

[68] Goble, C. A., Stevens, R., Ng, G., Bechhofer, S., Paton, N. W., Baker, P. G., Peim, M., Brass, A. Transparent Access to multiple Bioinformatics Information Sources. *IBM Systems Journal 40*, 2 (2001).

[69] Goldberg, D., Nichols, D., Oki, B. M., Terry, D. B. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM 35*, 12 (1992).

[70] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[71] Golovin, N., Rahm, E. Reinforcement Learning Architecture for Web Recommendations. In *Proc. of the International Conference on Information Technology: Coding and Computing (ITCC)* (2004).

[72] Golovin, N., Rahm, E. Automatic Optimization of Web Recommendations Using Feedback and Ontology Graphs. In *Proc. of the 5th International Conference on Web Engineering (ICWE)* (2005).

[73] Gravano, L., Ipeirotis, P. G., Jagadish, H. V., Koudas, N., Muthukrishnan, S., Srivastava, D. Approximate String Joins in a Database (Almost) for Free. In *Proc. of 27th International Conference on Very Large Data Bases (VLDB)* (2001).

[74] Gray, J., Bosworth, A., Layman, A., Pirahesh, H. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In *Proc. of the 12th International Conference on Data Engineering (ICDE)* (1996).

[75] Gu, L., Baxter, R., Vickers, D., Rainsford, C. Record Linkage: Current Practice and Future Directions. Tech. rep., CSIRO Mathematical and Information Sciences, 2003.

[76] Gupta, P., Pu, P. Social Cues and Awareness for Recommendation Systems. In *Proc. of the 8th International Conference on Intelligent User Interfaces (IUI)* (2003).

[77] Haas, L. M., Schwarz, P. M., Kodali, P., Kotlar, E., Rice, J. E.,

Swope, W. C. DiscoveryLink: A system for Integrated Access to Life Sciences Data Sources. *IBM Systems Journal 40*, 2 (2001).

[78] Halevy, A. Y. Answering Queries using Views: A survey. *The VLDB Journal 10*, 4 (2001).

[79] Halevy, A. Y. Data Integration: A Status Report. In *Proc. of the 10th Conference on Database Systems for Business, Technology, and Web (BTW)* (2003).

[80] Halevy, A. Y. Why your Data won't mix. *ACM Queue 3*, 8 (2005).

[81] Halevy, A. Y., Etzioni, O., Doan, A., Ives, Z. G., Madhavan, J., McDowell, L., Tatarinov, I. Crossing the Structure Chasm. In *Proc. of the 1st Biennial Conference on Innovative Data Systems Research (CIDR)* (2003).

[82] Halevy, A. Y., Franklin, M. J., Maier, D. Principles of Dataspace Systems. In *Proc. of the 25th Symposium on Principles of Database Systems (PODS)* (2006).

[83] Halevy, A. Y., Ives, Z. G., Madhavan, J., Mork, P., Suciu, D., Tatarinov, I. The Piazza Peer Data Management System. *IEEE Transaction on Knowledge Data Engineering 16*, 7 (2004).

[84] Halevy, A. Y., Ives, Z. G., Mork, P., Tatarinov, I. Piazza: Data Management Infrastructure for Semantic Web Applications. In *Proc. of the 12th International World Wide Web Conference (WWW)* (2003).

[85] Halevy, A. Y., Rajaraman, A., Ordille, J. J. Data Integration: The Teenage Years. In *Proc. of the 32nd International Conference on Very Large Data Bases (VLDB)* (2006).

[86] Hayes, C., Massa, P., Avesani, P., Cunningham, P. An On-Line Evaluation Framework for Recommender Systems. In *Proc. of the Workshop on Personalization and Recommendation in E-Commerce* (2002).

[87] Herlocker, J. L., Konstan, J. A., Terveen, L. G., Riedl, J. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems 22*, 1 (2004).

[88] Hernández, M. A., Stolfo, S. J. The Merge/Purge Problem for Large Databases. In *Proc. of the International Conference on Management of Data (SIGMOD)* (1995).

[89] Hernández, M. A., Stolfo, S. J. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Mining and Knowledge Discovery 2*, 1 (1998).

[90] Hernandez, T., Kambhampati, S. Integration of Biological Sources: Current Systems and Challenges Ahead. *SIGMOD Record 33*, 3 (2004).

[91] Hirsch, J. E. An Index to quantify an Individual's Scientific Research Output. *Proc. of the National Academy of Sciences 102*, 46 (2005).

