Hochschule
Bonn-Rhein-Sieg
University of Applied Science

Fachbereich Informatik
Department of Computer Science

Fraunhofer Institut Angewandte Informationstechnik

Master - Thesis

im Studiengang
Master of Science in Computer Science

Objektrelationale Datenbanken und Rough Sets für die Analyse von Contextualized Attention Metadata

von

Frank Beer <frank.beer@smail.inf.h-brs.de>

Erstbetreuer: Dr. Martin E. Müller <martin.mueller@h-brs.de>
Zweitbetreuer: Prof. Dr. Peter Becker <peter.becker@h-brs.de>

Eingereicht am 07. Dezember 2009

Eidesstattliche Erklärung

An Eides statt versichere ich, die von mir vorgelegte Arbeit

Objektrelationale Datenbanken und Rough Sets
für die Analyse von
Contextualized Attention Metadata

selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Ort / Datum Unterschrift

Danksagung

Bevor diese Arbeit beginnt, möchte ich einigen Menschen, die mich fachlich, wie auch persönlich bei der Erstellung dieser Arbeit unterstützt haben, explizit meinen herzlichen Dank aussprechen. Wären sie nicht gewesen, hätte diese Arbeit sicherlich nicht mit der - aus meiner Sicht - hohen Qualität vorgelegt werden können.

Zunächst gilt mein fachlicher Dank meinen zwei Betreuern, Dr. Martin E. Müller und Prof. Dr. Peter Becker an der Hochschule. Herrn Müller möchte ich speziell für die Motivation in die reizvolle Thematik der *Rough Set Theory* danken. Die netten und fachlichen Gespräche vor und während dieser Arbeit haben mir stets gute Denkanstöße gegeben, die ich in dieser Arbeit wertbringend verarbeiten durfte. Herrn Becker möchte ich für seine organisatorische Flexibilität danken, so dass wichtige Themen, gerade in Bezug auf die Basisalgorithmik aus dem Datenbankbereich, während der Arbeit schnell und reibungslos diskutiert werden konnten.

Am Fraunhofer-Institut FIT danke ich Herrn Dr. Martin Wolpers. Er hat mir das interessante Thema dieser Arbeit vorgeschlagen und gerade am Ende der Arbeit fundierte wissenschaftliche Ratschläge gegeben. Einen besonderen Dank richte ich an Frau Maren Scheffel. Sie hat mich mit viel Geduld organisatorisch wie auch fachlich über den gesamten Zeitraum der Anfertigung dieser Arbeit konstruktiv unterstützt. Nicht zuletzt ihr Engagement in den letzten Tagen habe ich viel zu verdanken. Insgesamt möchte ich mich für den großzügigen Freiraum und die unkomplizierte Zusammenarbeit bedanken.

Nicht zu vergessen sind die Personen, die mir privat am nächsten stehen. Meiner Familie und meinen Freunden gebührt ein herzlicher Dank dafür, dass sie mir auch in stressigen Zeiten zur Seite standen. Ganz besonders möchte ich dabei meine Eltern hervorheben. Sie haben mir während der Arbeit, aber auch während des gesamten Studiums sehr viele Lasten von den Schultern genommen, so dass ich zielstrebig und ohne jede Sorge meine Ziele verfolgen konnte. Mein uneingeschränkter Dank gilt somit ihnen. Einen liebevollen Dank möchte ich meiner Freundin Eva aussprechen. Sie hat mich über weite Teile dieser Arbeit beherbergt und mir durch beruhigende und motivierende Gespräche den Rücken gestärkt.

Abstract

Das *Contextualized Attention Metadata Schema* ist ein auf XML basierendes Schema für das Speichern von Interaktionsinformationen zwischen Mensch und Computer. Ausgehend von vorliegenden Daten in diesem Format aus dem Forschungsprojekt *Context and Attention in Personalized Learning Environments* am Fraunhofer-Institut für Angewandte Informationstechnik FIT soll im Rahmen dieser Arbeit durch die Verwendung geeigneter Methoden eine Datenanalyse durchgeführt werden. Die Analysemethoden sollen sich an den Ideen und Konzepten der *Rough Set Theory* orientieren, um letztlich das Interesse von Benutzern respektive von Benutzergruppen am Computer zu extrahieren.

In den frühen 80er Jahren des letzten Jahrhunderts wurden *Rough Sets* als mathematische Hilfsmittel für die Approximation von Zielkonzepten entwickelt. Diese Approximationen basieren auf Äquivalenzrelationen, die es ermöglichen, „ähnliche“ Datenobjekte eines gegebenen Datenraums zu gruppieren. Damit sind *Rough Sets* eine Erweiterung der klassischen Mengentheorie. Seit den 1970er Jahren haben sich relationale Datenbanken in praktischen Anwendungsfällen als zuverlässige Systeme etabliert, die mengenorientierte Strukturen und effiziente Basisalgorithmen für das Speichern und Abfragen von Daten bereitstellen. Ein solches Datenbanksystem ist PostgreSQL.

Am Beispiel von PostgreSQL, sollen im Zuge dieser Arbeit, die genannten Datenbank-Features für die Analyse von *Contextualized Attention Metadata* mittels *Rough Sets* ausgenutzt werden. Diese Aufgabe impliziert zum einen die Transformation des *Contextualized Attention Metadata Schemas* in ein relationales Datenbankschema; zum anderen müssen bereits bestehende Verfahren aus dem Bereich des Data-Mining in Verbindung mit *Rough Sets* gefunden und ggf. adaptiert werden. Ziel dieser Arbeit ist es also, ein Datenbanksystem um sinnvolle Werkzeuge der *Rough Set Theory* und des Data-Mining zu erweitern, so dass ein parametrisierbares objektrelationales Gesamtsystem entsteht, mit dem Auswertungen aus der Mensch-Computer-Interaktion durchgeführt werden können.

Inhaltsverzeichnis

1 Einleitung

Das Angebot von personalisierten Umgebungen im Internet ist ein aktueller Trend in der Informationsgesellschaft. In großen Online-Kaufhäusern und -Marktplätzen wird diese Entwicklung beispielsweise aufgegriffen, um den Kunden eine Qualitätsverbesserung der Online-Dienstleistung zu offerieren. Die Basis für die Konstruktion solcher Umgebungen ist die Analyse des Benutzerverhaltens. So können Kunden aufgrund ihres bisherigen Profils genau die Produkte angeboten werden, die auch wirklich zu ihnen passen. Dennoch treten in diesem Zusammenhang häufig Fehleinschätzungen durch die Empfehlungsalgorithmen auf, weil sie mitunter keine globale Sicht auf den einzelnen Kunden besitzen, sondern ihre Analysen jeweils nur auf Erfahrungen und eine Momentaufnahme des Nutzerverhaltens stützen.

In einem anderen kontextuellen Zusammenhang wird im Rahmen des Forschungsprojekts *Context and Attention in Personalized Learning Environments* am Fraunhofer-Institut für Angewandte Informationstechnik FIT versucht, die genannte Schwäche zu überwinden, indem eine umfassende Perspektive bei der Analyse von Benutzerverhalten am Computer zugrunde gelegt wird. Das Projekt stellt sich dabei der Herausforderung individuelle Umgebungen bei einem computer-gestützten Lernen bereit zu stellen, um die stetig steigende Störgröße der Informationsüberflutung für Lerner am Computer bestmöglich zu minimieren. Das *Contextualized Attention Metadata Schema* als XML basierte Struktur hat sich hierbei als adäquate Datenorganisation für die ganzheitliche Analyse von Benutzern mit digitalen Inhalten bewehrt.

Ziel dieser Arbeit soll es sein, unter Zuhilfenahme der *Rough Set Theory* Methoden für die Musterextraktion von Benutzeraktivitäten zu entwickeln, um das Forschungsprojekt am Fraunhofer-Institut FIT in seinem Vorhaben zu unterstützen. Die grundlegende Idee ist es, eine sinnvolle Abbildung zwischen den Basiskonzepten der *Rough Set Theory* und den etablierten relationalen Anfragewerkzeugen aus der traditionellen Datenbankwelt zu finden. Zu diesem Zweck muss allerdings in einem ersten Schritt das *Contextualized Attention Metadata Schema* in ein nützliches relationales Datenbankmodell transformiert werden, damit die Benutzerdaten für die zu entwerfenden Algorithmen auch in einer nativen Form zur Analyse bereit

stehen. PostgreSQL als objektrelationales Datenbanksystem steht für die Persistierung dieser Daten zur Verfügung. Darüber hinaus stellt es die mengentheoretischen Basisalgorithmen bereit und dient ebenfalls als Erprobungssystem für die Praxistauglichkeit im Forschungsprojekt.

Insgesamt wird sich aus den Zielen dieser Arbeit versprochen, selbst auf großen Datenbeständen schnell Auswertungen durchführen zu können. In wie weit jedoch die Qualität der zu entwickelnden Methoden auf den Benutzerdaten ausreicht, bleibt zu evaluieren.

2 Einführung in die Rough Set Theory

Die *Rough Set Theory* (RST) ist ein mathematisches Framework für die Analyse von Daten, das zu Beginn der 1980er Jahre von Zdzislaw Pawlak entwickelt worden ist. Als zentraler Gegenstand dieser Arbeit werden in diesem Kapitel die grundlegenden Terminologien und Sichtweisen dieser Theorie eingeführt und diskutiert. Dabei werden ausgehend von der Strukturierung der zu analysierenden Daten und den Konzepten der Äquivalenzrelation sowohl die formalen Werkzeuge für die Approximation von Zielkonzepten als auch die Eliminierung redundanter Informationen im vorhandenen Datenbestand definiert.

2.1 Informationssystem

Informationen können auf verschiedene Arten betrachtet bzw. strukturiert werden. Eine gängige Form der Darstellung sind Tabellen. Im einfachsten Fall besteht eine solche Tabelle aus einer oder mehreren Spalten und Zeilen. Jede Spalte charakterisiert dabei die Informationen aus der Tabelle. In Anlehnung an diese Eigenschaften von Spalten finden im Folgenden auch die Begriffe *Merkmale* oder *Attribute* Anwendung. Auf der anderen Seite bilden die Zeilen eine abgeschlossene Einheit an vorhandener Information. In der mathematischen Betrachtung solcher tabellarischer Strukturen werden die einzelnen Zeilen auch häufig als Objekte bezeichnet [KPPS99]. Diese Bezeichnung ist allerdings problematisch, da das Wort *Objekt* in ähnlichen Thematiken[1] kontextuell anderen Bedeutungen unterliegt. Um dieser terminologischen Überstrapazierung entgegen zu wirken, sei im Folgenden von einem Objekt die Rede, sofern die vorliegenden Informationen einer Zeile ausreichen, um eine Sache für eine bestimmte Anwendungsdomäne vollständig zu beschreiben. Zur Verdeutlichung dieses Problems sei zunächst Abbildung 2.1 zu betrachten. Die dort

[1] Zum Beispiel kann in der Objektorientierung von einem Objekt gesprochen werden, wenn eine Sache ein gewisses Verhalten aufweist und einen definierten Zustand (Informationen) hält [JCJÖ92]. Angemerkt sei jedoch, dass bis dato keine einheitliche Begriffsdefinition vorliegt [SST97].

dargestellte Tabelle enthält für ein Szenario ausreichend viele Daten über geometrische Körper mit den Eigenschaften *Form*, *Größe* und *Farbe*.

-	*Form*	*Größe*	*Farbe*
○	Kreis	klein	blau
□	Quadrat	mittel	grün
◯	Kreis	groß	rot
△	Dreieck	mittel	schwarz
□	Quadrat	mittel	rot
△	Dreieck	klein	blau

Abbildung 2.1: Tabellarische Strukturierung von geometrischen Objekte

Eine eher formale Betrachtungsweise solch einer zweidimensionalen Struktur wurde ursprünglich von Pawlak unter dem Begriff *Informationssystem* (IS) geprägt [Paw81]. Es dient als Ausgangspunkt für alle im Weiteren betrachteten Aspekte der RST.

Definition 2.1. Das Tupel $\mathcal{A} = (\mathbb{U}, A)$ heißt IS. Dabei sei das Universum $\mathbb{U} = \{x_1, ..., x_n\}, n \in \mathbb{N}$ eine Menge von Objekten, die durch die Menge von Attributen $A = \{a_1, ..., a_m\}, m \in \mathbb{N}$ beschrieben wird. Für alle $a \in A$ gilt die Abbildungsvorschrift $a : \mathbb{U} \rightarrow V_a$, welche einem Objekt genau einen Wert aus dem Wertebereich V_a zuordnet.

Aus der der vorangegangenen Definition 2.1 geht nicht im Detail hervor, wie die Abbildungsvorschriften $a \in A$ mathematisch definiert sind. Es wird aber deutlich, dass alle $a \in A$ die Eigenschaft einer Abbildung haben, begründet durch ihre Rechtseindeutigkeit [Bri05]. Sie sind jedoch nicht linkseindeutig, weil von einander verschiedene Objekte bezüglich eines Attributs sehr wohl den gleichen Wert annehmen können [Hac05]. Zusätzlich kann für die einzelnen Abbildungen $a \in A$ gefordert werden, dass sie links- sowie rechtstotal sind, sofern das Vorkommen von Nullwerten ($\perp$) [MU03] in den V_a vernachlässigt wird. Gilt dennoch in einem gegebenen IS für mindestens ein Attribut $a \in A : a(x) = \perp, x \in \mathbb{U}$, so wird das IS auch *unvollständig* genannt [YLS07].

Beispiel 2.1. Formale Betrachtung eines IS
Gegeben sei das vollständige IS $\mathcal{A} = (\mathbb{U}, A)$ aus Abbildung 2.1 mit $\mathbb{U} = \{$○, ◯, □, □, △, △$\}$ und $A = \{Form, Größe, Farbe\}$. Für die Wertebereiche von $\mathcal{A}$ gilt dann: $V_{Form} = \{Kreis, Quadrat, Dreieck\}$, $V_{Größe} = \{klein, mittel, groß\}$, $V_{Farbe} = \{schwarz, blau, rot, grün\}$ und beispielsweise für die Attribute *Farbe* und *Form*: $Farbe(\circ) = blau$, $Farbe(\square) = rot$, $Form(\square) = Quadrat$, $Form(\triangle) = Dreieck$.

Eine Erweiterung eines IS ist das sogenannte *Entscheidungssystem* (engl. *Decision System* oder kurz DS). Im Gegensatz zum zuvor erläuterten IS besitzt ein DS ein oder mehrere zusätzliche Attribute $D = \{d_1, ..., d_p\}, p \in \mathbb{N}$, deren Werte eine anwendungsfallspezifische Aussage bezüglich der Objekte tätigen. Typischerweise finden Entscheidungssysteme im beaufsichtigten Lernen (engl. *Supervised Learning*), ein Teilgebiet des maschinellen Lernens (ML), Anwendung. Hierbei findet eine Vorklassifizierung der Objekte durch einen Lehrer oder einen Experten statt, welche auf seinem Wissen und seinen Erfahrungen beruht [Mit97, PTL00]. Ein solches System hat im Allgemeinen die Gestalt $\mathcal{A}_d = (\mathbb{U}, A \cup D)$, mit dem Universum $\mathbb{U}$, der Attributmenge A und den Entscheidungsattributen $D = \{d_1, ..., d_p\}$ $p \in \mathbb{N}$, wobei die einzelnen $d \in D$ Abbildungen der Art $d : \mathbb{U} \rightarrow V_d$ darstellen. Im einfachsten Fall jedoch hat $\mathcal{A}_d$ ein einziges Entscheidungsattribut $d \notin A$, welches typischerweise Objekte in bool'sche Werte abbildet: $d(x) \in \mathbb{B}, \forall x \in \mathbb{U}$ [KPPS99].

2.2 Ununterscheidbarkeit

Neben dem IS spielt die Ununterscheidbarkeit (engl. *Indiscernibility*) von Objekten eine zentrale Rolle in der RST. Aber auch andere naturwissenschaftliche Bereiche greifen auf dieses Charakteristika von Objekten zurück. Können Objekte nicht voneinander unterschieden werden, so eröffnet dieser Sachverhalt die Möglichkeit der Gruppierung gleichwertiger Objekte. Sie kann dazu genutzt werden, eine große Menge von Datenobjekten ohne Informationsverlust auf seine Grundstruktur zu reduzieren [Beu01]. Der Hintergrund dieses Gedankengangs lässt sich dadurch erläutern, dass die vorliegenden Gruppen gleicher Elemente wiederum durch einen Stellvertreter der Gruppe repräsentiert werden können, was letztlich zur Reduzierung des Ausgangsdatenbestands auf seine wesentliche Struktur führt. Zusätzlich lässt sich argumentieren, dass bei vermeintlich gleichen Objekten redundante Information vorliegt, was insbesondere im praktischen Umgang mit organisierten Daten zu Inkonsistenzproblemen führen kann [KE04]. Aus dem Blickpunkt des ML kann die Fähigkeit Objekte von einander zu differenzieren dazu genutzt werden, Wissen aufzubauen. Es ist nach Pawlak möglich, die Problematik der Klassifikation von Objekten auf die Unterscheidbarkeit dieser abzubilden [Paw91]. Mathematisch lässt sich die Ununterscheidbarkeit von Objekten allgemein mit Hilfe der Äquivalenzrelation definieren.

Definition 2.2. Sei $R \subseteq U \times U$ eine binäre Relation, die reflexiv ($\forall x \in U : (x, x) \in R$), symmetrisch ($\forall x, y \in U : (x, y) \in R \Leftrightarrow (y, x) \in R$) und

transitiv ($\forall x, y, z \in U : (x, y), (y, z) \in R \Rightarrow (x, z) \in R$) ist, dann wird R auch Äquivalenzrelationen genannt.

Die Äquivalenz zweier Objekte $x, y \in U$ bezüglich einer Äquivalenzrelation[2] $\sim$ wird mit $x \sim y$ oder mit $x \equiv_\sim y$ notiert, wobei alle äquivalenten Objekte zu einem bestimmten Objekt $x \in U$ in eine *Äquivalenzklasse* $[x]_\sim = \{y \in U \mid x \sim y\}$ fallen. x gilt dann als ein *Repräsentant* dieser Klasse. Wird nicht nur eine, sondern die Menge aller Äquivalenzklassen betrachtet, die sich aufgrund von $\sim$ und U einstellt, so entsteht eine *Partition* auf U, beschrieben durch $U/\sim = \{K_1, ..., K_n\}$, mit K_i als Äquivalenzklasse, $1 \leq i \leq n \in \mathbb{N}$. Der Grund für die Partitionierung lässt sich anhand der Eigenschaften der Äquivalenzrelation zeigen, denn es gilt: $U = \bigcup_{i=1}^{n \in \mathbb{N}} K_i$. Das bedeutet, dass zwei Äquivalenzklassen entweder gleich oder disjunkt von einander sind [Bri05, Hac05].

Ein wesentlicher Nachteil bei der Verwendung von klassischen Äquivalenzrelationen ist, dass Elemente einer Menge aufgrund ihrer Gesamtheit an Merkmalen als gleich oder ungleich charakterisiert werden. Dies wird deutlich, wenn man beispielsweise das Konzept der Äquivalenzrelationen auf das IS in Abbildung 2.1 abstrahiert. Keines der geometrischen Objekte ist mit einem anderen hinsichtlich aller Attribute äquivalent. Wird allerdings nur eine Teilmenge der Attribute betrachtet, so lassen sich bei manchen Objekten gewisse Ähnlichkeiten beobachten. Zum Beispiel sind die Objekte □, □ zwar hinsichtlich ihrer *Farbe* verschieden ($Farbe(\square) \neq Farbe(\square)$), aber bezüglich ihrer Merkmale *Form* und *Größe* identisch ($Form(\square) = Form(\square)$ und $Größe(\square) = Größe(\square)$). Wird nun das Konzept der Äquivalenzrelation auf den oben erläuterten Sachverhalt erweitert, so dass sich eine Äquivalenzrelation auf Basis der Merkmale von Objekten ergibt, erhält man eine parametrisierbare *Ununterscheidbarkeitsrelation*.