[92] Inmon, W. H. *Building the Data Warehouse, 3rd Edition.* John Wiley & Sons, Inc., New York, NY, USA, 2002.

[93] Ishikawa, H., Ohta, M., Yokoyama, S., Nakayama, J., Katayama, K. On the Effectiveness of Web Usage Mining for Page Recommendation and Restructuring. In *Proc. of the Workshop on Web, Web-Services, and Database Systems* (2002).

[94] Ives, Z. G., Khandelwal, N., Kapur, A., Cakir, M. ORCHESTRA: Rapid, Collaborative Sharing of Dynamic Data. In *Proc. of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR)* (2005).

[95] Jameson, A., Konstan, J. A., Riedl, J. AI Techniques for Personalized Recommendation. *Tutorial at 18th National Conference on Artificial Intelligence (AAAI)* (2002).

[96] Joachims, T., Freitag, D., Mitchell, T. M. Web Watcher: A Tour Guide for the World Wide Web. In *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI)* (1997).

[97] Jonas, J. Identity Resolution: 23 Years of Practical Experience and Observations at scale. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2006).

[98] Kalashnikov, D. V., Mehrotra, S., Chen, Z. Exploiting Relationships for Domain-Independent Data Cleaning. In *Proc. of the 5th SIAM International Conference on Data Mining (SDM)* (2005).

[99] Kammenhuber, N., Luxenburger, J., Feldmann, A., Weikum, G. Web search clickstreams. In *Proc. of the 6th Conference on Internet Measurement (IMC)* (2006).

[100] Kementsietsidis, A., Arenas, M., Miller, R. J. Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2003).

[101] Kim, K., Carroll, J. M., Rosson, M. B. An Empirical Study of Web Personalization Assistants: Supporting End-Users in Web Information Systems. In *Proc. of the International Symposium on Human-Centric Computing Languages and Environments (HCC)* (2002).

[102] Kirsten, T., Do, H. H., Körner, C., Rahm, E. Hybrid Integration of Molecular-Biological Annotation Data. In *Proc. of the 2nd International Workshop on Data Integration in the Life Sciences (DILS)* (2005).

[103] Kirsten, T., Thor, A., Rahm, E. Instance-Based Matching of Large Life Science Ontologies. In *Proc. of the 4th International Workshop on Data Integration in the Life Sciences (DILS)* (2007).

[104] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., Riedl, J. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM 40*, 3 (1997).

[105] Köpcke, H., Rahm, E. Analyse von Zitierungshäufigkeiten für die Datenbankkonferenz BTW. *Datenbank-Spektrum 20* (2007).

[106] Körner, C., Kirsten, T., Do, H. H., Rahm, E. Hybride Integration von molekularbiologischen Annotationsdaten. In *Proc. of the 11th Conference on Database Systems for Business, Technology, and Web (BTW)* (2005).

[107] Kosala, R., Blockeel, H. Web Mining Research: A Survey. *SIGKDD Explorations 2*, 1 (2000).

[108] Koutri, M., Daskalaki, S., Avouris, N. Adaptive Interaction with Web Sites: an Overview of Methods and Techniques. In *Proc. of the 4th International Workshop on Computer Science and Information Technologies (CSIT)* (2002).

[109] Kushmerick, N., McKee, J., Toolan, F. Towards Zero-Input Personalization: Referrer-Based Page Prediction. In *Proc. of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH)* (2000).

[110] Lee, M.-L., Hsu, W., Kothari, V. Cleaning the Spurious Links in Data. *IEEE Intelligent Systems 19*, 2 (2004).

[111] Lenzerini, M. Data Integration: A Theoretical Perspective. In *Proc. of the 21st Symposium on Principles of Database Systems (PODS)* (2002).

[112] Leser, U., Naumann, F. (Almost) Hands-Off Information Integration for the Life Sciences. In *Proc. of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR)* (2005).

[113] Leser, U., Naumann, F. *Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen.* dpunkt, 2007.

[114] Levenshtein, V. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission 1*, 1 (1965).

[115] Levy, A. Y., Rajaraman, A., Ordille, J. J. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proc. of 22th International Conference on Very Large Data Bases (VLDB)* (1996).

[116] Lieberman, H. Letizia: An Agent That Assists Web Browsing. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI)* (1995).

[117] Lim, E.-P., Srivastava, J., Prabhakar, S., Richardson, J. Entity Identification in Database Integration. In *Proc. of the 9th International Conference on Data Engineering (ICDE)* (1993).

[118] Lim, M., Kim, J. An Adaptive Recommendation System with a Coordina-

tor Agent. In *Proc. of the 1st Asia-Pacific Conference on Web Intelligence: Research and Development* (2001).

[119] Linden, G., Smith, B., York, J. Industry Report: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Distributed Systems Online 4*, 1 (2003).

[120] Litwin, W., Mark, L., Roussopoulos, N. Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys 22*, 3 (1990).

[121] Liu, X., Hui, Y., Sun, W., Liang, H. Towards Service Composition Based on Mashup. In *Proc. of IEEE Conference on Services* (2007).

[122] Low, W. L., Lee, M.-L., Ling, T. W. A Knowledge-based Approach for Duplicate Elimination in Data Cleaning. *Information Systems 26*, 8 (2001).

[123] McCallum, A., Nigam, K., Ungar, L. H. Efficient Clustering of high-dimensional Data Sets with Application to Reference Matching. In *Proc. of the 6th International Conference on Knowledge Discovery and Data Mining (SIGKDD)* (2000).

[124] Melnik, S., Rahm, E., Bernstein, P. A. Rondo: A Programming Platform for Generic Model Management. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2003).

[125] Michalowski, M., Thakkar, S., Knoblock, C. A. Exploiting secondary Sources for unsupervised Record Linkage. In *Proc. of the SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation* (2003).

[126] Michalowski, M., Thakkar, S., Knoblock, C. A. Automatically Utilizing Secondary Sources to Align Information Across Sources. *AI Magazine 26*, 1 (2005).

[127] Michelson, M., Knoblock, C. A. Learning Blocking Schemes for Record Linkage. In *Proc. of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI)* (2006).

[128] Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., Riedl, J. MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System. In *Proc. of the International Conference on Intelligent User Interfaces* (2003).

[129] Minton, S., Nanjo, C., Knoblock, C. A., Michalowski, M., Michelson, M. A Heterogeneous Field Matching Method for Record Linkage. In *Proc. of the 5th IEEE International Conference on Data Mining (ICDM)* (2005).

[130] Mobasher, B., Cooley, R., Srivastava, J. Automatic Personalization based on Web Usage Mining. *Communications of the ACM 43*, 8 (2000).

[131] Mobasher, B., Dai, H., Luo, T., Nakagawa, M. Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. *Data Mining and Knowledge Discovery 6*, 1 (2002).

[132] Monge, A. E., Elkan, C. The Field Matching Problem: Algorithms and Applications. In *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining (SIGKDD)* (1996).

[133] Monge, A. E., Elkan, C. P. An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. In *Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD)* (1997).

[134] Nakagawa, M., Mobasher, B. A Hybrid Web Personalization Model Based on Site Connectivity. *Proc. of the International WebKDD Workshop* (2003).

[135] Naumann, F. Folien zur Vorlesung Informationsintegration. http://www.informatik.hu-berlin.de/forschung/gebiete/wbi/ii/folien.

[136] Navarro, G. A guided Tour to approximate String Matching. *ACM Computing Surveys 33*, 1 (2001).

[137] Neiling, M., Jurk, S. The Object Identification Framework. In *Proc. of the SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation* (2003).

[138] Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., Risch, T. EDUTELLA: A P2P Networking Infrastructure based on RDF. In *Proc. of the 11th International World Wide Web Conference (WWW)* (2002).

[139] Newcombe, H. Record Linking: The Design of Efficient Systems for Linking Records into Individual and Family Histories. *American Journal of Human Genetics 19*, 3 (1967).

[140] Newcombe, H. B., Kennedy, J. M., Axford, S. J., James, A. P. Automatic Linkage of vital Records. *Science 130*, 3381 (1959).

[141] Ng, W. S., Ooi, B. C., Tan, K.-L., Zhou, A. PeerDB: A P2P-based System for Distributed Data Sharing. In *Proc. of the 19th International Conference on Data Engineering (ICDE)* (2003).

[142] Oliveira, P., Rodrigues, F., Henriques, P., Galhardas, H. A Taxonomy of Data Quality Problems. In *Proc. of the International Workshop on Data and Information Quality* (2005).