Definition 2.3. Sei $\mathcal{A} = (\mathbb{U}, A)$ mit $B \subseteq A$, dann lässt sich die Ununterscheidbarkeitsrelation $IND_{\mathcal{A}}(B)$ wie folgt formalisieren:

$$IND_{\mathcal{A}}(B) = \{(x, y) \in \mathbb{U}^2 \mid a(x) = a(y), \forall a \in B\} \tag{2.1}$$

Es lässt sich nachvollziehen, dass $IND_{\mathcal{A}}(B)$ Definition 2.2 erfüllt und somit eine Äquivalenzrelation darstellt. Für verschiedenste Anwendungsfälle kann $IND_{\mathcal{A}}(B)$ durch den Parameter B, einer Teilmenge der Gesamtheit an vorhandenen Attributen A in $\mathcal{A}$, angepasst werden. Dadurch ist es möglich, die Gruppierung äquivalenter

[2] Wenn es sich um eine Äquivalenzrelation handelt, findet anstelle von R häufig das Symbol $\sim$ Anwendung [Bri05].

Objekte nur auf einem Teilbereich ihrer Eigenschaften durchzuführen. Im Bezug auf die geometrischen Körper □ und □ aus Abbildung 2.1 kann somit nun gesagt werden, dass beide Objekte durch $IND_{\mathcal{A}}(\{Form,\ Größe\})$ auf dem Universum $\mathbb{U}$ äquivalent, also ununterscheidbar, sind.

Es sei im Folgenden auf die etablierten und verkürzten Schreibweisen bezüglich der Ununterscheidbarkeitsrelation hingewiesen: Da es sich häufig um eine bekannte Domäne handelt, in der die „neue“ Äquivalenzrelation Verwendung findet, wird der Indikator $\mathcal{A}$ für ein bestimmtes Informations-/ Entscheidungssystem in der Bezeichnung der Relation ignoriert [KPPS99]. Ebenso wird der Parameter B der Kompaktheit halber in den Index der Relation verlagert, so dass insgesamt IND_B anstelle von $IND_{\mathcal{A}}(B)$ notiert wird. Für die Partition $\mathbb{U}/IND_B = \{K_1, ..., K_n\}, n \in \mathbb{N}$, wird im Folgenden auch IND_B geschrieben [PTL00]. Der semantische Unterschied zwischen der Ununterscheidbarkeitsrelation und der Partition, die sie induziert, ergibt sich dann aus dem jeweiligen Kontext. Des Weiteren handelt es sich bei einer Partition $IND_B = \{K_1, ..., K_n\}, n \in \mathbb{N}$, um eine Menge von disjunkten Teilmengen, deren Vereinigung wieder die Basismenge ergibt [Hac05]. Allgemein gilt $K_i \subseteq IND_B$. Um die Zugehörigkeit einer solchen Teilmenge K_i zu einer Partition IND_B zu unterstreichen, wird nachkommend $K_i \in IND_B$ verwendet[3], was mengentheoretisch dazu führt, dass $IND_B = \mathbb{U}$ ist. Der *Index* $n \in \mathbb{N}$ von IND_B sei durch die Kardinalität $|IND_B| = n$ beschrieben. Häufig ist ein Repräsentant $x \in \mathbb{U}$ einer Äquivalenzklasse K_i bekannt. Als Kurzform der bekannten Schreibweise $[x]_{IND_B}$ wird auch $[x]_B$ geschrieben [KPPS99].

Die bisherigen Aussagen bezüglich der Ununterscheidbarkeitsrelation sind jeweils auf ein festes B bezogen. Es lässt sich jedoch ersehen, dass IND_B nicht eine Ununterscheidbarkeitsrelation beschreibt, sondern mehrere. Es können maximal $2^{|A|} - 1$ sinnvolle und unterschiedliche Relationen auf Basis von IND_B entstehen, wobei mit $|A|$ die Anzahl der Attribute eines zugrunde liegenden IS gemeint ist. Diese Aussage findet Begründung anhand der möglichen Kombinationen, die $B \in \mathcal{P}(A) - \emptyset$ annehmen kann. Es kann aber durchaus das Phänomen auftreten, dass die beiden vermeintlich unterschiedlichen Relationen IND_{B_1} und $IND_{B_2}, B_1, B_2 \subseteq A, B_1 \neq B_2$ die gleiche Partition auf $\mathbb{U}$ induzieren und somit $IND_{B_1} = IND_{B_2}$ ist.

Beispiel 2.2. Verschiedene Granularitäten der Ununterscheidbarkeit
Gegeben sei $\mathcal{A} = (\mathbb{U}, A)$ mit dem Universum $\mathbb{U} = \{x_1, ..., x_5\}$ und der Attributmenge $A = \{f, g, h\}$ aus Abbildung 2.2. Betrachtet man nun die Objekte des Universums, die sich bezüglich A von einander unterscheiden, so ergeben sich $|IND_A| =$

[3] Bei dieser Betrachtung sei der strikte Formalismus wie u.a. in [Mod93, ZY04] vernachlässigt.

-	f	g	h
x_1	12	qu	2
x_2	30	we	5
x_3	19	qu	2
x_4	12	qu	2
x_5	19	we	5

Abbildung 2.2: Ein abstraktes Informationssystem

$|IND_{\{f,g,h\}}| = 4$ von einander disjunkte Äquivalenzklassen mit $IND_{\{f,g,h\}} = \{\{x_1, x_4\}, \{x_2\}, \{x_3\}, \{x_5\}\}$. Für die Attribute $\{g, h\} \subset A$ ergibt sich bezüglich der Ununterscheidbarkeitsrelation eine Partition bestehend aus $|IND_{\{g,h\}}| = 2$ Teilmengen mit $IND_{\{g,h\}} = \{\{x_1, x_3, x_4\}, \{x_2, x_5\}\}$. Es ist ersichtlich, dass anhand der Wahl der Attribute $B \subseteq A$ für IND_B sich durchaus unterschiedliche Granularitäten auf $\mathbb{U}$ einstellen. Durch die Gesamtheit an Attributen A können die Objekte in $\mathbb{U}$ viel feiner voneinander unterschieden werden als beispielsweise aus der Kombination $\{g, h\}$, da $|IND_{\{g,h\}}| < |IND_{\{f,g,h\}}|$ ist. Des Weiteren lässt sich aufgrund des gegebenen IS $\mathcal{A}$ zeigen, dass mehrere verschiedene Ununterscheidbarkeitsrelationen die gleichen Partitionen auf $\mathbb{U}$ induzieren. Dazu sei betrachtet, dass $IND_{\{g,h\}} = IND_{\{g\}} = IND_{\{h\}}$ und $IND_{\{f,g,h\}} = IND_{\{f,g\}} = IND_{\{f,h\}} = IND_{\{f\}}$ gilt. Zusammengefasst gibt es letztendlich also nur zwei Ununterscheidbarkeitsrelationen in $\mathcal{A}$, die induzierbar sind.

2.3 Approximation von Konzepten

Eine klassische Disziplin im ML ist das Lösen des *Konzeptlernproblems*[4] in einer gegebene Anwendungsdomäne. Hierbei soll eine Maschine in der Lage sein, auf Basis einer gegebenen Datenmenge ein gewisses Zielkonzept (engl. *Target Concept*) zu lernen. Präziser formuliert geht es um die Approximation einer meist unbekannten Zielfunktion $c : U \rightarrow \mathbb{B}$ durch eine geeignete Hypothese h, so dass $h(x) = c(x)$ für möglichst alle Datensätze x im Datenbestand U [Mit97]. Um den Begriff des zu lernenden Konzepts zu veranschaulichen seien im Folgenden drei Beispielszenarien für mögliche Zielkonzepte genannt:

- Sind detaillierte Krankheitsverläufe von Patienten mit Symptomen der tatsächlich eingetretenen Krankheit bekannt, so wäre es von praktischer Relevanz,

[4] Im Speziellen wird hier das Zwei-Klassen-Lernproblem erläutert.

herauszufinden, anhand welcher Symptome beispielsweise das Krankheitsbild *Grippe* diagnostiziert werden kann [PGBSZ95].

- In einem Finanzinstitut haben sich über die Jahre Erfahrungen bzw. Einschätzungen über Kunden angesammelt, die die monatlichen Raten für ihr Darlehn entweder pünktlich, verspätet oder gar nicht bezahlt haben. Es ist für die Bank von Bedeutung, aufgrund der Erfahrung zu bewerten, wann ein potenzieller neuer Kunde kreditwürdig ist und wann nicht [Alp08].
- Der Spamfilter eines Emailprogramms soll anhand einer gegebenen Beispielmenge von Emails lernen, wann eingehende Emails als Spam markiert werden sollen und wann nicht [Tre04].

Aus den genannten Beispielen wird deutlich, dass sich für viele Aufgabenstellungen ein Konzept aus einer Teilmenge der zu analysierenden Daten ergibt bzw. beschreiben lässt [Mit97]. Liegen diese Daten in einem DS $\mathcal{A}_d = (\mathbb{U}, A \cup \{d\})$ vor, so lässt sich typischerweise ein Konzept $X \subseteq \mathbb{U}$ durch $X = \{x \in \mathbb{U} \mid d(x) = 1\}$ mit $d : \mathbb{U} \to \mathbb{B}$ formulieren.

Um herauszufinden, welche Gruppen ähnlicher Objekte im Zusammenhang mit dem Zielkonzept stehen, sind zunächst die Objekte $x \in \mathbb{U}$ eines gegebenen IS $\mathcal{A} = (\mathbb{U}, A)$ bezüglich ihrer Ununterscheidbarkeit IND_B mit $B \subseteq A$ (s. Kapitel 2.2) und eines Konzepts $X \subseteq \mathbb{U}$ zu untersuchen: Es ist ersichtlich, dass abhängig davon wie X definiert ist, einige Äquivalenzklassen der Partition $IND_B = \bigcup [x_i]_B, x_i \in \mathbb{U}, 0 < i \leq |\mathbb{U}|$ vollständig, andere nur teilweise oder gar nicht in X fallen. Ist eine Klasse $[x_j]_B = X_{in} \cup X_{out} \in IND_B$ und $X_{in} \subseteq X$ aber $X_{out} \not\subseteq X$, so kann keinerlei Aussage darüber getroffen werden, ob Objekte $x \in [x_j]_B$ immer im Konzept enthalten sind. Für andere $[x_k]_B \subseteq X, x_k \not\equiv_{IND_B} x_j$ kann dies sehr wohl getan werden. Es können also Szenarien in einem IS bzw. DS auftreten, in denen unter Zuhilfenahme der Informationen in $B \subseteq A$ das Konzept X mengentheoretisch nicht exakt beschrieben werden kann. Einen formalen Lösungsansatz für Problemstellungen dieser Art liefert die *Rough Set Theory* (dt. Grobmengen-Theorie) [Paw82, MP84, Paw91].

Aus Sicht der Ununterscheidbarkeitsrelation IND_B besteht der Ansatz der RST darin, dass unexakte Konzept X durch zwei exakte Mengen zu approximieren. Diese zwei Mengen heißen *B-Lower* und *B-Upper Approximation* (dt. B-untere und B-obere Approximation) und werden typischerweise mit $\underline{X}_B$ und $\overline{X}_B$ abgekürzt [FBRS96]. Formal lassen sich $\underline{X}_B$ und $\overline{X}_B$ wie folgt beschreiben:

Definition 2.4. Sei $\mathcal{A} = (\mathbb{U}, A)$ mit dem Zielkonzept $X \subseteq \mathbb{U}$ und der Attributmenge $B \subseteq A$. X lässt sich durch die B-untere und B-obere Approximation

zusammenfassen:

$$\underline{X}_B = \{x \in \mathbb{U} \mid [x]_B \subseteq X\} \tag{2.2}$$

$$\overline{X}_B = \{x \in \mathbb{U} \mid [x]_B \cap X \neq \emptyset\} \tag{2.3}$$

Definition 2.4 besagt, dass die B-untere Approximation ($\underline{X}_B$) gerade aus den Äquivalenzklassen besteht, die vollständig im Konzept enthalten sind, während die B-obere Approximation ($\overline{X}_B$) zusätzlich die Klassen enthält, welche nur teilweise durch das Konzept beschrieben werden. Mittels dieser beiden Mengen kann eine Teilmenge der vorliegenden Objekte approximiert werden.

Definition 2.5. Ein Konzept $X \subseteq \mathbb{U}$ in einem IS $\mathcal{A} = (\mathbb{U}, A), B \subseteq A$ kann durch die zwei Mengen $\underline{X}_B$ und $\overline{X}_B$ beschrieben werden. Diese Konzeptapproximation für X nennt sich *Rough Set*, beschrieben durch:

$$RoughSet_{\mathcal{A}}^{B}(X) = (\underline{X}_B, \overline{X}_B) \tag{2.4}$$

Es lässt sich überlegen, dass die Größe der zwei approximativen Mengen $\overline{X}_B$ und $\underline{X}_B$ abhängig von den Informationen in $B \subseteq A$ respektive $X \subseteq \mathbb{U}$ ist. Sind sie nicht identisch, so ergibt sich zusätzlich eine weitere Menge.

Definition 2.6. Sei $\mathcal{A} = (\mathbb{U}, A)$ mit $X \subseteq \mathbb{U}$ und $B \subseteq A$. Die *B-Boundary Region* (dt. B-Grenzregion) errechnet sich aus der mengentheoretischen Differenz zwischen B-oberer und B-unterer Approximation:

$$\underline{\overline{X}}_B = \overline{X}_B - \underline{X}_B \tag{2.5}$$

$\underline{\overline{X}}_B$ enthält also die Objekte solcher Äquivalenzklassen, welche sowohl innerhalb als auch außerhalb des Konzepts X liegen, während die Objekte, die keinen direkten Zusammenhang mit dem Konzept X haben, in die *B-Outside Region* (dt. B-Außenregion) fallen. Sie lässt sich durch $\mathbb{U} - \overline{X}_B$ angeben und ist für weitere Untersuchungen von X nicht weiter relevant.

Angemerkt sei an dieser Stelle, dass sich die Symbolik für $\overline{X}_B, \underline{X}_B$ und $\underline{\overline{X}}_B$ aus dem Zielkonzept $X \subseteq \mathbb{U}$ und der Attributmenge $B \subseteq A$ ergibt. Für eine andere Betrachtung eines Konzept $Y \neq X$ und einer Attributmenge $F \neq B$ sind die F-obere und F-untere Approximation und F-Grenzregion bezüglich IND_F wie folgt benannt: $\overline{Y}_F, \underline{Y}_F$ und $\underline{\overline{Y}}_F$ [KPPS99]. Im Folgenden wird auf einige wichtige Eigenschaften der Rough-Sets hingewiesen:

Satz. Auf einem IS $\mathcal{A} = (\mathbb{U}, A)$ mit $B \subseteq A$ gilt stets:

(i) $X \subseteq \mathbb{U} : \underline{X}_B \subseteq X \subseteq \overline{X}_B$

(ii) $X = \emptyset : \underline{X}_B = \overline{X}_B = \emptyset$

(iii) $X = \mathbb{U} : \underline{X}_B = \overline{X}_B = \mathbb{U}$

(iv) $X \subseteq Y \subseteq \mathbb{U} : \underline{X}_B \subseteq \underline{Y}_B, \overline{X}_B \subseteq \overline{Y}_B$

(v) $x \notin X \subseteq \mathbb{U}, x \in \underline{X}_B$: $\exists y \in X$, so dass $x \equiv_{IND_B} y$

Beweisidee.

zu (i) $\underline{X}_B$ ist die Vereinigung aller Teilmengen K_i der Partitionierung von IND_B, die in X vollständig enthalten sind. $\underline{X}_B$ kann also auch nur selbst eine Teilmenge von X sein. Auf der anderen Seite ist $\overline{X}_B$ die Vereinigung aller Teilmengen K_j, die X schneiden. $\overline{X}_B$ kann neben X somit auch andere Objekte außerhalb von X enthalten: $X \subseteq \overline{X}_B$.

zu (ii) Ist $X = \emptyset$, dann gibt es in X keine Objekte und somit auch keine Äquivalenzklassen. $\underline{X}_B$ muss leer sein. Es gibt also auch keine Klasse, die einen nicht-leeren Schnitt mit X hat. Folglich ist $\overline{X}_B$ auch leer.

zu (iii) $\underline{X}_B$ enthält die Klassen, die in X enthalten sind. Da $X = \mathbb{U}$ ist also die gesamte Partition, welche durch IND_B induziert wird in $\underline{X}_B$ und das ist genau $\mathbb{U}$. Weiter enthält $\overline{X}_B$ die Objekte der Klassen K_i für die gilt: $K_i \cap X \neq \emptyset$. Da $X = \mathbb{U}$ sind alle $K_i \subseteq \overline{X}_B$ und das ist $\mathbb{U}$. Es folgt: $\underline{X}_B = \overline{X}_B = \mathbb{U}$.

zu (iv) Es gilt: $X \subseteq Y$. In X sind möglicherweise weniger Klassen vollständig enthalten als in Y. Somit muss $\underline{X}_B \subseteq \underline{Y}_B$. Die gleiche Argumentation greift für die B-obere Approximation: $\overline{X}_B \subseteq \overline{Y}_B$.

zu (v) In $\underline{X}_B$ sind nur die Objekte der Klassen K_i enthalten, für die gilt: $K_i = X_{in} \cup X_{out}, (X_{in}, X_{out} \neq \emptyset)$, mit $X_{in} \subseteq X$ und $X_{out} \not\subseteq X$. Für die Objekte $x \in X_{out}$ muss es also mindestens einen äquivalenten Partner $y \in X_{in}$ geben, sonst wäre $K_i \not\subseteq \underline{X}_B$.

□

Beispiel 2.3. Schematische Visualisierung eines Rough-Sets
Es sei eine Verteilung von Objekten auf dem vollständigen Datenraum eines IS $\mathcal{A} = (\mathbb{U}, B)$ durch Abbildung 2.3 (a) gegeben. Betrachtet man diese Verteilung näher, so ist ersichtlich, dass $\frac{21}{61} \approx 34\%$ der zugrunde liegenden Daten durch das Zielkonzept X beschrieben werden, während die anderen 40 Objekte (ca. 66% der

Gesamtheit) sich außerhalb dieser Region befinden. Da diese Darstellungsart von $\mathcal{A}$ die Möglichkeit bietet, auch die Zwischenräume bzw. Werte zu veranschaulichen, die nicht von Objekten angenommen wurden, zeigt Abbildung 2.3 (b) alle möglichen Äquivalenzklassen, welche durch IND_B induziert werden könnten. Im Gegensatz dazu illustriert Abbildung 2.3 (c) die tatsächliche Partition, die sich aufgrund von $IND_B = \{K_1, K_2, ..., K_{29}\} = \bigcup_{i=1}^{n=29} K_i = \mathbb{U}$ im gegebenen Raum einstellt.