[143] Omar Benjelloun, Hector Garcia-Molina, H. G. H. K. T. E. L. D. M. S. T. D-Swoosh: A Family of Algorithms for Generic, Distributed Entity Resolution. In *Proc. of the 27th International Conference on Distributed Computing Systems (ICDCS)* (2007).

[144] Ooi, B. C., Tan, K.-L., Zhou, A., Goh, C. H., Li, Y., Liau, C. Y., Ling, B., Ng, W. S., Shu, Y., Wang, X., Zhang, M. PeerDB: Peering into Personal Databases. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2003).

[145] Pant, G., Srinivasan, P., Menczer, F. Crawling the Web. In *Web Dynamics - Adapting to Change in Content, Size, Topology and Use*. Springer, 2004.

[146] Papakonstantinou, Y., Abiteboul, S., Garcia-Molina, H. Object Fusion in Mediator Systems. In *Proc. of the 22th International Conference on Very Large Data Bases (VLDB)* (1996).

[147] Pasula, H., Marthi, B., Milch, B., Russell, S. J., Shpitser, I. Identity Uncertainty and Citation Matching. In *Proc. of Advances in Neural Information Processing Systems 15* (2002).

[148] Perkowitz, M., Etzioni, O. Adaptive Web Sites: Automatically Synthesizing Web Pages. In *Proc. of the 15th National Conference on Artificial Intelligence and the 10th Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI)* (1998).

[149] Perkowitz, M., Etzioni, O. Adaptive Web Sites. *Communications of the ACM 43*, 8 (2000).

[150] Perugini, S., Gonçalves, M. A., Fox, E. A. Recommender Systems Research: A Connection-Centric Survey. *Journal of Intelligent Information Systems 23*, 2 (2004).

[151] Pierrakos, D., Paliouras, G., Papatheodorou, C., Spyropoulos, C. D. Web Usage Mining as a Tool for Personalization: A Survey. *User Modeling and User-Adapted Interaction 13*, 4 (2003).

[152] Podnar, I., Luu, T., Rajman, M., Klemm, F., Aberer, K. A peer-to-peer architecture for information retrieval across digital library collections. In *Proc. of the 10th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)* (2006).

[153] Quass, D., Starkey, P. Record Linkage for Genealogical Databases. In *Proc. of the SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation* (2003).

[154] Rahm, E., Bernstein, P. A. A Survey of Approaches to automatic Schema Matching. *The VLDB Journal 10*, 4 (2001).

[155] Rahm, E., Bernstein, P. A. An online Bibliography on Schema Evolution. *SIGMOD Record 35*, 4 (2006).

[156] Rahm, E., Do, H. H. Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin 23*, 4 (2000).

[157] Rahm, E., Thor, A. Citation Analysis of Database Publications. *SIGMOD Record 34*, 4 (2005).

[158] Rahm, E., Thor, A., Aumueller, D., Do, H. H., Golovin, N., Kirsten, T. IFuice - Information Fusion utilizing Instance Correspondences and Peer Mappings. In *Proc. of the 8th International Workshop on the Web & Databases (WebDB)* (2005).

[159] Raman, V., Hellerstein, J. M. Potter's Wheel: An Interactive Data Cleaning System. In *Proc. of 27th International Conference on Very Large Data Bases (VLDB)* (2001).

[160] Resnick, P., Varian, H. R. Recommender Systems - Introduction to the Special Section. *Communications of the ACM 40*, 3 (1997).

[161] Reuther, P., Walter, B., Ley, M., Weber, A., Klink, S. Managing the Quality of Person Names in DBLP. In *Proc. of the 10th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)* (2006).

[162] Ritter, O., Kocab, P., Senger, M., Wolf, D., Suhai, S. Prototype Implementation of the Integrated Genomic Database. *Computers and Biomedical Research 27*, 2 (1994).

[163] Rodríguez-Gianolli, P., Garzetti, M., Jiang, L., Kementsietsidis, A., Kiringa, I., Masud, M., Miller, R. J., Mylopoulos, J. Data Sharing in the Hyperion Peer Database System. In *Proc. of the 31st International Conference on Very Large Data Bases (VLDB)* (2005).

[164] Roth, A., Naumann, F., Hübner, T., Schweigert, M. System P: Query Answering in PDMS under Limited Resources. In *Proc. of the 5th International Workshop on Information Integration on the Web (IIWeb)* (2006).

[165] Rother, K., Müller, H., Trissl, S., Koch, I., Steinke, T., Preissner, R., Frömmel, C., Leser, U. Columba: Multidimensional Data Integration of Protein Annotations. In *Proc. of the 1st International Workshop on Data Integration in the Life Sciences (DILS)* (2004).

[166] Salton, G., Wong, A., Yang, C. S. A Vector Space Model for Automatic Indexing. *Communications of the ACM 18*, 11 (1975).

[167] Sarawagi, S., Bhamidipaty, A. Interactive Deduplication using Active Learning. In *Proc. of the 8th International Conference on Knowledge Discovery and Data Mining (KDD)* (2002).

[168] Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. Analysis of Recommendation Algorithms for E-Commerce. In *Proc. of the ACM Conference on Electronic Commerce* (2000).

[169] Schafer, J. B., Konstan, J. A., Riedl, J. E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery 5*, 1/2 (2001).

[170] Schafer, J. B., Konstan, J. A., Riedl, J. Meta-Recommendation Systems: User-controlled Integration of Diverse recommendations. In *Proc. of the International Conference on Information and Knowledge Management (CIKM)* (2002).

[171] Schrödl, H. *Business Intelligence mit Microsoft SQL Server 2005*. Hanser Verlag, 2006.

[172] Shahabi, C., Chen, Y.-S. An Adaptive Recommendation System without Explicit Acquisition of User Relevance Feedback. *Distributed and Parallel Databases 14*, 2 (2003).

[173] Shaker, R., Mork, P., Brockenbrough, J., Donelson, L., Tarczy-Hornoch, P. The BioMediator System as a Tool for Integrating Biologic Databases on the Web. In *Proc. of the Workshop on Information Integration on the Web (IIWeb)* (2004).

[174] Shardanand, U., Maes, P. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proc. of the Conference on Human Factors in Computing Systems (CHI)* (1995).

[175] Shen, W., Li, X., Doan, A. Constraint-Based Entity Matching. In *Proc. of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI)* (2005).

[176] Sheth, A. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. *Interoperating Geographic Information Systems* (1999).

[177] Singla, P., Domingos, P. Multi-Relational Record Linkage. In *Proc. of the 3rd Workshop on Multi-Relational Data Mining (MRDM)* (2004).

[178] Sinha, R., Swearingen, K. Comparing Recommendations Made by Online Systems and Friends. In *Proc. of the Workshop on Personalization and Recommender Systems in Digital Libraries* (2001).

[179] Sizov, S., Theobald, M., Siersdorfer, S., Weikum, G., Graupmann, J., Biwer, M., Zimmer, P. The BINGO! System for Information Portal Generation and Expert Web Search. In *Proc. of the 1st Biennial Conference on Innovative Data Systems Research (CIDR)* (2003).

[180] Slaton, G., McGill, M. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc. New York, NY, USA, 1986.

[181] Spiliopoulou, M., Mobasher, B., Berendt, B., Nakagawa, M. A Framework for the Evaluation of Session Reconstruction Heuristics in Web-Usage Analysis. *INFORMS Journal on Computing 15*, 2 (2003).

[182] Srivastava, J., Cooley, R., Deshpande, M., Tan, P.-N. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations 1*, 2 (2000).

[183] Staab, S., Studer, R., Eds. *Handbook on Ontologies.* International Handbooks on Information Systems. Springer, 2004.

[184] T. Stöhr, E. Rahm, Q. S. OLAP-Auswertung von Web-Zugriffen. In *Proc. of GI-Workshop Internet-Datenbanken* (2000).

[185] Tan, P., Kumar, V. Discovery of Web Robot Sessions based on their Navigational Patterns. *Data Mining and Knowledge Discovery 6* (2002).

[186] Tejada, S., Knoblock, C. A., Minton, S. Learning Object Identification Rules for Information Integration. *Information Systems 26*, 8 (2001).

[187] Tejada, S., Knoblock, C. A., Minton, S. Learning Domain-independent String Transformation Weights for High Accuracy Object Identification. In *Proc. of the 8th International Conference on Knowledge Discovery and Data Mining (KDD)* (2002).

[188] Thor, A., Aumueller, D., Rahm, E. Data Integration Support for Mashups. In *Proc. of the 6th International Workshop on Information Integration on the Web (IIWeb)* (2007).

[189] Thor, A., Golovin, N., Rahm, E. Adaptive Website Recommendations with AWESOME. *The VLDB Journal 14*, 4 (2005).