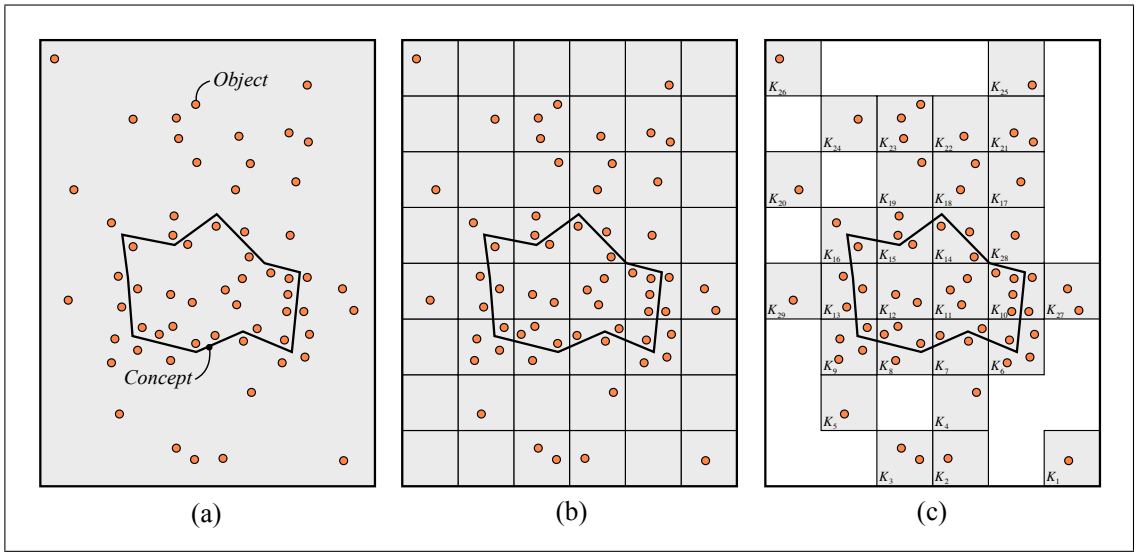


Abbildung 2.3: Schematische Darstellung eines Informationssystems

Abbildung 2.3 (c) zeigt weiter, dass durch die Ununterscheidbarkeitsrelation IND_B das gesamte Konzept X nicht exakt dargestellt werden kann. Es gibt genau zwei Klassen, K_{11} und K_{12}, die mit Gewissheit in das Zielkonzept fallen, neun weitere Äquivalenzklassen, K_6 bis K_{10} und K_{13} bis K_{16}, schneiden X lediglich. Mittels $\overline{X}_B$ lässt sich eine obere Abschätzung für das Konzept angeben mit $\overline{X}_B = \bigcup_{i=6}^{n=16} K_i$. Relativ gesehen enthält $\overline{X}_B$ $\frac{37}{61} \approx 61\%$ aller Daten. Es handelt sich also um eine wirklich grobe Abschätzung für X, wenn bedacht wird, dass X real nur aus 34% aller gegebenen Daten besteht (s. Abbildung 2.4 (a)). Die B-untere Approximation $\underline{X}_B = K_{11} \cup K_{12}$ besteht aus 5 Objekten, was 9% der Daten ausmacht. Somit ergibt sich eine Differenz beider Mengen von $\overline{X}_B = \overline{X}_B - \underline{X}_B = \bigcup_{i=6}^{n=10} K_i \cup \bigcup_{j=13}^{m=16} K_j$. Dies entspricht $\frac{32}{61} \approx 52\%$ der Daten, die nicht eindeutig dem Zielkonzept zugeordnet werden können (s. Abbildung 2.4 (b)). Offensichtlich ist aus Sicht von IND_B das Zielkonzept als sehr grob einzustufen. Die Äquivalenzklassen, welche nicht von X erfasst werden sind für eine nähere Betrachtung uninteressant. Sie sind in Abbildung 2.4 (c) dargestellt und entsprechen insgesamt $\frac{24}{61} \approx 39\%$ aller Objekte aus $\mathcal{A}$.

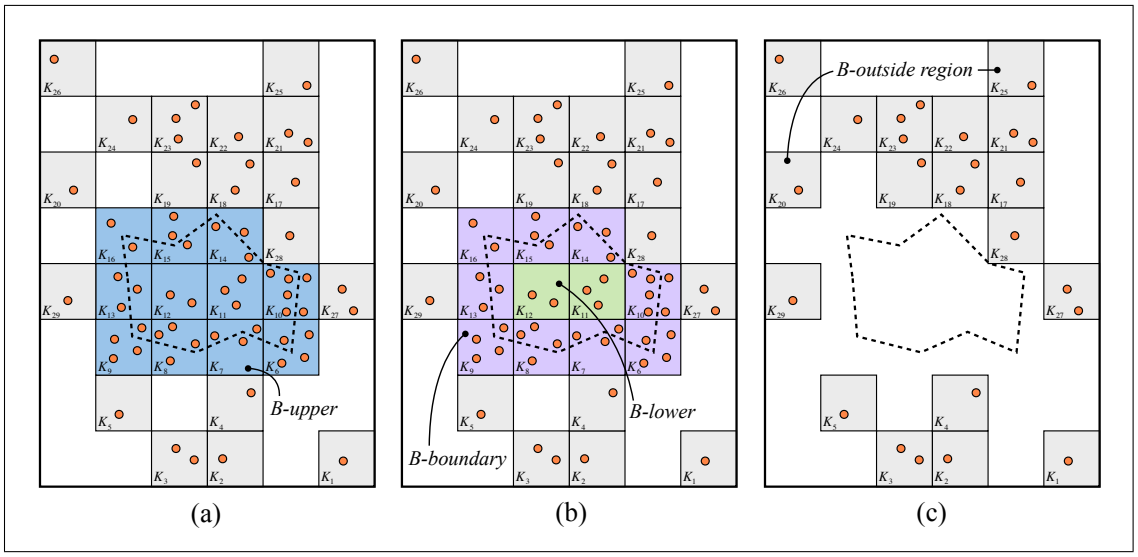


Abbildung 2.4: Schematische Darstellung eines Rough-Sets

Beispiel 2.4. Berechnung eines Rough-Sets
Es sei das DS $\mathcal{A}_d = (\mathbb{U}, A \cup \{d\})$ aus Abbildung 2.5 gegeben mit $\mathbb{U} = \{x_1, ..., x_8\}$, $A = \{Form, Größe, Farbe\}$ und mit d, dem Entscheidungsattribut. Bezüglich der gege-

-	$Form$	$Farbe$	$Größe$	d
x_1	Dreieck	grün	klein	0
x_2	Kreis	rot	groß	0
x_3	Quadrat	blau	mittel	0
x_4	Kreis	rot	groß	1
x_5	Kreis	grün	groß	0
x_6	Dreieck	blau	klein	0
x_7	Kreis	grün	groß	1
x_8	Kreis	blau	groß	1

Abbildung 2.5: Entscheidungssystem für geometrischer Objekte

benen Informationen in A lässt sich die Partition $IND_A = \{\{x_1\}, \{x_2, x_4\}, \{x_3\}, \{x_5, x_7\}, \{x_6\}, \{x_8\}\}$ ermitteln. Betrachtet man nun ein gegebenes Konzept $X = \{x \in \mathbb{U} \mid d(x) = 1\} = \{x_4, x_7, x_8\}$, so kann X mit IND_A nicht exakt beschrieben werden. Mittels der RST kann aber eine Abschätzung für X angegeben werden: Objekte der Äquivalenzklassen, die einen nicht-leeren Schnitt mit dem Konzept X haben, fallen in die B-obere Approximation $\overline{X}_A = \{x_2, x_4, x_5, x_7, x_8\}$. Die Klassen in IND_A, welche vollständig durch X beschrieben werden, sind in der B-unteren Approximation $\underline{X}_A = \{x_8\}$ enthalten. Objekte, die nicht eindeutig X zugeordnet werden können, liegen in der A-Grenzregion $\underline{X}_A = \overline{X}_A - \underline{X}_A = \{x_2, x_4, x_5, x_7\}$. Es ist also möglich, mithilfe der Informationen in A zu entscheiden, dass das Objekt

x_8 in X enthalten ist. Für die anderen Objekte $x \in \overline{\underline{X}}_A$ kann dies nicht exakt entschieden werden. Es lässt sich sehen, dass sich mit keiner anderen Kombination an Attributen $F \subset A$ ein besseres Ergebnis bezüglich der Objektklassifikation zu X erzielen lässt.

Es kann natürlich die Frage gestellt werden, ob auf einem IS $\mathcal{A} = (\mathbb{U}, A)$, $B \subseteq A$ für ein gegebenes $x \in \mathbb{U}$ und $X \subseteq \mathbb{U}$ sofort entschieden werden kann, ob die Klasse $[x]_B$ in die obere oder untere Approximation fällt. Dies lässt sich mit der sogenannten *Rough Membership Function* (dt. unexakte Zugehörigkeitsfunktion) ermitteln:

Definition 2.7. Sei $\mathcal{A} = (\mathbb{U}, A)$ mit $X \subseteq \mathbb{U}$ und $B \subseteq A$. Dann ist die *Rough Membership Function* μ bezüglich X und IND_B für $x \in \mathbb{U}$ wie folgt definiert:

$$\mu_X^B(x) = \frac{|X \cap [x]_B|}{|[x]_B|} \tag{2.6}$$

Diese Funktion beschreibt das Verhältnis zwischen der Anzahl der Objekte einer Klasse $[x]_B$, welche im Zielkonzept X enthalten sind, und der Gesamtanzahl der Objekte in $[x]_B$. Es lässt sich leicht sehen, dass sich das Ergebnis von $\mu_X^B(x)$ zwischen den Zahlen 0 und 1 bewegt und somit $\mu_X^B(x) \in [0, 1]$ gilt. Verifizieren kann man dies anhand der drei möglichen Fälle, welche auftreten können: ist $[x]_B \subseteq X$, dann ist $\mu_X^B(x) = 1$. Auf der anderen Seite gilt $0 < \mu_X^B(x) < 1$, wenn $[x]_B = X_{in} \cup X_{out}$ und $X_{in} \subseteq X$, aber $X_{out} \not\subseteq X$. Bewegt sich $[x]_B$ außerhalb von X ($[x]_B \cap X = \emptyset$), dann ist der Quotient $\mu_X^B(x) = 0$ [Paw]. Mit diesem Gedankengang lassen sich daher äquivalente Beschreibungen zu der ursprünglich eingeführten Konzeptapproximation formulieren [PS94]:

Definition 2.8. Sei $\mathcal{A} = (\mathbb{U}, A)$ mit $X \subseteq \mathbb{U}$ und $B \subseteq A$. Dann lässt sich die B-untere Approximation, die B-Grenzregion sowie die B-Außenregion wie folgt berechnen:

$$\underline{X}_B = \{x \in \mathbb{U} \mid \mu_X^B(x) = 1\} \tag{2.7}$$

$$\overline{\underline{X}}_B = \{x \in \mathbb{U} \mid 0 < \mu_X^B(x) < 1\} \tag{2.8}$$

$$\mathbb{U} - \overline{X}_B = \{x \in \mathbb{U} \mid \mu_X^B(x) = 0\} \tag{2.9}$$

Beispiel 2.5. Rough-Sets und die Rough-Membership-Function
Gegeben sei die Verteilung von $\mathcal{A} = (\mathbb{U}, B)$ aus Abbildung 2.3. Man erkennt, dass beispielsweise $\mu_X^B(x) = \frac{4}{6} > 0, \mu_X^B(y) = 1$ und $\mu_X^B(z) = 0$ gilt, für $x \not\equiv_{IND_B} y \not\equiv_{IND_B} z, x \in K_{10}, y \in K_{11}, z \in K_{28}$. Es lässt sich nun unter Zuhilfenahme von μ_X^B entscheiden, dass wegen Definition 2.8: $K_{10} \subseteq \overline{\underline{X}}_B \subseteq \overline{X}_B, K_{11} \subseteq \underline{X}_B \subseteq \overline{\underline{X}}_B$ und $K_{28} \not\subseteq \overline{X}_B$.

Rückblickend auf die Beispiele 2.3 und 2.4 ist zu sehen, dass nicht immer für alle Objekte eines gegebenen IS bzw. DS anhand der gegebenen Informationen entschieden werden kann, ob sie zur Zielklassifikation gehören oder nicht. Dies gilt für die Objekte eines Rough-Sets, die genau zwischen unterer und oberer Approximation (s. Definition 2.6) liegen. Offensichtlich hat die Anzahl der Objekte in der Grenzregion etwas mit der Güte der Konzeptapproximation zu tun, was durch den Quotienten der folgenden Definition ausdrückbar ist.

Definition 2.9. Die Exaktheit (engl. *Accuracy*) der Approximation $(\underline{X}_B, \overline{X}_B)$ von $X \neq \emptyset$ ist auf $\mathcal{A} = (\mathbb{U}, A)$ mit $B \subseteq A$ und $X \subseteq \mathbb{U}$ wie folgt definiert:

$$\alpha_X^B = \frac{|\underline{X}_B|}{|\overline{X}_B|} \tag{2.10}$$

α_X^B bewegt sich im Intervall $[0, 1]$, da zum einen $|\underline{X}_B| \leq |\overline{X}_B|$ und zum anderen $|\overline{X}_B| > 0$. Ist $\alpha_X^B = 0$, dann ist offenbar $\underline{X}_B = \emptyset$. Es können also keine Objekte definitiv dem Zielkonzept zugeordnet werden. Für $0 < \alpha_X^B < 1$ können einige Objekte bezüglich X korrekt klassifiziert werden, andere nicht. Insgesamt bedeutet dies für die Fälle $0 \leq \alpha_X^B < 1$, dass das Rough-Set $(\underline{X}_B, \overline{X}_B)$ X grob beschreibt. Ist jedoch $\alpha_X^B = 1$, dann kann gesagt werden, dass $(\underline{X}_B, \overline{X}_B)$ exakt das Zielkonzept X approximiert. Folglich kann mit den vorliegenden Informationen des IS für alle Objekte eine Aussage darüber getroffen werden, ob sie zu X gehören oder nicht.

2.4 Merkmalsreduktion

Mit der Ununterscheidbarkeitsrelation ist es möglich, die Menge an Objekten in einem gegebenen IS massiv zu reduzieren. Es können nämlich die Objekte, die zu allen Repräsentanten des IS äquivalent sind, ausgeblendet werden (vgl. Kapitel 2.2). Das Ergebnis dieses Prozesses ist also letztlich die Eliminierung redundanter Information auf Objektebene. Ein weiterer Schritt beim Verwerfen unbenötigter Information in einem IS ist der Prozess der sogenannten *Feature Selection*[5] (dt. Merkmalsauswahl). Dabei geht es im Wesentlichen um das Finden von Attributen, welche für eine vollständige Klassifikation von Objekten überflüssig sind [ZY04]. Bevor diese Idee jedoch weiter präzisiert werden kann, werden im Kontext der Rough-Sets noch weitere Hilfsmittel benötigt, welche im Folgenden eingeführt und diskutiert werden.

[5] Die Disziplin der Feature-Selection lässt sich mit Hilfe vieler mathematischer Modelle beschreiben. Rough-Sets bieten dabei eine Möglichkeit [Swi01].

Definition 2.10. Sei $\mathcal{A}_d = (\mathbb{U}, A \cup D)$ mit $B \subseteq A$ und $E \subseteq D$, dann lassen sich die *Positive*, *Boundary* und *Negative Region* wie folgt berechnen:

$$\begin{aligned} POS_B(E) &= \bigcup_{X \in IND_E} \underline{X}_B \\ BND_B(E) &= \bigcup_{X \in IND_E} \overline{X}_B \\ NEG_B(E) &= \mathbb{U} - \bigcup_{X \in IND_E} \overline{X}_B \end{aligned} \tag{2.11}$$

Bei dieser Rough-Set-Betrachtung werden nun alle Entscheidungsklassen X, die IND_E induziert, als Zielkonzepte betrachtet. In die Positive-Region fallen dann die Objekte der Äquivalenzklassen, welche vollständig in den jeweiligen Konzepten $X \in IND_E$ enthalten sind. Anders formuliert kann gesagt werden, dass $POS_B(E)$ die Menge aller möglichen unteren Approximationen in $\mathcal{A}_d$ ist. Zwei Konzepte $X, Y \in IND_E$ sind immer überschneidungsfrei (s. Kapitel 2.2). Dennoch kann es passieren, dass Objekte $x \in \overline{X}_B$ auch in die Konzeptapproximation von $\overline{Y}_B$ fallen und somit $x \in \underline{X}_B, \underline{Y}_B$ gilt. Die Menge all solcher Objekte wird durch $BND_B(E)$ beschrieben. Letztlich enthält die Negative-Region Objekte, die nicht bezüglich IND_E klassifiziert werden können.

Beispiel 2.6. Berechnung der einzelnen Regionen
Es sei das DS $\mathcal{A}_d = (\mathbb{U}, A \cup \{d\})$ aus Beispiel 2.4 bzw. aus Abbildung 2.5 gegeben. Aus der Partition, die $IND_{\{d\}}$ induziert, ergeben sich zwei Klassen $X = \{x \in \mathbb{U} \mid d(x) = 0\} = \{x_1, x_2, x_3, x_5, x_6\}$ und $Y = \{x \in \mathbb{U} \mid d(x) = 1\} = \{x_4, x_7, x_8\}$. Betrachtet man nun diese beiden Klassen als Konzepte, so lassen sich die untere und obere Approximation sowie die Grenzregion ermitteln:

$$\begin{aligned} &\underline{X}_A = \{x_1, x_3, x_6\}, && \overline{X}_A = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}, && \underline{X}_A = \{x_2, x_4, x_5, x_7\}, \\ &\underline{Y}_A = \{x_8\}, && \overline{Y}_A = \{x_2, x_4, x_5, x_7, x_8\}, && \underline{Y}_A = \{x_2, x_4, x_5, x_7\} \end{aligned}$$

Insgesamt ergeben sich aus dieser Zwischenberechnung die Positive-, Boundary- und Negative-Region:

$$\begin{aligned} POS_A(\{d\}) &= \underline{X}_A \cup \underline{Y}_A &&= \{x_1, x_3, x_6, x_8\} \\ BND_A(\{d\}) &= \overline{X}_A \cup \overline{Y}_A &&= \{x_2, x_4, x_5, x_7\} \\ NEG_A(\{d\}) &= \mathbb{U} - (\overline{X}_A \cup \overline{Y}_A) &&= \emptyset \end{aligned}$$

Es fällt auf, dass die Negative-Region $NEG_A(\{d\})$ keine Objekte enthält. Die Begründung dafür liegt in der Betrachtung der Partition $IND_{\{d\}}$, denn sie deckt alle Objekte im Universum ab ($IND_{\{d\}} = \mathbb{U}$).

Anhand der Informationen im IS aus Beispiel 2.6 ist es möglich, vier von acht Objekten ihren Konzepten eindeutig zuzuordnen (s. $POS_A(\{d\})$). Für die anderen vier verbleibenden Objekte reichen die Merkmale, welche die Objekte aufweisen, nicht aus (s. $BND_A(\{d\})$). Setzt man nun die korrekt klassifizierbaren Objekte in das Verhältnis zur Gesamtheit aller Objekte, so lässt sich mittels der RST die Abhängigkeit bzw. Unabhängigkeit von Attributen ermitteln. Schließlich sind deren Merkmalswerte für die korrekte Klassifizierung der Objekte verantwortlich.

Definition 2.11. Seien B und E Attributmengen in $\mathcal{A} = (\mathbb{U}, A), B, E \subseteq A$, dann lässt sich der Grad der Abhängigkeit (engl. *Degree of Dependency*) zwischen B und E durch den Quotienten $\gamma_B(E) \in [0, 1]$ angeben:

$$\gamma_B(E) = \frac{|POS_B(E)|}{|\mathbb{U}|} \tag{2.12}$$

Die Abhängigkeit von E zu B mit dem Grad $\gamma_B(E)$ ist dann: $B \xrightarrow{\gamma_B(E)} E$.

Ist $B \xrightarrow{1} E$, dann ist E voll funktional abhängig von B, während $B \xrightarrow{0} E$ bedeutet, dass $B \not\rightarrow E$, also E unabhängig von B ist. Aus $0 < \gamma_B(E) < 1$ folgt, dass E partiell von B abhängt [SC01]. Es lässt sich nun überlegen, dass, wenn ein Attribut a aus $B \subseteq A$ entfernt wird, sich der Grad der Abhängigkeit zwischen $B - \{a\}$ und E möglicherweise ändert oder nicht. Dieser Gedanke führt zur *Signifikanz* des Attributs a.

Definition 2.12. Seien B und E Attributmengen in $\mathcal{A} = (\mathbb{U}, A), B, E \subseteq A$, dann sei die folgende Differenz

$$\sigma_B^E(a) = \gamma_B(E) - \gamma_{B-\{a\}}(E) \tag{2.13}$$

die Signifikanz des Attributs $a \in B$ bezüglich der Abhängigkeit zwischen B und E.

Beispiel 2.7. Abhängigkeit und Signifikanz von Attributen
Es sei das DS $\mathcal{A}_d = (\mathbb{U}, A \cup \{d\})$ aus Abbildung 2.5 gegeben. Die Signifikanz der einzelnen Attribute *Form*, *Farbe* und *Größe* ergibt sich aus:

$$\sigma_A^{\{d\}}(Farbe) = \frac{|\{x_1, x_3, x_6, x_8\}| - |\{x_1, x_3, x_6\}|}{|\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}|} = \frac{1}{8}$$

$$\sigma_A^{\{d\}}(Form) = \frac{|\{x_1, x_3, x_6, x_8\}| - |\{x_1, x_3, x_6, x_8\}|}{|\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}|} = 0$$

$$\sigma_A^{\{d\}}(Gr\ddot{o}\ddot{\beta}e) = \frac{|\{x_1, x_3, x_6, x_8\}| - |\{x_1, x_3, x_6, x_8\}|}{|\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}|} = 0$$

Daraus lässt sich ersehen, dass das Entfernen des Merkmals *Farbe* aus der Attributmenge $A = \{Form, Farbe, Gr\ddot{o}\ddot{\beta}e\}$ den Grad der Abhängigkeit verändert, denn vor dem Entfernen ist $A \xrightarrow{1/2} \{d\}$ und nach dem Entfernen ist $A - \{Farbe\} \xrightarrow{3/8} \{d\}$. Das Attribut *Farbe* spielt also eine wichtige Rolle bei der Klassifizierung der Objekte $x \in \mathbb{U}$ in die Konzepte $X \in IND_{\{d\}}$. Die Signifikanz der Attribute *Farbe* und *Größe* ist $\sigma_A^{\{d\}}(Form) = \sigma_A^{\{d\}}(Gr\ddot{o}\ddot{\beta}e) = 0$. Das bedeutet, dass die Attribute keinen wesentlichen Beitrag für die Klassifizierung liefern. Die Abhängigkeit der Attribute ändert sich nämlich nicht: $A - \{Farbe\} \xrightarrow{1/2} \{d\}$, $A - \{Gr\ddot{o}\ddot{\beta}e\} \xrightarrow{1/2} \{d\}$.

Die Erläuterungen aus Beispiel 2.7 führen dazu, dass ein oder mehrere Attribute aus einem DS für die Approximation der Zielkonzepte nicht unbedingt benötigt werden, um die Klassifikation aufrecht zu erhalten. Ein solches Attribut $a \in A$, welches verworfen werden kann, nennt sich dann überflüssig (engl. *Dispensable*).

Definition 2.13. Auf einem DS $\mathcal{A}_d = (\mathbb{U}, A \cup D)$ mit $R \subseteq A$ und $E \subseteq D$ nennt sich die Attributmenge R Redukt (engl. *Reduct*), wenn:

$$\gamma_R(E) = \gamma_A(E) \tag{2.14}$$

und zusätzlich:

$$\gamma_{R-\{a\}}(E) \neq \gamma_R(E), \forall a \in R \tag{2.15}$$

Nach dieser Definition ruft jegliches Entfernen eines Attributes a aus einem Redukt R eine Signifikanz von $\sigma_R^E(a) > 0$ hervor. R heißt dann auch *orthogonal* [HCHZ02]. Folglich gibt es keine kleinere Menge $R' \subset R$, deren Aussagekraft ausreichen würde, um die Positive-Region $POS_R(E) = POS_A(E)$ zu beschreiben. Alle Attribute in R sind unverzichtbar (engl. *Indispensable*) [ZY04]. Angemerkt sei an dieser Stelle, dass bisher der Begriff des Redukts auf Basis von DS definiert wurde, welche typischerweise mehrere Entscheidungsklassen[6] aufweisen. Aus diesem Grund wurde auch die Begrifflichkeit der Positive-Region eingeführt. In einem herkömmlichen IS $\mathcal{A} = (\mathbb{U}, A)$ ohne Entscheidungsattribut kann die Positive-Region in der aktuellen Formulierung (s. Definition 2.10) nicht als Klassifikationskriterium für ein Redukt R angewandt werden. Stattdessen könnte man die R-untere Approximation $\underline{X}_R$ als Bedingung für ein zu analysierendes Zielkonzept $X \in \mathbb{U}$ verwenden. Schließlich besteht die Positive-Region eines DS aus der mengentheoretischen Vereinigung der

[6] Ein Entscheidungssystem $\mathcal{A}_d = (\mathbb{U}, A \cup \{d\})$ mit $d : \mathbb{U} \to \mathbb{B}$ hat beispielsweise $|IND_{\{d\}}| = 2$ Entscheidungsklassen.

lokalen unteren Approximationen der Entscheidungsklassen. Eine minimale Untermenge $R \subseteq A$, für die die zwei Bedingungen $\underline{X}_R = \underline{X}_A$ und $\underline{X}_{R-\{a\}} \neq \underline{X}_R, \forall a \in R$ gelten, ist also auch ein Redukt, bezüglich der Klassifikation von X. Ohne jegliches Zielkonzept lässt sich der Begriff des Redukts R auf einem IS durch folgende Klassifizierungsbedingung einführen [BMA06]:

Definition 2.14. Auf einem IS $\mathcal{A} = (\mathbb{U}, A)$ mit $R \subseteq A$ nennt sich R Redukt, wenn:

$$IND_R = IND_A \tag{2.16}$$

sowie:

$$IND_{R-\{a\}} \neq IND_R, \forall a \in R \tag{2.17}$$

Die Aussage von Definition 2.14 besagt, dass mit R eine genau so gute Klassifikation der Objekte $x \in \mathbb{U}$ erzielt werden kann, wie mit der vollständigen Attributmenge A. R ist in dem Bezug minimal, als dass aus R kein Attribut entfernbar ist. In den weiteren Ausführungen zu diesem Thema soll aus dem Kontext ersichtlich sein, welche Art von Redukt gemeint ist.

Beispiel 2.8. Ermittlung von Redukten mittels der Signifikanz
In Beispiel 2.7 wurde bereits ein Teil der Attributmenge aus dem DS $\mathcal{A}_d = (\mathbb{U}, A \cup \{d\})$ (s. Abbildung 2.5) analysiert. Es hat sich herausgestellt, dass: $\sigma_A^{\{d\}}(Farbe) = \frac{1}{8}$, $\sigma_A^{\{d\}}(Form) = 0$ und $\sigma_A^{\{d\}}(Größe) = 0$. Das Attribut *Farbe* kann also nicht aus der Attributmenge entfernt werden, während die Attribute *Form* oder *Größe* überflüssig sind. Man erhält die zwei Attributmengen $R_1 = \{Farbe, Form\}$ oder $R_2 = \{Farbe, Größe\}$, welche immer noch die gleiche Klassifizierungskraft besitzen wie A. Weitere Versuche R_1 und R_2 zu reduzieren führen zu:

$$\sigma_{R_1}^{\{d\}}(Form) = \frac{|\{x_1, x_3, x_6, x_8\}| - 0}{|\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}|} = \frac{1}{2}$$

$$\sigma_{R_2}^{\{d\}}(Größe) = \frac{|\{x_1, x_3, x_6, x_8\}| - 0}{|\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}|} = \frac{1}{2}$$

Daraus folgt, dass aus R_1 respektive R_2 keine Attribute entfernt werden können, ohne dass sich die Klassifikation ändert. R_1 und R_2 erfüllen Definition 2.13 und sind somit Redukte. Das wohl wichtigste Attribut in diesen zwei Teilmengen ist das Attribut *Farbe*, da es sowohl in R_1 als auch in R_2 vorkommt. Bildlich gesprochen basieren R_1 und R_2 auf dem Attribut *Farbe*. Dies legt folgende Definition nahe:

Definition 2.15. Auf $\mathcal{A}_d = (\mathbb{U}, A \cup D)$ mit $E \subseteq D$ sei der mengentheoretische Schnitt aller Redukte mit *Core* (dt. Kern) bezeichnet:

$$Core = \cap Red \tag{2.18}$$

$$Red = \{R \subseteq A \mid R \textit{ ist ein Redukt}\} \tag{2.19}$$

Zur Veranschaulichung von Definition 2.15 soll die schematische Darstellung aller Redukte und des *Core* auf einem IS in Abbildung 2.6 dienen.

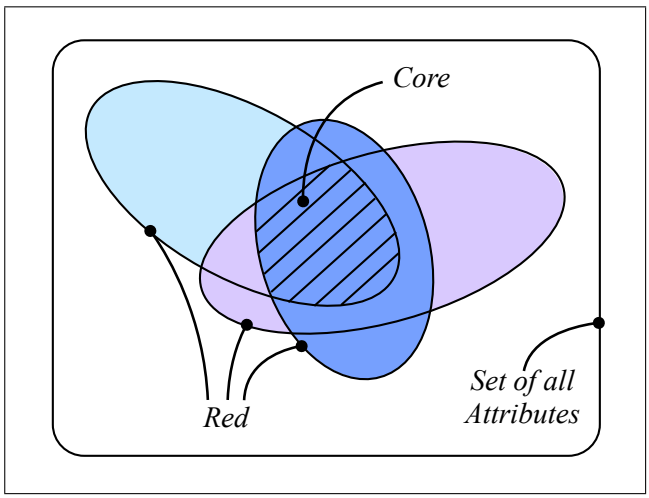


Abbildung 2.6: Schematische Mengendarstellung des *Core* und der Redukte

Der Ablauf zur Herleitung eines der erläuterten Redukttypen kann als Suchproblem aufgefasst werden. Dabei wird im Suchraum, dem sogenannten *Feature Space* (Menge aller Attributkombinationen), nach einer möglichst kleinen Attributmenge gesucht, mit der es immer noch möglich ist das festgelegte Klassifikationskriterium zu erfüllen. Es ist vorstellbar, dass dieser Suchprozess sehr aufwendig sein kann, denn der Raum ist schon für eine moderate Anzahl zu analysierender Attribute $A = \{a_1, ..., a_n\}, n \in \mathbb{N}$ sehr groß. Veranschaulichen kann man dies anhand der Potenzmengenbetrachtung von A, denn es gibt $|\mathcal{P}(A)| = 2^{|A|} = 2^n$ Attributkombinationen, welche anhand des Klassifikationskriteriums zu untersuchen sind, um die kleinste Untermenge sicher auf einem naiven Weg zu finden [Swi01, ZY04].

Anhand von Beispiel 2.7 und 2.8 ist klar geworden, dass es im Feature-Space eines IS mehrere Redukte geben kann. Findet man eines davon, ist nach Definition 2.13 jedoch nicht gewährleistet, dass das Optimale, also das Redukt mit der kleinsten Kardinalität gefunden wurde (vgl. lokale Minima-Problematik). Es kann gezeigt werden, dass die Suche nach dem kleinsten Redukt NP-schwer (engl. *NP-hard*) ist

[SR92]. Folglich ist die Entscheidung, ob eine Attributmenge ein minimales Redukt ist, mindestens genau so schwer, wie jedes Problem aus der Komplexitätsklasse NP (nichtdeterministisch polynomiell)[7] [Sip05]. Es haben sich jedoch in der Praxis einige Heuristiken etabliert, mit denen gute Lösungen gefunden werden können [KPPS99]. Auf eine kleine Auswahl an weiterführender Literatur zu diesem Thema, sei an dieser Stelle verwiesen: [Wro95, Wro98, HLH03, BMA06, CABG07, TCX07].

[7] In NP befinden sich u.a. auch solche Probleme, für die bis dato kein exaktes und gleichzeitig praktikables Berechnungsverfahren (vgl. *NP-vollständige Probleme*) angegeben werden konnte [WV06].

3 Überblick und Stand der Forschung

Nachdem Kapitel 2 eine Einführung in die wesentlichen Ideen der klassischen *Rough Set Theory* gegeben hat, beleuchtet dieses Kapitel weitere Systeme und Methoden aus den Forschungsbereichen, welche in unmittelbarem Zusammenhang mit dieser Arbeit stehen.

3.1 Objektrelationale Datenbanken

Dieser Abschnitt bietet einen grundlegenden Überblick über die Komponenten traditioneller Datenbanksysteme (DBS). Ferner werden aber auch weiterführende Konzepte aus dem Gebiet der Datenbanken vorgestellt, die objektorientierte Aspekte berücksichtigen.

3.1.1 Relationale Datenbanksysteme

Edgar F. Codd gilt als der Erfinder[1] des relationalen Datenmanagements. Zu Beginn der 1970er Jahre veröffentlichte er in [Cod70] ein flexibles Modell für das strukturierte Speichern von Daten, den Datenbanken. Die Kernidee dieses Modells sind mehrstellige *Basisrelationen* (Tabellen), definiert auf Wertebereichen mit atomaren Werten. Unter Zuhilfenahme dieser Relationen ist es möglich, Entitäten und deren Beziehung aus der realen Welt untereinander abzubilden. Operiert werden kann auf diesem relationalen Modell mit Hilfe der relationalen Algebra, die formale Werkzeuge für das Anfragen und Verknüpfen der gespeicherten Daten liefert [HSS07]. Zur Veranschaulichung eines solchen Modells sei zunächst Abbildung 3.1 zu betrachten. Sie stellt ein mögliches Szenario aus der realen Welt in Form eines *Entity-Relationship-Diagramms* (ERD) dar. Dabei wird die Interaktion von Benutzern am Computer über Relationen modelliert.

[1] Streng genommen muss erwähnt werden, dass David L. Childs in [Chi68] zwei Jahre vor Codd erste Ergebnisse im Umgang mit der mengentheoretischen Datenorganisation erzielen konnte.

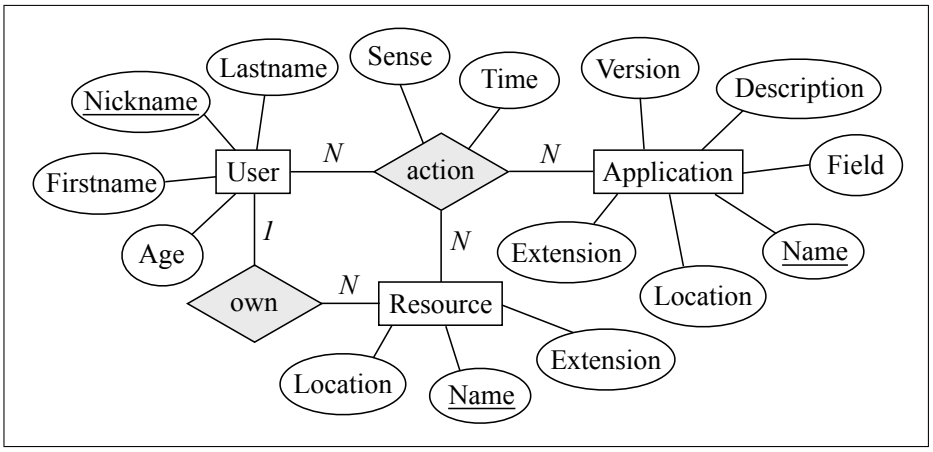


Abbildung 3.1: Entitäten abgebildet in ein relationales Modell

Diese grafische Repräsentation enthält die Entitäten: *User*, *Resource* und *Application*, modelliert als Basisrelationen der Form:

$$User \subseteq String \times String \times String \times Integer$$

$$Resource \subseteq String \times String \times String$$

$$Application \subseteq String \times String \times String \times String \times String \times Integer$$

Die Beziehungen zwischen den einzelnen Entitäten lauten: *own* und *action* und lassen sich grafisch beispielsweise durch folgende Ausprägungen (s. Abbildung 3.2) visualisieren mit $\{a_1, a_2, a_3, a_4\} \subseteq action$ und $\{o_1, o_2\} \subseteq own$:

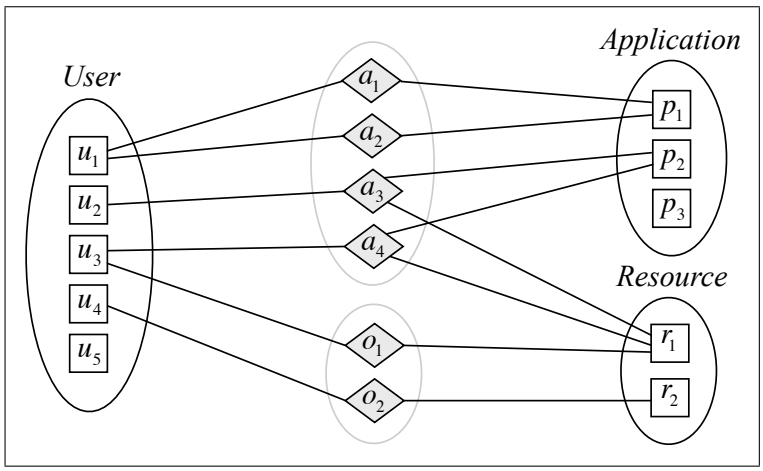


Abbildung 3.2: Instanzen von Entitäten mit ihren Beziehungen

In diesem Szenario hat Benutzer u_1 bestimmte Aktionen (hier: a_1, a_2) in Programm p_1 getätigt, während die Benutzer u_2, u_3 mit dem Programm p_2 die Datei r_1 verändert haben. Möchte man nun ermitteln welche *User* zu unterschiedlichen Zeitpunkten

auf $r_1 = (\text{'} \sim \text{/docs'}, \text{'todo.txt'}, \text{'txt'}) \in Resource$ zugegriffen haben, kann dies algebraisch durch den allgemein gehaltenen Ausdruck:

$$\Pi_{\{Nickname\}}(\sigma_{Resource.Name=\text{'todo.txt'}}((User \bowtie action) \bowtie Resource)) \tag{3.1}$$

notiert werden, wobei $\bowtie$ für den natürlichen Verbundoperator[2] steht, der die Entitäten $User$ und $Resource$ über deren Beziehung $action$ verschmilzt. Die Selektion σ filtert genau die Tupel aus der Ergebnisrelation von $(User \bowtie action) \bowtie Resource$, die mit dem gegebenen Selektionsprädikat aus Ausdruck 3.1 übereinstimmen. Durch Anwendung der Projektion Π auf die Selektion werden nun alle Attribute, die nicht im Index von Π stehen ausgeblendet. Das finale Ergebnis von Ausdruck 3.1 bezogen auf die Ausprägungen in Abbildung 3.2 ist nun eine Relation $Q \subseteq String$, welche die Entitäten u_2 und u_3 reduziert auf ihr Merkmal $Nickname$ enthält.