[190] Thor, A., Kirsten, T., Rahm, E. Instance-based Matching of Hierarchical Ontologies. In *Proc. of the 12th Conference on Database Systems for Business, Technology, and Web (BTW)* (2007).

[191] Thor, A., Rahm, E. AWESOME - A Data Warehouse-based System for Adaptive Website Recommendations. In *Proc. of the 30th International Conference on Very Large Data Bases (VLDB)* (2004).

[192] Thor, A., Rahm, E. MOMA - A Mapping-based Object Matching System. In *Proc. of the 3rd Biennial Conference on Innovative Data Systems Research (CIDR)* (2007).

[193] Tomasic, A., Amouroux, R., Bonnet, P., Kapitskaia, O., Naacke, H., Raschid, L. The distributed Information Search Component (Disco) and the World Wide Web. In *Proc. of the International Conference on Management of Data (SIGMOD)* (1997).

[194] Tran, T. Designing Recommender Systems for E-Commerce: An Integration Approach. In *Proc. of the 8th International Conference on Electronic Commerce (ICEC)* (2006).

[195] Ullman, J. D. Information Integration Using Logical Views. In *Proc. of the 6th International Conference on Database Theory (ICDT)* (1997).

[196] Venter, J. C., et al. The Sequence of the Human Genome. *Science 291*, 5507 (2001).

[197] Vollmer, S. Portierung des DBLP-Systems auf ein relationales Daten-

banksystem und Evaluation der Performance. Diplomarbeit, Univeristät Trier, 2006.

[198] Weis, M., Naumann, F. DogmatiX Tracks down Duplicates in XML. In *Proc. of the International Conference on Management of Data (SIGMOD)* (2005).

[199] Weis, M., Naumann, F. Detecting duplicates in complex xml data. In *Proc. of the 22nd International Conference on Data Engineering (ICDE)* (2006).

[200] Wexelblat, A., Maes, P. Footprints: History-Rich Tools for Information Foraging. In *Proc. of the Conference on Human Factors in Computing Systems (CHI)* (1999).

[201] Wiederhold, G. Mediators in the Architecture of Future Information Systems. *IEEE Computer 25*, 3 (1992).

[202] William Phillips, J., Bahn, A. K., Miyasaki, M. Person-matching by electronic Methods. *Communications of the ACM 5*, 7 (1962).

[203] Winkler, W. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. In *Proc. of the Section on Survey Research Methods, American Statistical Association* (1990).

[204] Winkler, W. The State of Record Linkage and Current Research Problems. Tech. rep., US Bureau of the Census, 1999.

[205] Witten, I., Frank, E. *Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 2000.

[206] Wong, L. Kleisli, its Exchange Format, Supporting Tools, and an Application in Protein Interaction Extraction. In *Proc. of 1st IEEE International Symposium on Bioinformatics and Biomedical Engineering (BIBE)* (2000).

[207] Yu, K., Schwaighofer, A., Tresp, V., Xu, X., Kriegel, H.-P. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering 16*, 1 (2004).

[208] Ziegler, P. Data Integration Projects World-Wide. http://www.ifi.unizh.ch/~pziegler/IntegrationProjects.html.

[209] Ziegler, P., Dittrich, K. R. Three Decades of Data Integration - All Problems solved? In *Proc. of the 18th IFIP World Computer Congress Congress* (2004).

Wissenschaftlicher Werdegang

Persönliche Angaben: Andreas Thor
geboren am 22. Februar 1979 in Bad Muskau

Schulbildung und Bundeswehr:

09/1985 – 08/1987	7. POS „Hermann Matern“, Weißwasser
09/1987 – 07/1991	6. POS „Hans Beimler“, Weißwasser
08/1991 – 06/1998	6. Gymnasium „Max Steenbeck“, Cottbus
07/1998 – 04/1999	Grundwehrdienst im 1./425 PzArtBtl Lehnitz

Ausbildung und Beruf:

04/1999 – 12/2002	Studium der Informatik an der Universität Leipzig Diplomarbeit: Wissensrepräsentation im Sport
01/2003 – 12/2005	Promotionsstudent im Graduiertenkolleg „Wissensrepräsentation“, Universität Leipzig
seit 01/2006	Wissenschaftlicher Mitarbeiter an der Abteilung Datenbanken, Universiät Leipzig

Bibliografische Angaben

Thor, Andreas: Automatische Mapping-Verarbeitung auf Webdaten. Dissertation, Universität Leipzig, 2008

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 9. November 2007 Andreas Thor