Bis heute liefert dieses theoretisch formulierte Modell von Codd die Grundlage für viele praktische Datenbanksysteme (vgl. z.B. DB2 9.5[3], Oracle 11g[4], PostgreSQL 8.4[5], SQL Server 2008[6]). Nach den Einschätzungen von Kemper und Eickler liegt der Grund für den Erfolg des relationalen Modells in seiner strukturellen Einfachheit [KE04].

Für einen Mehrbenutzerbetrieb auf einer Datenbank reichen die bisher vorgestellten Mechanismen für das Organisieren und Anfragen von Daten unter Umständen nicht aus (vgl. Datenintegritätsprobleme). Deshalb integrieren alle namhaften Datenbankhersteller zwischen der Zugriffsschnittstelle zur Datenbank und der eigentlichen Datenbank eine logische Schicht. Diese Schicht nennt sich Datenbankmanagementsystem (DBMS) und ist ein komplexes Software-System, bestehend aus vielen Teilkomponenten. Nach Heuer et al. sind die folgenden Punkte wichtige Aufgaben eines DBMS [HSS07]:

- **Konsistenzüberwachung:** stellt die Datenintegrität sicher
- **Zugriffskontrolle:** verwaltet Ressourcen im Bezug auf Sicherheit
- **Transaktionsmanagement:** stellt Konzepte nach dem ACID[7]-Prinzip bereit
- **Datensicherung:** ermöglicht die fehlerfreie Wiederherstellung von Daten

[2] Der natürliche Verbund benötigt als Hinweis für das Verschmelzen zweier Relationen in jeder der Relationen jeweils mindestens ein gleiches Attribut. Dies sei hiermit vorrausgesetzt.
[3] `http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/`, Stand: 26.07.09
[4] `http://www.oracle.com/technology/products/database/oracle11g/`, Stand: 26.07.09
[5] `http://www.postgresql.org/`, Stand: 26.07.09
[6] `http://www.microsoft.com/germany/sql/2008/`, Stand: 26.07.09
[7] Atomicity Consistency Isolation Durability

Abbildung 3.3 illustriert dazu ein stark vereinfachtes Architekturmodell eines DBS mit ihrer Interaktion zu externen Systemen.

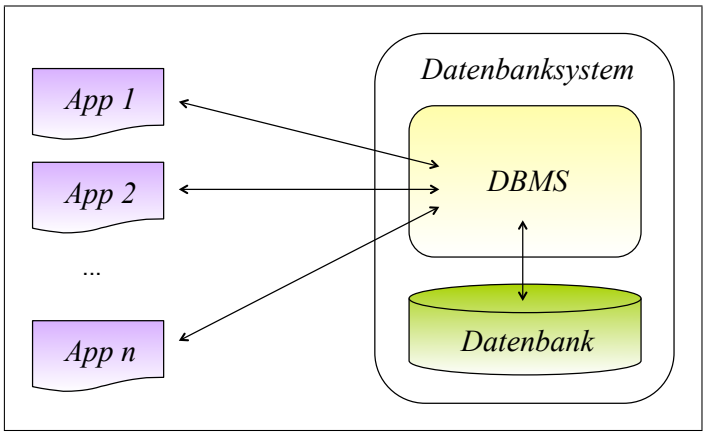


Abbildung 3.3: Komponenten eines Datenbanksystems [ES00]

3.1.2 Objektrelationale Erweiterungen

Die Mechanismen von aktuellen Implementierungen für relationale Datenbanksysteme (RDBS) bilden aus Sicht der Mengentheorie ein mächtiges Werkzeug für das effiziente Persistieren und Anfragen von Daten. In den 1980er Jahren wurde jedoch deutlich, dass relationale Datenbanken für komplexe Zusammenhänge (z.B. aus dem Ingenieurswesen oder aus dem Bereich der semantischen Modellierung) nicht ausreichen [Cod79, LKM+85]. Aus diesem Grund wurden Unternehmungen angestrebt, diese Unzulänglichkeiten im praktischen Interesse zu überwinden. Das wesentliche Forschungsresultat waren objektorientierte Datenbanksysteme (OODBS). Diese Systeme waren in der Lage, die oben genannten Problematiken durch komplexe Objektstrukturen zu modellieren. Bedingt durch den Paradigmenwechsel brachten diese Systeme jedoch den Nachteil mit sich, dass unverzichtbare RDBS-Vorteile aufgegeben werden mussten [HSS07]. Mitte der 1990er Jahre etablierten sich DBS, die bis heute unter dem Begriff objektrelationale Datenbanksysteme (ORDBS) bekannt sind [SM99, MW03, TS05]. Diese Art von Systemen verfolgt nicht streng den Gedanken der Objektorientierung (OO), sondern nach der Meinung von Heuer et al. zwei wesentliche Punkte [HSS07]:

- Ausgehend vom erfolgreichen relationalen Modell, soll dieses mit Konzepten der OO angereichert werden.
- OO-Anwendungen sollen über objektrelationale-Mappings (ORM) auf die stabilen Strukturen des relationalen Modells zugreifen.

Im Folgenden wird auf den erstgenannten Punkt weiter eingegangen, da der zweite Punkt heutzutage durch ORM-Frameworks wie z.B. Hibernate[8] oder ADO.NET Entity[9] unterstützt wird.

Die Kernerweiterungen des RDBS mit Aspekten der OO wurden durch den SQL[10]-Standard SQL:1999 vom amerikanischen *National Institute of Standards and Technology* (NIST) eingeführt, der sich im Wesentlichen von seinem Vorgänger SQL:1992 durch neue Typen, die Individualisierung dieser Typen, deren Umwandlung, Methoden[11] und der Typisierung von Relationen (Objekttabellen) mit Subtabellenbildung unterscheidet. In SQL:2003 wurde der bestehende Standard SQL:1999 u.a. durch Objektidentitäten und tabellenwertige Funktionen erweitert [Tür03].

Dennoch gelten aus Sicht von Heuer et al. die objektrelationalen SQL-Standards des NIST eher als Referenz. Diese Aussage findet Begründung in der Beobachtung, dass die verschiedenen Hersteller mit ihrer jeweiligen Implementierung von ORDBS im Bezug auf die einheitliche Syntax und die funktionalen Erweiterungen von RDBS zu ORDBS sehr stark vom Standard abweichen [HSS07].

Um die Vorteile von ORDBS im Vergleich zu traditionellen RDBS bei der Modellierung von Szenarien aus der realen Welt herauszustellen, zeigt Abbildung 3.4 eine modifizierte Variante des Relationsschemas aus Abbildung 3.1, in der einige OR-Konzepte Anwendung finden. Es wird auf den ersten Blick deutlich, dass eine neue Relation *File* hinzugekommen ist, welche die Verallgemeinerung der Relationen *Application* und *Resource* darstellt. Der Basisdatentyp für das Attribut *Location* ist nun nicht mehr *String*, sondern der individuelle Typ *Path*. Ebenfalls gibt es eine Modifikation in der Relation *Resource*, welche den neuen Datentyp *Binary Large Object* (BLOB) als Basis für das Speichern des Inhalts der *Resource* verwendet. Letztlich wird in diesem Beispiel über Methoden das ursprüngliche relationale Modell mit Funktionslogik angereichert, was in reinen RDBS nicht möglich ist.

[8] `https://www.hibernate.org`, Stand: 26.07.09
[9] `http://www.microsoft.com/sqlserver/2008/en/us/ado-net-entity.aspx`, Stand: 26.07.09
[10] Structured Query Language
[11] Hiermit sind funktionale Erweiterungen auf der Datenbankseite gemeint, so dass sich das System mit Elementen der prozeduralen Programmierung anreichern lässt. Auf diese Weise lassen sich die Restriktionen von SQL ausräumen und es zu einer vollständigen Programmiersprache ausbauen.

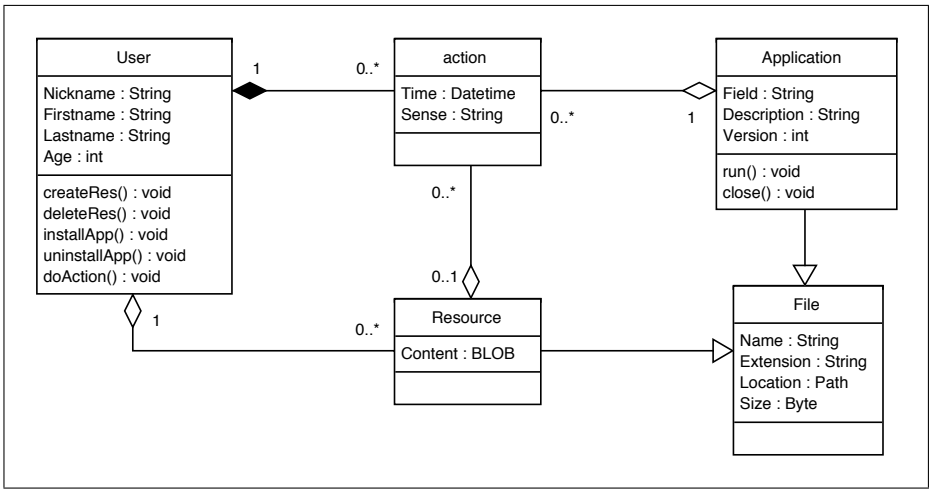


Abbildung 3.4: Relationales Modell erweitert mit OO-Konzepten in UML-Notation

3.2 Knowledge Discovery in Databases

Dieses Kapitel umfasst einen Einblick in das Gebiet des *Knowledge Discovery in Databases* (KDD). Dabei werden die Phasen dieses iterativen Prozesses aufgezählt und kurz erläutert. Letztlich wird auf das Data-Minig (DM) als Hauptdisziplin des KDD näher eingegangen.

3.2.1 Zentrale Elemente

Das massive Datenaufkommen in der wissenschaftlichen Analyse hat in den 1990er Jahren ein neues interdisziplinäres Forschungsgebiet hervorgebracht, dessen Kernkonzept die Schnittstelle zwischen Analysten und den großen Mengen an Daten bilden soll [FHS96]. Dieses Gebiet ist unter dem Namen *Knowledge Discovery in Databases* bekannt und beschäftigt sich mit der (semi)-automatischen Extraktion von Wissen aus Datenbanken, welches nach Ester und Sander folgende Eigenschaften besitzen sollte [ES00]:

- Gültigkeit (im statistischen Sinne)
- Unbekanntheit
- Nützlichkeit

KDD kann als iterativer Prozess aufgefasst werden, dessen Bestandteile im Folgenden kurz erläutert werden [Düs06]:

- **Selektion:** Ausgehend von einem massiven Datenbestand muss eine Vorselektion der Daten in eine kleinere zu analysierende Teilmenge stattfinden. Die Selektion fokussiert sich auf die Daten, aus denen bisher unbekanntes Wissen extrahiert werden soll.
- **Vorverarbeitung:** Die Daten liegen nach der Selektionsphase meist noch in einem unbrauchbaren Format vor. Ziel der Vorverarbeitung ist es, die Datenmenge in einen konsistenten bzw. vollständigen Zustand zu überführen.
- **Transformation:** In diesem Schritt wird versucht, unbenötigte Informationen aus dem brauchbaren Datenbestand zu entfernen. Typische Transformationen sind die Merkmalsauswahl (vgl. Kapitel 2.4) und die Diskretisierung von Wertebereichen bestimmter Attribute.
- **Data-Mining:** Es wird meist versucht, mittels bekannter Algorithmen Datenmuster aus den adäquaten Daten zu extrahieren. Eine detaillierte Beschreibung folgt in Kapitel 3.2.2.
- **Interpretation:** Die Ergebnisse des Data-Mining werden in dieser Phase evaluiert und interpretiert. Besitzen die Ergebnisse die oben genannten Eigenschaften für Wissen des KDD, so kann das Ergebnis als neues Wissen dokumentiert werden.

Die Ergebnisse einer KDD-Iteration können gut, manche aber auch beliebig schlecht und unbrauchbar sein. Aus diesem Grund bietet KDD die Möglichkeit vom Interpretationsschritt zu jeder beliebigen vorangegangen Phase zurück zu springen [ES00]. Folglich geht KDD also nicht davon aus mit dem erstbesten Durchgang des Prozesses immer gute Resultate zu erhalten. Zur grafischen Veranschaulichung des iterativen KDD-Prozesses sei auf Abbildung 3.5 verwiesen.

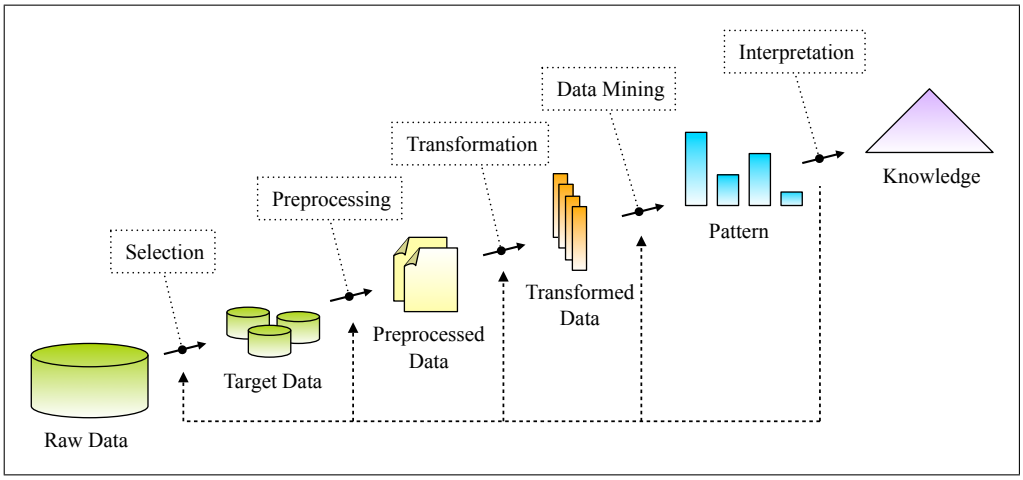


Abbildung 3.5: Prozess des *Knowledge Discovery in Databases* [FPSS96]

3.2.2 Data-Mining

Viele praxisnahe Lehrbücher, u.a. [WF01, BL04, CCF+09], motivieren die Thematik des DM als eigenständigen Prozess zur Extraktion von strukturellen Mustern aus zugrunde liegenden Datenquellen. Diese Vorgehensweise findet in den unterschiedlichen Anwendungsgebieten der jeweiligen Literatur durchaus Berechtigung. Aus der Sicht von Fayyad et al. ist DM jedoch als integraler Bestandteil des KDD-Prozesses (vgl. Kapitel 3.2.1) zu sehen, der sich durch das Anwenden von spezifischen Algorithmen auszeichnet und letztlich zu neuem, bisher unbekanntem Wissen führen soll [FPSS96]. Die Frage, ob aus DM-Aktivitäten Wissen entdeckt werden kann, ist nach Peterson kritisch zu betrachten. Ihrer Ansicht nach kann Wissen nicht entdeckt werden. Vielmehr können Erkenntnisse aus dem DM gewonnen werden, welche das bisherige Wissen erweitern [Pet05]. Im Folgenden sollen, stellvertretend für wichtige Analyseverfahren des DM, die Klassifikation und die Clusteranalyse vorgestellt werden.

Klassifikation

Die Klassifikation ist eine bekannte Aufgabe aus dem maschinellen Lernen und wird häufig auch als beaufsichtigtes Lernen verstanden. Informell geht es dabei um das Finden einer Zuordnung zwischen Objekten eines Datenbestands zu vordefinierten Klassen [ES00]. Damit diese Aufgabe bewältigt werden kann, ist für einen Teil der Objekte deren Klassenzugehörigkeit als Hilfestellung bekannt, für die restliche Menge nicht (vgl. Kapitel 2.1). Die Schwierigkeit besteht nun darin, aus den bekannten Zugehörigkeiten ein Modell (Klassifikator) zu erzeugen, das in der Lage ist, die Klassen ungesehener Objekte korrekt vorherzusagen. Das Erstellen des Modells geschieht häufig auf Basis von Verallgemeinerungsstrategien (vgl. induktives Lernen) [SN04]. Bildlich kann von einem Lernprozess gesprochen werden, wobei ein Lernalgorithmus (engl. *Learner*), unterstützt durch einen Lehrer, aufgrund von Beispieldaten trainiert wird [Alp08]. Eine eher formale Sichtweise zu dieser Thematik wurde bereits in Kapitel 2.3 mit dem Konzeptlernproblem eingeführt, was im Wesentlichen dem Lernen der Zuordnungsvorschrift von Objekten zu deren Klassen, also dem Lernen der Klassifikation, entspricht [Mit97].

Clusteranalyse

Im Gegensatz zur Klassifikation, sind bei der Clusteranalyse die Klassen, in die die Objekte fallen, nicht bekannt [BC06]. Demnach ist die Aufgabe der Clusteranalyse nicht, eine Klassifikation zu lernen, sondern das Finden von gruppenähnlichen

Strukturen (Cluster, Klassen, Hierarchien) aus den zugrunde liegenden Daten ohne Hilfe eines Lehrers oder Experten [Seg08]. Aus diesem Grund gehört das Clustering zur Aufgabengruppe des unbeaufsichtigten Lernens (engl. *Unsupervised Learning*). Im Allgemeinen jedoch ist nach Russell und Norvig ohne Hilfe kein Lernalgorithmus in der Lage, etwas zu lernen, da er seine Aufgabe nicht kennt [SN04]. Anweisungen an eine Clustermethode können unterschiedliche Formen annehmen. In einigen Anwendungen reicht es z.B. aus, dem Lernverfahren die Anzahl der zu bildenden Cluster mitzuteilen. Anderen Algorithmen wird eine Funktion als Hilfestellung gegeben, die die Distanz zwischen Objekten messen kann. Ein Überblick über verschiedene Distanz- bzw. Ähnlichkeitsmaße auf unterschiedlichen Merkmalsausprägungen ist beispielsweise in [GM98] und [MR05] zu finden. Ist ein solches Maß über der Objektmenge definiert, sollte nach Bacher der Vorgang des Clusterns unter Berücksichtigung folgender Eigenschaften durchgeführt werden [Bac96]:

- Objekte in einem Cluster sind möglichst ähnlich zueinander
- Objekte in unterschiedlichen Clustern sind möglichst verschieden voneinander

Über die Jahre haben sich verschiedene Cluster-Algorithmen aus den verschiedensten Anwendungsgebieten entwickelt, die nach Fraley und Raftery im Wesentlichen in zwei Kategorien fallen [FR98]:

- **partitionierende Verfahren:** Ziel dieser Verfahren ist es, die gegebenen Objekte so zu gruppieren, dass disjunkte Teilmengen von Objekten im Datenraum entstehen [GM98]. Es entstehen „Inseln“ ähnlicher Objekte. Das wohl prominenteste partitionierende Clusterverfahren ist der sogenannte k-Means-Algorithmus für numerische Daten von MacQueen [Mac67]. Bei dieser Methode wird a priori festgelegt, dass k Cluster, repräsentiert durch Referenzpunkte (Centroide), in m Durchläufen gefunden werden sollen. Ausgehend von einer initialen Verteilung der k Centroide im Raum werden in jeder Iteration die Objekte ihrem nächstliegenden Centroid zugeordnet. In jedem der folgenden $(m-1)$ Durchläufe wird für jedes Centroid seine neue Position anhand des Mittelwerts der im Cluster befindlichen Objekte berechnet. Modifizierte Varianten des k-Means-Algorithmus finden sich zum Beispiel in [KR87] und [Hua98], wobei die adaptierte Methode von Huang auch symbolische Daten verarbeiten kann.
- **hierarchische Verfahren:** Bei Methoden dieser Art geht es um das Finden ineinander verschachtelter Mengen. Algorithmen, die nach solchen Hierarchien im Datenbestand suchen, können anhand ihrer Suchstrategie kategorisiert werden [Pet05]. Es wird unterschieden in:

- agglomerative hierarchische Verfahren: Ausgehend von einer feinen initialen Partitionierung des Datenraums wird versucht, diese in jeder Iteration zu vergrößern, so dass letztlich wieder die Grundmenge entsteht (vgl. Bottom-Up-Strategie).
- divisive hierarchische Verfahren: Ausgehend von der Grundmenge findet eine Unterteilung der Objekte in immer feinere Teilmengen statt (vgl. Top-Down-Strategie).

3.3 Vorangegangene Arbeiten

Der Forschungsbereich der RST hat in den letzten 30 Jahren stetig neue Erkenntnisse geliefert. Es entstanden mathematische Erweiterungen des traditionellen Rough-Set-Modells im Bezug auf Toleranz, neue Methoden für die Mustererkennung aus Daten und Software-Systeme, welche die klassischen, aber auch neuen Sichtweisen und Verfahren der RST implementieren. Dieses Kapitel stellt einige wichtige Forschungsarbeiten vor, die auch, aber nicht nur, mit der traditionellen RST die Grundlage dieser Arbeit bilden.

3.3.1 Rough Sets und Datenbanken

Die RST und RDBS stammen aus unterschiedlichen Forschungsrichtungen. Während sich die RST mit der Extraktion von Mustern unter Unsicherheit beschäftigen, fokussieren sich RDBS auf die Speicherung von Daten und das effiziente Anfragen dieser als Information [Lin97]. Die gemeinsamen Eigenschaften beider Ansätze sind Tabellen, respektive Relationen, welche dazu genutzt werden, Wissen über die jeweilige Domäne zu repräsentieren. Die meisten Publikationen, die sich mit dieser Thematik beschäftigen, verfolgen das Ziel, Aspekte der RST in die Domäne der RDBS zu transformieren, um die effiziente Algorithmik der relationalen Algebra für Berechnungen der RST auszunutzen.

Mitte der 1990er Jahre war Tsau Y. Lin einer der ersten Autoren, der sich der oben genannten Aufgabe stellte. In [Lin96] beschreibt er eine Methodik, mit der aus einer sehr großen Datenbankrelation schnell Assoziationsregeln abgeleitet werden können. Lin zu Folge bringt die klassische und gleichzeitig elegante Regelextraktion mittels RST [Paw91] den Nachteil mit sich, dass sie auf einer einzigen Tabelle basiert. Dies birgt die Gefahr, dass für große Datenbestände die Tabelle nicht vollständig im

Hauptspeicher verarbeitet werden kann und somit diese Form der Erzeugung von Regeln nicht praktikabel genug ist. Vielmehr versucht Lin, iterativ, durch Ausnutzen von Abhängigkeiten in den Daten, kleinere Datenmengen zu erzeugen, die im Speicher gehalten werden können. Erfüllen diese eine Mindestanforderung, werden sie in der nächsten Iteration weiter berücksichtigt, so dass sich letztlich eine Regelmenge ergibt, die dem Ergebnis des klassischen Ansatzes im Wesentlichen gleicht. Deshalb spricht er auch von einer *Bottom Up Rough Set Methodology*. In einer weiteren Veröffentlichung zu diesem Thema, zeigen Lin und Chen anhand praktischer Testergebnisse, dass dieses Modell für eine große Datenmenge anwendbar ist [LC96].

Eine andere Möglichkeit der Extraktion von Regeln lässt sich mit Hilfe einer modifizierten Variante der traditionellen Konzeptapproximation aus Kapitel 2.3 erreichen [FBRS96]. Ursprünglich eingeführt in [Zia93], handelt es sich bei den so genannten *Variable Precision Rough Sets* (VPRS) um eine generalisierte Betrachtung der RST, in der ein relativer Klassifikationsfehler bezüglich des Zielkonzepts erlaubt ist. Diese tolerantere Konzeptapproximation setzen Fernández-Baizán et al. in [FBRS96] unter Zuhilfenahme von SQL ein. Indirekt formulieren sie dadurch erstmals eine Abbildung von Rough-Sets in die Sprache der relationalen Algebra. Das wesentliche Ergebnis dieser Arbeit sind zwei Algorithmen, mit denen es möglich ist, scharfe und charakteristische Regeln aus Instanzen der variablen Lower- und Upper-Approximation zu extrahieren. Sie sind eingebettet in das ebenfalls in [FBRS96] vorgestellte Data-Mining-System *Rough Set Data Miner* (RSDM), dass den Zugriff auf diverse Datenbanken ermöglicht. Weitere Details zum RSDM finden sich in [FBRSP98, FBRW98].

In [HLH03] wird ein Ansatz vorgestellt, der die relationale Algebra ausnutzt, um effizient Redukte (vgl. Kapitel 2.4) aus den Merkmalen einer Datenbanktabelle zu finden. Hu et al. motivieren ihr Vorgehen anhand der Tatsache, dass sehr viele bestehende Algorithmen für die Reduktberechnung auf flachen Dateistrukturen operieren und bisher nur wenig Bestrebungen in der Literatur zu finden sind, die RST mit der effizienten Algorithmik moderner RDBS zu nutzen. Die Idee in [HLH03] für das Finden eines Redukts basiert im Wesentlichen auf den Überlegungen von Fernández-Baizán et al. [FBRS96] (s. vorangeganger Paragraph), woraus bereits eine generische Implementierung der Ununterscheidbarkeitsrelation (vgl. Kapitel 2.2) mittels SQL hervor geht. Für das Finden eines Redukts werden in einem ersten Schritt die Anzahl der Äquivalenzklassen der Basisrelation ermittelt. Dann werden sukzessive alle Attribute darauf hin überprüft, ob ihre Löschung (durch den Projektionsoperator) einen Effekt auf die Partitionierung hat. Es entsteht letztlich in quadratischer Zeit die Menge des *Core* (vgl. Kapitel 2.4), der als Ausgangspunkt für das Finden eines

Redukts dient. Im letzten Schritt werden ausgehend vom *Core* weitere Attribute dieser Menge hinzugefügt und geprüft, ob das Klassifikationskriterium für das Redukt erfüllt ist. Das von Hu et al. präsentierte Rough-Set-Modell setzt allerdings voraus, dass der Datenbestand der zu analysierenden Relation konsistent ist [HLH03], was in vielen reellen Anwendungen nicht der Fall ist.

3.3.2 Clustering und Rough Sets

Beim Clustering geht es im Allgemeinen darum, Gruppierungen aus Daten zu erzeugen. Dabei sollen sich die Charakteristika der Daten einer Gruppe ähneln, während Daten unterschiedlicher Gruppen möglichst heterogen sein sollen. Im ursprünglichen Sinne korrelieren die Ideen des Clustering und der RST nur unwesentlich miteinander. Im letzten Jahrzehnt hat sich jedoch vermehrt gezeigt, dass die verschiedensten mathematischen Konzepte der RST für viele Anwendungsbereiche, in denen Daten geclustert werden müssen, hilfreich sein können. Es muss jedoch Erwähnung finden, dass die Verwendung von Methoden der RST im Bezug auf Clustering keinem allgemeinen Trend folgt. Viele haben jedoch miteinander gemein, dass sie auf soliden und wohlbekannten hierarchischen oder partitionierenden Clustering-Verfahren basieren.

In [KK04] beispielsweise verwenden Kumar und Krishna für das Finden von Zugriffsmustern auf Web-Seiten einen hierarchischen agglomerativen Algorithmus. Hierbei wird versucht anhand von gegebenen Aktionssequenzen von Benutzern jene Folgen zu gruppieren, die ähnlich zu denen anderer Benutzer sind. Somit kann dann ermittelt werden, welche Benutzer ähnliche Interessen haben bzw. mit einem gewissen Grad an Unsicherheit vorausgesagt werden, dass Benutzer im weiteren Verlauf eine bestimmte Web-Seite besuchen. Die wesentliche Idee beim Vorgehen von Kumar und Krishna ist die Verwendung einer Toleranzrelation[12], ursprünglich eingeführt in [SS96]. Sind Aktionssequenzen über einen vordefinierten Schwellwert ähnlich zueinander, fallen sie in eine Toleranzklasse. Die Toleranzrelation wird dann immer wieder auf die zuvor erzeugte Partition angewandt. Es entstehen sogenannte *Similarity Upper Approximations*, die jeweils miteinander verglichen und bei Gleicheit agglomerativ zusammengefasst werden.

Ein partitionierendes Clustering-Verfahren findet in [YY06] als Ausgangspunkt für weitere Analysen von numerischen Daten Anwendung. Zunächst werden anhand des

[12] Diese binäre Relation ist keine Äquivalenzrelation, denn sie besitzt nicht die Eigenschaft der Transitivität [KK04].

gängigen k-Medoids-Algorithmus eine vordefinierten Anzahl an Clustern aus den Daten erzeugt. Sie gelten für Yang und Yang als Pseudo-Äquivalenzklassen. Zusätzlich spannt jedes der Datenobjekte sein eigenes Konzept auf, dass sich aus seinem Umkreis (Nachbarschaft) ergibt. Es entstehen somit n überlappende Teilmengen aus n numerischen Objekten. Mittels der zuvor berechneten Pseudo-Äquivalenzklassen (Cluster) lassen sich dann mit Hilfe der Konzeptapproximation der RST n untere und obere Approximationen berechnen, wobei die teilweise identischen oberen Approximationen die finale Clusterpartitionierung darstellt.

Eine andere Idee kommt ganz ohne Clustering-Algorithmus aus. In [UAJ06] nutzen Upadhyaya et al. die Ununterscheidbarkeitsrelation als Werkzeug, um die gegebenen symbolischen Daten zu partitionieren. Ausgehend von der strikten Ununterscheidbarkeitsrelation (alle m Attribute werden berücksichtigt), wird in jedem i-ten Schritt diese durch den Wegfall von i Merkmalen gelockert. Es fallen also die Objekte in ein Cluster, die in $(m - i)$ Attributen übereinstimmen. Dieses Verfahren funktioniert zwar ohne klassischen Cluster-Algorithmus, das Ergebnis hängt jedoch hochgradig von der Anzahl der Iterationen ab, die nach Upadhyaya et al. aus Erfahrung resultiert bzw. von einem Experten gegeben wird.

Weitere Bestrebungen das Clustering von Daten unterstützt durch Methoden der RST durchzuführen, stammen aus den relativ jungen Bereichen des e-Learning und e-Business: [TB07, CPC08, QW09]. Andere Ansätze wiederum versuchen ein Framework zu liefern, welches gleichermaßen numerische und symbolische Datenmengen in den Clustering-Prozess mittels RST einbindet oder gar autonom Gruppierungen erstellt: [HTOH01, BK08].

3.3.3 Weitere Software-Systeme

Die Implementierung der RST kam in den 1990er Jahren auf. Seitdem wurden viele Software-Systeme entwickelt, die zur wissenschaftlichen Analyse von Daten mittels RST beitragen. Die meisten dieser Systeme gleichen sich in ihrer Kernfunktionalität, zumal sie teilweise aufeinander basieren. Im Folgenden werden einige von ihnen kurz vorgestellt.

RSES[13] (Rough Set Exploration System) [BS05] ist 1994 aus einer Master-Thesis an der Universität Warschau entstanden und wurde seither bis zur Version 2.2. weiterentwickelt. Es ist unter den Betriebsystemen Windows XP oder Linux frei erhältlich.

[13] `http://alfa.mimuw.edu.pl/~rses/`, Stand: 19.08.09

Es unterstützt gewisse Vorverarbeitsungmethoden wie z.B. das Behandeln von Nullwerten oder das Diskretisieren der Daten. Weiter können Redukte berechnet bzw. Klassifizierer erzeugt werden. In vielen Anwendungsbereichen eingesetzt, basieren die ältesten Komponenten des Systems auf C++ Programmcode, während neuere Aspekte und die Benutzeroberfläche in Java implementiert wurden. RSES besitzt ein eigenes Dateiformat. Seit der letzten Version werden aber auch die Dateiformate von ROSETTA und WEKA unterstützt.

ROSETTA[14] [ØK97] ist an der Technischen Universität Trondheim in Zusammenarbeit mit der Universität Warschau Mitte der 1990er Jahre entstanden. Es ist ein Open-Source-Projekt und in seiner aktuellen Version unter Windows XP verfügbar. Für weitere Betriebssysteme steht eine Kommandozeilen-Version zur Verfügung. Die wesentlichen Funktionalitäten von ROSETTA sind die Vor- und Nachverarbeitung von Daten, das Erzeugen und Testen von Klassifizierern, die Unterstützung von VPRS, sowie das Erstellen von Clustern anhand von Toleranzrelationen. Für den Export von Daten stellt ROSETTA u.a. die Formate XML, Prolog, Matlab und GraphViz bereit. Der Kern des Systems ist in C++ implementiert. ROSETTA besitzt ein zusätzliches Modul, welches es ermöglicht, bestimmte Algorithmen des RSES zu verwenden.

RSL (Rough Set Library) [GS94] ist eine spartanische auf der Programmiersprache C basierende Software-Bibliothek und war ursprünglich sowohl für den akademischen wie auch für den kommerziellen Bereich gedacht. Es stellt weder Routinen für die Ein- und Ausgabe bereit, noch ist eine Benutzeroberfläche vorhanden. Dennoch besitzt RSL gängige Funktionalitäten wie die Berechnung von Redukten, das Erstellen von Regeln und das Erzeugen eines Klassifikators auf Basis dieser Regeln.

GROBIAN[15] [DG97] basiert ursprünglich auf einer nach C++ portierten Variante der RSL. Durch Veränderungen am Code sind über die Jahre allerdings nur wenig Codefragmente der RSL übrig geblieben. GROBIAN zeichnet sich neben der Funktionalität der RSL dadurch aus, dass es statistische Werkzeuge wie Entropiemaße oder die Evaluation von erstellten Regeln mittels klassicher RST bereit stellt.

ROSE[16] (Rough Set Data Explorer) [PSS+98] ist aus den Systemen RoughDAS, eine der ersten erfolgreichen Rough-Set Implementierungen, und RoughClass an der Universität Posen entstanden. Es ist in C++ geschrieben und in der aktuellen Version 2.0 auf Windows 95 installierbar. Eine Portierung auf andere Systeme

[14] `http://www.lcb.uu.se/tools/rosetta/`, Stand 19.08.09
[15] `http://www.infj.ulst.ac.uk/~cccz23/grobian/grobian.html`, Stand: 19.08.09
[16] `http://idss.cs.put.poznan.pl/site/rose.html`, Stand: 19.08.09

ist mittels Recompilierung möglich. Neben den klassischen Funktionalitäten bietet ROSE wie ROSETTA eine Erweiterung mittels VPRS an. Die Ausgabedaten speichert das System in einem ASCII-Format. Die Konvertierung in andere Formate ist dem Anwender überlassen.

Neben den genannten Systemen existieren weitere Code-Implementierungen. Dabei sei auf die folgende Auswahl verwiesen: [Bjo97, FBRSP98, Alv98, GB05].

4 Konzeption

Kapitel 2 und 3 haben sich u.a. auf die Grundlagen der *Rough Set Theory*, sowie auf bereits existente Forschungsergebnisse dieser Theorie im Bezug auf Datenbanken und Musterextraktion fokussiert. Das vorliegende Kapitel greift die Grundlagen und Ideen der zwei vorangegangenen Kapitel auf und umfasst den Schwerpunkt, ein implementierungsunabhängiges Konzept von objektrelationalen Erweiterungen für Datenbanksysteme zu entwickeln, damit das Verhalten von Benutzern im Umgang mit digitalen Inhalten am Computer effizient analysiert werden kann. Ausgangspunkt für die Analyse ist das XML-basierte *Contextualized Attention Metadata Schema* (CAMS), mit dem sich solche Verhaltensdaten erfassen lassen. Dieses wird zunächst in ein OR-Datenbankschema überführt. Das resultierende Schema gilt im weiteren Verlauf dieses Kapitels als Basis für bestehende, angepasste und neue Methoden, mit denen Verhaltensmuster extrahiert werden können. Die Methoden verfolgen dabei das Ziel, Aspekte der RST in relationale Algebra zu transformieren, um auf diesem Fundament große Mengen von Daten zu analysieren.

4.1 Relationale Umstrukturierung des CAMS

Dieser Abschnitt beschreibt eine mögliche Transformation des CAMS in ein relationales Modell, das für die Analyse mittels der RST in Datenbanken benötigt wird. Zuvor werden jedoch kurz zentrale Begrifflichkeiten über die Verwendung von *Contextualized Attention Metadata* (CAM) im Zusammenhang mit dem Forschungsprojekt *Context and Attention in Personalized Learning Environments* am Fraunhofer-Institut für Angewandte Informationstechnik FIT erläutert.

4.1.1 CAMS und das CAPLE-Framework

Das CAMS ist ein XML-Schema, welches aus Forschungsergebnissen an der Katholieke Universiteit Leuven resultiert. Es stellt eine Erweiterung des *Attention.XML-*

Standards dar, mit dem es möglich ist, den Umgang von Benutzern mit digitalen Inhalten (z.B. Web-Seiten, Word-Dokumente, Bilder etc.) am Computer über Systemgrenzen hinweg informationstechnisch zu modellieren [NWD06]. Abbildung 4.1 zeigt den Aufbau des CAMS. Details über die Semantik der einzelnen Entitäten des CAMS sind an dieser Stelle nicht weiter beschrieben. Dafür sei z.B. auf Wolpers et al. [WNVD07], respektive auf Kapitel 4.1.2 verwiesen.

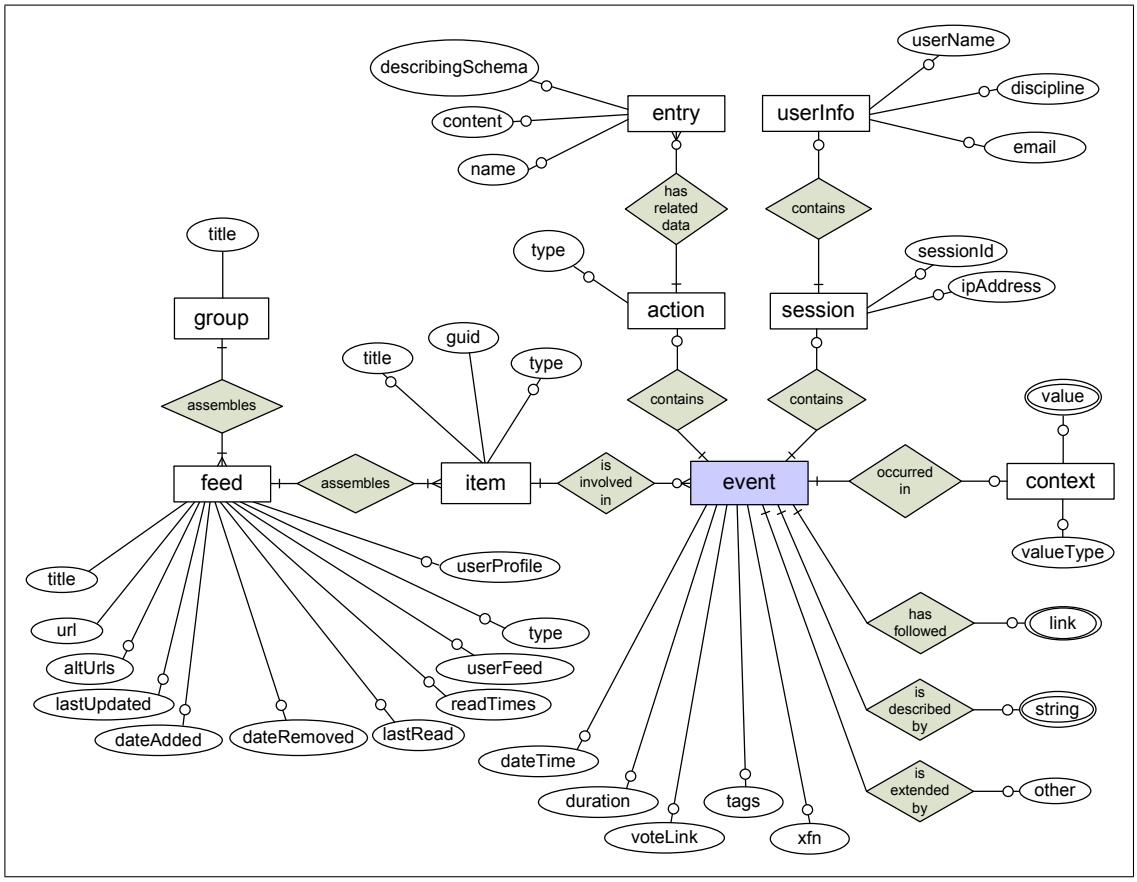


Abbildung 4.1: CAMS in ERD-Krähenfuß-Notation [WNVD07]

Das CAPLE-Forschungsprojekt beschäftigt sich u.a. mit der Implementierung eines Frameworks, dass auf Ideen von Wolpers et al. [WNVD07] basiert. Die Kernfunktionalitäten des sogenannten CAPLE-Frameworks fokussieren sich auf das serviceorientierte Speichern und Anfragen von Benutzerinteraktionen in bzw. aus Daten-Repositories. Es besteht aus einer Softwarebibliothek für den abstrahierten Zugriff auf die Repositories sowie aus einer Konfigurationsoberfläche, über die der Benutzer Einstellungen tätigen kann. Aktiv genutzt wird das Framework von externen Systemen, die auf die bereitgestellte Bibliothek zugreifen können. Für das CAPLE-Framework lassen sich diese externen Systeme in zwei Kategorien unterteilen: *Wrapper* und *Analysis Tools*. Die *Wrapper* sind Programme, die Benutzeraktivitäten für

bestimmte Anwendungen (z.B. Firefox, Skype, Thunderbird etc.) erfassen und über das Framework speichern. Die *Analysis Tools* sind für spezifische Auswertungen der gespeicherten Daten verantwortlich und nutzen das Framework für das Abfragen dieser Daten. Das CAMS dient als Kommunikationsprotokoll zwischen Framework und externen Systemen [Bee09]. Abbildung 4.2 illustriert schematisch die Architektur des Frameworks.

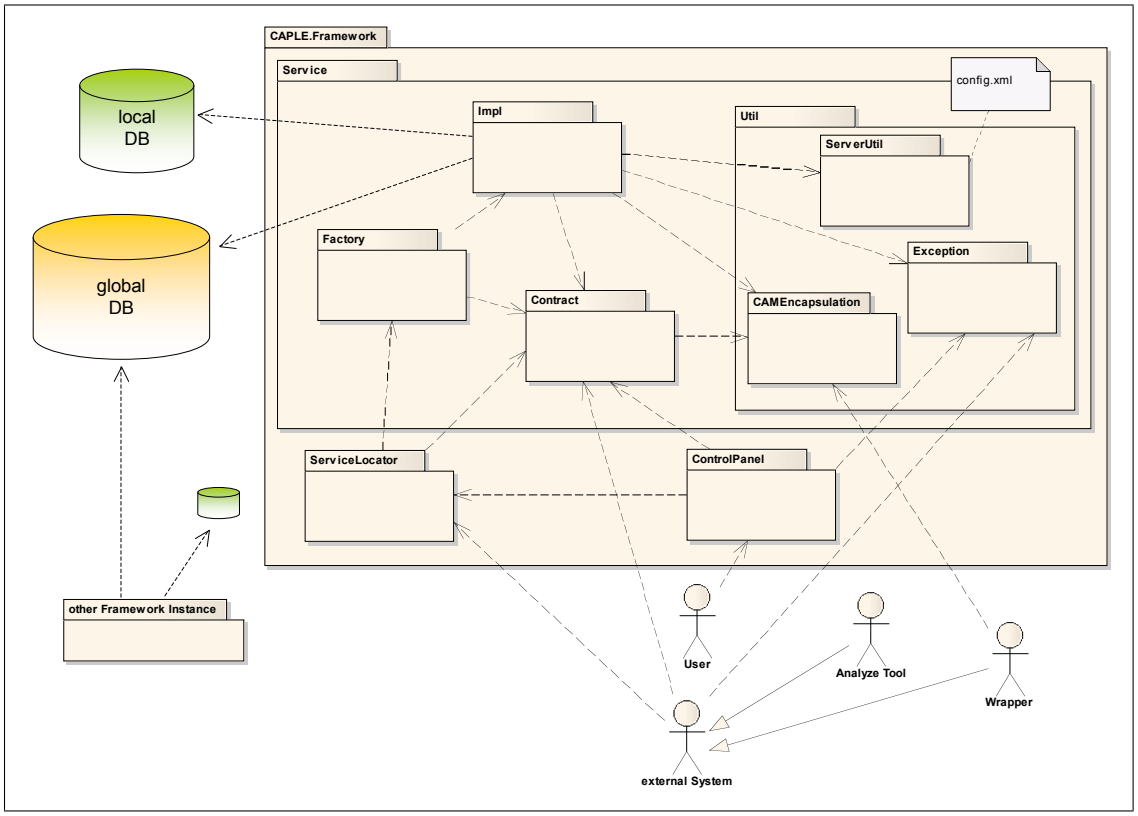


Abbildung 4.2: Architektur des CAPLE-Frameworks als Paketdiagramm

Es ist zu erkennen, dass das Framework CAM lokal und global speichern, respektive anfragen kann. Aktuell dienen für die Datenhaltung die nativen XML-Datenbanken eXist und Tamino [Bee09]. Im folgenden Abschnitt wird ein OR-Schema vorgestellt, dass dazu verwendet werden kann, das CAPLE-Framework auf traditionelle DBS umzustellen.

4.1.2 CAMS für den relationalen Zugriff

Das illustrierte CAMS aus Abbildung 4.1 beinhaltet neun Entitäten, die den Umgang von Nutzern mit Inhalten am Computer abbilden können. Aktuell werden allerdings nicht alle dieser Entitäten verwendet. Dies liegt daran, dass die *Wrapper*

im aktuellen Entwicklungsstadium des Frameworks nur spezifische Informationen liefern. Am Beispiel des *User Activity Logger Wrappers* soll herausgestellt werden, welche Entitäten des CAMS zur Zeit Anwendung finden.

Der *User Activity Logger Wrapper* ist eine Applikation, die auf Implementierungen des *User Activity Logger* der Universität Hannover basiert. Er ist in der Lage Aktivitäten des Benutzers, die vom Betriebsystem *Windows* detektiert werden, zu erfassen. Dazu gehören u.a. Informationen über den Zeitpunkt des Auftretens einer Aktion, der Applikations-, Dokumenten- und Benutzername und Zustände von Applikationen und Dokumenten [L3S07]. Abbildung 4.3 zeigt die Strukturierung dieser Informationen im Kontext des CAMS.

CAMS	**User Activity Logger Wrapper**
group:	Informationen des aktuell angemeldeten Benutzers
feed:	Name des *Wrappers*
item:	Information der aktuell verwendeten Ressource
event:	Zeitstempel der aktuellen Aktivität
action:	Aktivität des Events, z.B. *create*, *activate*, *destroy* für Applikationen und *open*, *activate*, *close* für Dokumente
entry:	Name der benutzten Applikation, z.B. *skype.exe* und spezifische Informationen der aktuellen Aktivität, z.B. *sending email message*

Abbildung 4.3: Nutzung des CAMS durch den *User Activity Logger Wrapper*

Ähnlich dem *User Activity Logger Wrapper* werden applikationsspezifische *Wrapper* wie der *Skype-* oder *Thunderbird Wrapper* im CAPLE-Projekt eingesetzt, die ebenfalls ihre gesammelten Informationen über das CAMS strukturieren. Sie liefern ihrer Anwendung entsprechend feiner granulierte Informationen, wie den Sender oder Empfänger einer Skype- respektive E-Mail-Nachricht und werden zur Zeit in der generisch ausgelegten CAMS-Entität *Entry* gespeichert. Dadurch wird suggeriert, dass das Schema einheitlich von den einzelnen Wrappern verwendet wird.

Grundsätzlich kann das Schema aus der relationalen Perspektive auf verschiedenste Weisen modelliert werden. Da das CAMS bereits Beziehungshierarchien aufweist, besteht die einfachste Möglichkeit darin, dieses Schema eins-zu-eins relational abzubilden. Das Ergebnis des Modellierungsprozesses würde im Wesentlichen dem Schema aus Abbildung 4.1 gleichen, hätte jedoch den praxisrelevanten Nachteil, dass die referenzielle Integrität gerade mit Elementen der Entität *Entry* nur schwer zu gewährleisten wäre. Ein intuitiveres Modell, welches sich dieser Problematik annimmt, zeigt das OR-Schema aus Abbildung 4.4, dargestellt als UML-Klassendiagramm. Es besteht aus den Basisklassen *User*, *Activity*, *File* und *Type*,

wobei die Entität *File* durch die zwei Klassen *Application* und *Document* erweitert wird. Eine weitere Vererbungshierarchie findet für die spezifischen Informationen der einzelnen *Wrapper* Anwendung. *Wrapper*, welche ähnliche Informationen liefern, werden durch eine OR-Superklasse repräsentiert, die wiederum die Eigenschaften von *Activity* erbt. Andere *Wrapper*, die keine Attribute teilen, leiten sich direkt von *Activity* ab (vgl. *ActivityLogger_Wrapper*).

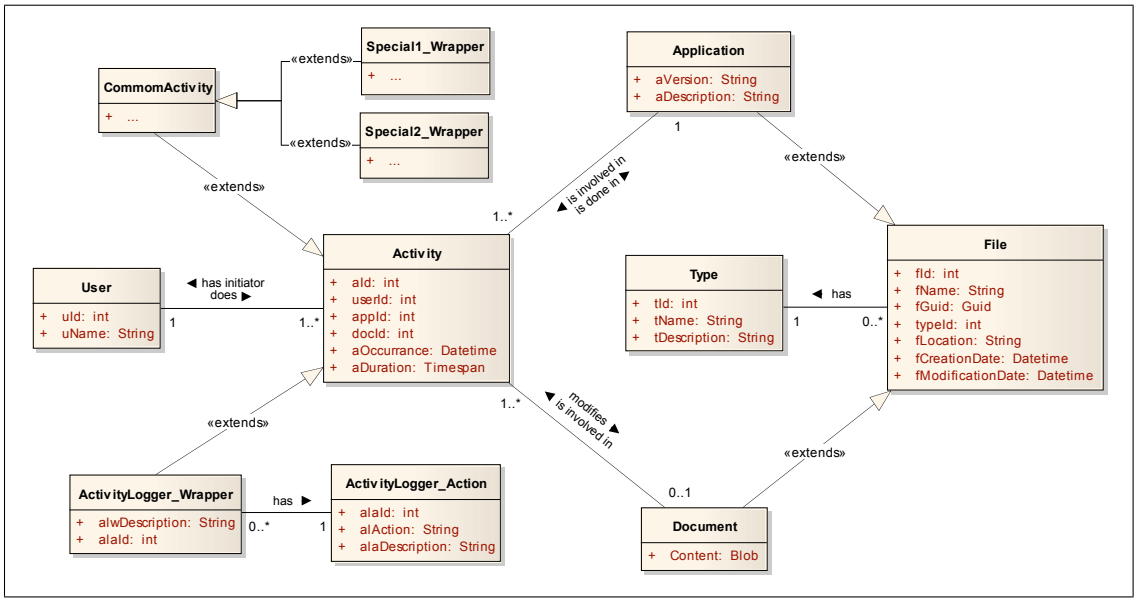


Abbildung 4.4: Abbildung des CAMS in ein OR-Modell

4.2 Weitere Vorüberlegungen und Notationen

Bevor nun die einzelnen Ideen für die Musterextraktion in den folgenden Kapiteln erläutert werden, werden an dieser Stelle noch einige Vorüberlegungen und Notationshinweise getätigt, die es für die Analyse mittels der RST zu berücksichtigen gilt.

Relation und Informationssystem

Relationen und Informationssysteme lassen sich als Tabellen darstellen (vgl. Kapitel 3.3.1). Aus Gründen der Einheitlichkeit, sei in den weiteren Ausführungen auf den strengen Formalismus von Relationen verzichtet und anstelle dessen auf die etablierten Schreibweisen des IS verwiesen (vgl. Kapitel 2.1). Existiert beispielsweise eine

m-stellige Datenbankrelation

$$R \subseteq V_{a_1} \times ... \times V_{a_m}, x_i \in R, 1 \leq i \leq n, \tag{4.1}$$

wird im Folgenden anstatt R die Notation des IS

$$\mathcal{R} = (\mathbb{U}_{\mathcal{R}} = \{x_1, ..., x_n\}, A_{\mathcal{R}} = \{a_j \mid a_j : \mathbb{U}_{\mathcal{R}} \rightarrow V_{a_j}, 1 \leq j \leq m\}), n, m \in \mathbb{N} \tag{4.2}$$

angegeben und von einer Relation respektive einem IS gesprochen.

Relationales CAMS als Informationssystem

Das relationale Modell aus Abbildung 4.4 ist die Grundlage für das weitere Vorgehen. Es besteht aus einzelnen Relationen, die miteinander in Beziehung stehen. Isoliert betrachtet lässt sich jede dieser Relationen als IS verstehen. Für eine ganzheitliche Betrachtung mittels der RST müssen alle Informationen der einzelnen Tabellen jedoch zu einem großen IS $\mathcal{A} = (\mathbb{U}_{\mathcal{A}}, A_{\mathcal{A}})$ verschmolzen werden. Der folgende algebraische Ausdruck zeigt eine Möglichkeit, dieses zu leisten:

$$\begin{aligned} \mathcal{A} = (\mathbb{U}_{\mathcal{A}}, A_{\mathcal{A}}) = \Pi_{\{a_1,a_2,...,a_n\}}(Activity \bowtie_{\phi_1} User \bowtie_{\phi_2} Application \bowtie_{\phi_3} \\ Type \bowtie_{\phi_4} Document \bowtie_{\phi_5} Activity\_Logger\_Wrapper \bowtie_{\phi_6} \\ Activity\_Logger\_Action \bowtie_{\phi_7} Type) \end{aligned} \tag{4.3}$$

mit $\{a_1, ..., a_n\} \subseteq A_{Activity} \cup A_{User} \cup A_{Application} \cup A_{Document} \cup A_{Activity\_Logger\_Wrapper} \cup A_{Activity\_Logger\_Action} \cup A_{Type}$ und den Join-Bedingungen $\phi_i, 1 \leq i \leq 7$:

$\phi_1 : userId(x) = uId(y), x \in \mathbb{U}_{Activity}, y \in \mathbb{U}_{User}$

$\phi_2 : appId(x) = fId(y), x \in \mathbb{U}_{Activity}, y \in \mathbb{U}_{Application}$

$\phi_3 : tId(x) = typeId(y), x \in \mathbb{U}_{Type}, y \in \mathbb{U}_{Application}$

$\phi_4 : docId(x) = fId(y), x \in \mathbb{U}_{Activity}, y \in \mathbb{U}_{Document}$

$\phi_5 : aId(x) = aId(y), x \in \mathbb{U}_{Activity\_Logger\_Wrapper}, y \in \mathbb{U}_{Activity}$

$\phi_6 : alaId(x) = alaId(y), x \in \mathbb{U}_{Activity\_Logger\_Wrapper}, y \in A_{Activity\_Logger\_Action}$

$\phi_7 : tId(x) = typeId(y), x \in \mathbb{U}_{Type}, y \in \mathbb{U}_{Document}.$

Ausdruck 4.3 kann durch weitere Verschmelzungsoperationen mit anderen Wrapper-Entitäten ergänzt werden, so dass letztlich wieder ein ganzheitliches IS entsteht.

Die Bedeutung der Notation von Ausdruck 4.3 lässt sich weitestgehend aus den Ausführungen in Kapitel 3.1.1 ableiten. Für die weitere Anwendung der relationalen

Algebra und dessen Erweiterungen, insbesondere in Kapitel 4.3, sei auf die gängige Fachliteratur verwiesen [RG02, BC03, MU03, KE04, Dat05, EN05, HSS07].

Behandlung von unvollständigen Informationssystemen

Das Ergebnis des algebraischen Ausdrucks 4.3 ist ein IS, welches Nullwerte enthalten kann. Solche Situationen treten insbesondere dann auf, wenn beispielsweise eine Applikation von einem Benutzer geöffnet wird, ohne dass ein Dokument in diese Aktion involviert ist. Zusätzlich entstehen Nullwerte, falls spezifische Wrapper-Informationen aus den jeweiligen Relationen in Ausdruck 4.3 eingebunden werden, welche andere Aktivitäten nicht beinhalten (vgl. Kapitel 4.1.2). Es wird im weiteren Verlauf dieser Arbeit deshalb folgende Strategie verfolgt: die zu erstellenden Modelle werden mit dieser Unsicherheit konfrontiert, wobei jeder Nullwert durch ein adäquates Symbol, wie z.B. '?' oder '−' repräsentiert wird. Das '?'-Symbol bedeutet, dass der Attributwert existent, aber temporär nicht verfügbar ist. '−' verdeutlicht, dass der Wert unbekannt ist.

Wahl der Attribute

Welche Attribute $a_1, ..., a_n, n \in \mathbb{N}$ aus dem resultierenden IS von Ausdruck 4.3 verwendet werden, wird in den jeweiligen Kapiteln explizit angegeben und diskutiert. Die Basisattribute für die Auswertungen sind in der Regel: *uName*, *fName*, *tName*, *alwDescription*, *alAction*.

Dauer von Benutzeraktionen

Die Dauer von Benutzeraktionen kann nicht ohne weiteres ermittelt werden, da ein Benutzer beispielsweise ein Dokument öffnet, sich möglicherweise aber nicht aktiv mit diesem beschäftigt. Dennoch werden in den weiteren Kapiteln drei Möglichkeiten untersucht, diese zu erfassen:

(i) Die Dauer einer Aktion wird durch den Abstand zur nächsten auftretenden Aktion ermittelt. Somit ergibt sich die Dauer aus der Differenz der Zeitstempel beider Aktionen. Sie wird im Attribut $aDuration \in A_{Activity}$ gespeichert.

(ii) Wie bei (i) wird die Dauer einer Aktion a durch Differenzbildung berechnet. Ist das Ergebnis die Zahl $n \in \mathbb{R}$, so wird n abgerundet und a wird n-mal in das zu analysierende IS projiziert.

(iii) Wie bei (i) wird die Dauer einer Aktion durch das Merkmal $aDuration \in A_{Activity}$ repräsentiert. Durch einen Diskretisierungsschritt werden die Werte von $aDuration$ auf einen symbolischen Wertebereich wie z.B. $V_{sym\_Duration} = \{kurz, mittel, lang\}$ abgebildet.

4.3 Abbildung von Rough Sets in die relationale Algebra

Die Grundlage der RST ist das IS und die Ununterscheidbarkeitsrelation. Nachdem die Generierung eines IS mit Formel 4.3 bereits beschrieben wurde, beschäftigt sich dieses Kapitel mit der Abbildung der Ununterscheidbarkeitsrelation in adäquate Ausdrücke der relationalen Algebra. Sie dient als Basis für das Berechnen der Konzeptapproximation und dem Finden von Redukten, die ebenfalls in diesem Kapitel entwickelt werden.

4.3.1 Ununterscheidbarkeit aus relationaler Perspektive

Die Ununterscheidbarkeitsrelation ist das mathematische Werkzeug der RST, um Objekte in einem IS anhand ihrer Attribute zu unterscheiden (vgl. Kapitel 2.2). Zwei Objekte sind nach Definition 2.3 äquivalent, respektive ununterscheidbar, wenn sie sich in allen zu untersuchenden Merkmalsausprägungen gleichen. Somit wird also eine Partition über dem Universum erzeugt, die im Folgenden Ausgangspunkt für die Formulierung der Ununterscheidbarkeitsrelation mit algebraischen Ausdrücken ist.

Betrachtet sei dazu $\mathcal{R} = (\mathbb{U}_{\mathcal{R}}, A_{\mathcal{R}})$, ein IS, und $IND_B, B \subseteq A_{\mathcal{R}}$, eine definierte Ununterscheidbarkeitsrelation, welche die Partitionierung $\{K_1, ..., K_n\} = IND_B, n > 1$ auf $\mathbb{U}_{\mathcal{R}}$ bezüglich B hervorruft. Jede Klasse $K_i, 1 \leq i \leq n$ für sich betrachtet enthält äquivalente Objekte, die durch einen Repräsentanten $x_i \in K_i$ eindeutig beschrieben werden kann, da $[x_i]_{IND_B} = K_i$. Fasst man jeweils einen einzigen Repräsentanten x_i aus jeder Klasse in die Menge $\{x_i \mid [x_i]_{IND_B} \in IND_B\}$ zusammen, kann diese adäquat durch

$$represent_B(\mathcal{R}) := \Pi_B(\mathcal{R}) \tag{4.4}$$

deklariert werden. $represent_B(\mathcal{R})$ erzeugt ein neues $\mathcal{A} = (\mathbb{U}_{\mathcal{A}}, B)$ mit $\mathbb{U}_{\mathcal{A}} \subseteq \mathbb{U}_{\mathcal{R}}$. Durch die Projektion auf die Attributmenge B wird bewirkt, dass alle bis auf ein

Objekt, die in B übereinstimmen für die Ergebnisrelation eliminiert werden. Das bedeutet, dass eine Duplikateneliminierung auf der Multimenge $\mathbb{U}_{\mathcal{R}}$ zu $\mathbb{U}_{\mathcal{A}}$ durchgeführt wird. Zur Vereinfachung wird im Folgenden auch $represent_B(\mathcal{R}) = \mathcal{A}$ bzw. $Ausdruck = \mathcal{A}$ geschrieben.

Der Nachteil der Ergebnisrelation aus Ausdruck 4.4 besteht darin, dass aus jeder Äquivalenzklasse zwar genau ein Repräsentant in $\mathbb{U}_{\mathcal{A}}$ enthalten ist, aber die Information über die Kardinalität der Klassen verloren geht. Formel 4.4 lässt sich durch eine Erweiterung der relationalen Algebra dahingehend modifizieren, dass die Information der Kardinalität berücksichtigt wird:

$$ind_B(\mathcal{R}) := \rho_{card \leftarrow count(*)}(\gamma_{\texttt{count(*)};B}(\mathcal{R})) \tag{4.5}$$

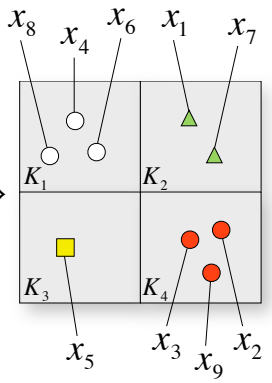
Es ist $ind_B(\mathcal{R}) = \mathcal{A} = (\mathbb{U}_{\mathcal{A}}, A_{\mathcal{A}})$ mit dem Universum $\mathbb{U}_{\mathcal{A}} \subseteq \mathbb{U}_{\mathcal{R}}$ und der Attributmenge $A_{\mathcal{A}} = B \cup \{card\}$. Die Semantik von Ausdruck 4.5 lässt sich wie folgt erklären: Zunächst wird durch die Gruppierungsfunktion γ die Relation $\mathcal{R}$ anhand von B in disjunkte Teilmengen zerlegt. Dabei erwirkt γ, dass nur solche Objekte in eine Gruppe fallen, die in allen Attributen von B übereinstimmen (vgl. Äquivalenzklasse). Nach der Gruppierungsphase wird die Aggregatfunktion `count(*)`, welche die Kardinalität jeder Gruppe ermittelt, auf jede erzeugte Gruppe angewandt. Das Ergebnis wird im letzten Schritt im neuen Attribut $card \in A_{\mathcal{A}}$ für jede Gruppe hinterlegt.

Beispiel 4.1. Illustration der Ununterscheidbarkeitsabbildung
Sei $\mathcal{R} = (\mathbb{U}_{\mathcal{R}}, A_{\mathcal{R}})$ mit dem Universum $\mathbb{U}_{\mathcal{R}} = \{x_1, ..., x_9\}$ und der Attributmenge $A_{\mathcal{R}} = \{Form, Farbe\}$ gegeben durch die linke Tabelle in Abbildung 4.5.

Es ist ersichtlich, dass die Ununterscheidbarkeitsrelation $IND_{A_{\mathcal{R}}}$ die Partition $\{K_1, K_2, K_3, K_4\} = IND_{A_{\mathcal{R}}}$ erzeugt mit den Äquivalenzklassen $K_1 = \{x_4, x_6, x_8\}$, $K_2 = \{x_1, x_7\}$,

$\mathcal{R}$		
-	*Form*	*Farbe*
x_1	Dreieck	grün
x_2	Kreis	rot
x_3	Kreis	rot
x_4	Kreis	weiß
x_5	Quadrat	gelb
x_6	Kreis	weiß
x_7	Dreieck	grün
x_8	Kreis	weiß
x_9	Kreis	rot

⇔



⇔

$ind_{A_{\mathcal{R}}}(\mathcal{R})$		
card	*Form*	*Farbe*
2	Dreieck	grün
3	Kreis	rot
3	Kreis	weiß
1	Quadrat	gelb

Abbildung 4.5: Ununterscheidbarkeit mittels relationaler Algebra

$K_3 = \{x_5\}, K_4 = \{x_2, x_3, x_9\}$. Durch Anwendung von Ausdruck 4.5 ist eine Abbildung der Ununterscheidbarkeitsrelation mit Datenbankoperationen möglich. Die rechte Tabelle aus Abbildung 4.5 zeigt dazu die Ergebnisrelation. Jede Zeile repräsentiert eine Äquivalenzklasse $K_i, 1 \leq i \leq 4$, wobei die Kardinalität $|K_i|$ der Klassen jeweils über das Merkmal *card* beschrieben wird.

4.3.2 Berechnung der Konzeptapproximation

Die RST definiert aus Sicht der Ununterscheidbarkeitsrelation IND_B für die Darstellung einer Menge X eines IS $\mathcal{R} = (\mathbb{U}_\mathcal{R}, A_\mathcal{R}), X \subseteq \mathbb{U}_\mathcal{R}, B \subseteq A_\mathcal{R}$ zwei weitere Mengen $\underline{X}_B, \overline{X}_B$, welche X approximieren. Die B-untere Approximation $\underline{X}_B$ enthält dabei die Äquivalenzklassen aus IND_B, welche vollständig in X enthalten sind. Mit anderen Worten bedeutet dies, dass nur solche Klassen aus $\mathbb{U}_\mathcal{R}$ in die B-untere Approximation fallen, deren Kardinalität mit den äquivalenten Klassen aus X übereinstimmen. Auf der anderen Seite besteht die B-obere Approximation $\overline{X}_B$ aus den Klassen in $\mathbb{U}_\mathcal{R}$, die einen nicht-leeren Schnitt mit X haben. Anders ausgedrückt enthält die B-obere Approximation also die Äquivalenzklassen, welche aus X hervorgehen und zusätzlich Objekte außerhalb von X, die äquivalent zu den Klassen in X sind (vgl. Kapitel 2.3). Diese Eigenschaften werden im Folgenden genutzt, um die Konzeptapproximation relational zu beschreiben.

Rein formal lässt sich die Konzeptapproximation aus der relationalen Perspektive wie folgt beschreiben: seien die zwei IS $\mathcal{R} = (\mathbb{U}_\mathcal{R}, A_\mathcal{R})$ und $\mathcal{R}' = (\mathbb{U}_{\mathcal{R}'}, A_{\mathcal{R}'})$ gegeben mit den Eigenschaften: $\mathbb{U}_{\mathcal{R}'} \subseteq \mathbb{U}_\mathcal{R}, A_{\mathcal{R}'} = A_\mathcal{R}, B \subseteq A_\mathcal{R}, A_{\mathcal{R}'}$. $\mathcal{R}'$ symbolisiert dabei das zu approximierende Konzept in einem eigenständigen IS.

B-untere Approximation

Die B-untere Approximation des Konzepts lässt sich durch folgenden mengentheoretischen Schnitt angeben:

$$lower_B(\mathcal{R}, \mathcal{R}') := ind_B(\mathcal{R}) \cap ind_B(\mathcal{R}') \tag{4.6}$$

Nach oben stehender Argumentation zur Ermittlung von $\underline{X}_B$ berechnet $lower_B(\mathcal{R}, \mathcal{R}')$ zunächst die Äquivalenzklassen mit den jeweiligen Kardinalitäten für $\mathcal{R}$ und $\mathcal{R}'$. Die Ergebnisrelation ergibt sich dann genau aus den Klassen, die gleichermaßen in beiden IS enthalten sind.

B-obere Approximation

Die B-obere Approximation von $\mathcal{R}'$ lässt sich wiederum durch den Ausdruck

$$upper_B(\mathcal{R}, \mathcal{R}') := ind_B(represent_B(\mathcal{R}') \bowtie_\phi \mathcal{R}) \tag{4.7}$$

angeben mit der Join-Bedingung $\phi : a_1(x) = a_1(y) \wedge ... \wedge a_n(x) = a_n(y), a_i \in B, 1 \leq i \leq n \in \mathbb{N}, x \in \mathbb{U}_\mathcal{R}, y \in \mathbb{U}_{\mathcal{R}'}$. In einem ersten Schritt werden Repräsentanten $x \in [x]_B$ aus dem Konzept extrahiert. Sie spiegeln die Äquivalenzklassen der B-oberen Approximation wider. Damit die genaue Anzahl der Objekte in den einzelnen Klassen ermittelt werden kann, wird das Ergebnis in einem zweiten Schritt mit $\mathcal{R}$ durch $\bowtie_\phi$ verschmolzen. Letzlich kann dann Ausdruck 4.5 angewandt werden, der die korrekte Kardinalität der Klassen berechnet.

B-Grenzregion

Die B-Grenzregion ergibt sich analog zur traditionellen Beschreibung aus Definition 2.5 durch Differenzbildung von B-oberer und B-unterer Approximation:

$$boundary_B(\mathcal{R}, \mathcal{R}') := upper_B(\mathcal{R}, \mathcal{R}') - lower_B(\mathcal{R}, \mathcal{R}') \tag{4.8}$$

B-Außenregion

Die B-Außenregion lässt sich ähnlich wie $upper_B(\mathcal{R}, \mathcal{R}')$ ausdrücken. Zunächst wird durch die mengentheoretische Differenz nach Repräsentanten von Äquivalenzklassen gesucht, die sich in $\mathbb{U}_\mathcal{R} - \mathbb{U}_{\mathcal{R}'}$ befinden. Dann wird das Pseudo-Ergebnis für die B-Außenregion mit der Basisrelation über die oben erläuterte Join-Bedingung ϕ verknüpft, um letztlich die tatsächlichen Kardinalitäten der Klassen zu berechnen:

$$outside_B(\mathcal{R}, \mathcal{R}') := ind_B((represent_B(\mathcal{R}) - represent_B(\mathcal{R}')) \bowtie_\phi \mathcal{R}) \tag{4.9}$$

Beispiel 4.2. Konzeptapproximation mit Datenbankoperationen
Seien die IS $\mathcal{R}$ und $\mathcal{R}'$ gegeben durch Abbildung 4.6. Es ist leicht zu erkennen, dass $\mathbb{U}_{\mathcal{R}'} \subset \mathbb{U}_\mathcal{R}$ und $A_{\mathcal{R}'} = A_\mathcal{R}$. Für $B = A_{\mathcal{R}'} = A_\mathcal{R}$ induziert $IND_\mathcal{R}(B)$ nach dem traditioniellen Rough-Set-Modell die Partition[1]:

$$IND_\mathcal{R}(B) = \{\underbrace{\{x_1, x_7\}}_{K_2}, \underbrace{\{x_2, x_3, x_9\}}_{K_4}, \underbrace{\{x_4, x_6, x_8\}}_{K_1}, \underbrace{\{x_5\}}_{K_3}\}$$

[1] Die Abzählung $i, 1 \leq i \leq 4$ der einzelnen Äquivalenzklasen K_i von $IND_\mathcal{R}(B)$ ist dabei an die Illustration von Beispiel 4.1 angelehnt.

$\mathcal{A}$		
-	*Form*	*Farbe*
x_1	Dreieck	grün
x_2	Kreis	rot
x_3	Kreis	rot
x_4	Kreis	weiß
x_5	Quadrat	gelb
x_6	Kreis	weiß
x_7	Dreieck	grün
x_8	Kreis	weiß
x_9	Kreis	rot

$\mathcal{A}'$		
-	*Form*	*Farbe*
x_2	Kreis	rot
x_3	Kreis	rot
x_4	Kreis	weiß
x_5	Quadrat	gelb
x_6	Kreis	weiß
x_8	Kreis	weiß

Abbildung 4.6: Informationssysteme zur Berechnung der Konzeptapproximation

Da sich $\mathcal{R}'$ als ein eigenständiges IS darstellt, ergibt sich rein formal die Partition:

$$IND_{\mathcal{R}'}(B) = \{\underbrace{\{x_2, x_3\}}_{K_4'}, \underbrace{\{x_4, x_6, x_8\}}_{K_1'}, \underbrace{\{x_5\}}_{K_3'}\}$$

Ist nun $\mathbb{U}_{\mathcal{R}'}$ das zu approximierende Konzept, dann ist $K_1' = K_1, K_3' = K_3$. K_1 und K_3 sind vollständig im Konzept enthalten und bilden somit die B-untere Approximation. Weiter gilt $K_4' \subset K_4$. Das bedeutet, dass sich mindestens ein Objekt aus K_4 außerhalb des Konzepts befindet. K_4 fällt also in die B-obere Approximation. K_2 bildet die B-Außenregion. Die Approximation mittels relationaler Algebra greift die zuvor erläuterten Ideen auf. Durch die Abbildungen 4.7, 4.8, 4.9, 4.10 soll dies verdeutlicht werden:

$ind_B(\mathcal{A})$		
card	*Form*	*Farbe*
2	Dreieck	grün
3	Kreis	rot
3	Kreis	weiß
1	Quadrat	gelb

$\cap$

$ind_B(\mathcal{A}')$		
card	*Form*	*Farbe*
2	Kreis	rot
3	Kreis	weiß
1	Quadrat	gelb

$=$

$lower_B(\mathcal{A}, \mathcal{A}')$		
card	*Form*	*Farbe*
3	Kreis	weiß
1	Quadrat	gelb

Abbildung 4.7: Berechnung der B-unteren Approximation mit relationaler Algebra

$represent_B(\mathcal{A}')$	
Form	*Farbe*
Kreis	rot
Kreis	weiß
Quadrat	gelb

$\bowtie_\phi$

$\mathcal{A}$	
Form	*Farbe*
Dreieck	grün
Kreis	rot
Kreis	rot
Kreis	weiß
Quadrat	gelb
Kreis	weiß
Dreieck	grün
Kreis	weiß
Kreis	rot

$=$

$\mathcal{T}$	
Form	*Farbe*
Kreis	rot
Kreis	rot
Kreis	weiß
Quadrat	gelb
Kreis	weiß
Kreis	weiß
Kreis	rot

$ind_B(\mathcal{T}) = upper_B(\mathcal{A}, \mathcal{A}')$		
card	*Form*	*Farbe*
3	Kreis	rot
3	Kreis	weiß
1	Quadrat	gelb

Abbildung 4.8: Berechnung der B-oberen Approximation mit relationaler Algebra

$upper_B(\mathcal{A}, \mathcal{A}')$		
card	*Form*	*Farbe*
3	Kreis	rot
3	Kreis	weiß
1	Quadrat	gelb

$-$

$lower_B(\mathcal{A}, \mathcal{A}')$		
card	*Form*	*Farbe*
3	Kreis	weiß
1	Quadrat	gelb

$=$

$boundary_B(\mathcal{A}, \mathcal{A}')$		
card	*Form*	*Farbe*
3	Kreis	rot

Abbildung 4.9: Berechnung der B-Grenzregion mit relationaler Algebra

$represent_B(\mathcal{A})$	
Form	*Farbe*
Dreieck	grün
Kreis	rot
Kreis	weiß
Quadrat	gelb

$-$

$represent_B(\mathcal{A}')$	
Form	*Farbe*
Kreis	rot
Kreis	weiß
Quadrat	gelb

$=$

$\mathcal{S}$	
Form	*Farbe*
Dreieck	grün

$\mathcal{S} \bowtie_\phi \mathcal{A}$	
Form	*Farbe*
Dreieck	grün
Dreieck	grün

$ind_B(\mathcal{S} \bowtie_\phi \mathcal{A}) = outside_B(\mathcal{A}, \mathcal{A}')$		
card	*Form*	*Farbe*
2	Dreieck	grün

Abbildung 4.10: Berechnung der B-Außenregion mit relationaler Algebra

4.3.3 Suche nach Redukten

Das Finden eines minimalen Reduktes in einem IS ist im Allgemeinen NP-schwer (vgl. Kapitel 2.4). Trotz dieses Umstandes ist analog zur Transformationsphase im KDD-Prozess eine Datenreduktion auf Attributebene für viele Anwendungsgebiete unumgänglich, um den massiven Bestand an Informationen zu verarbeiten. Einem solchen Szenario unterliegt auch teilweise das CAPLE-Projekt. Hier werden die Aktionen des Nutzers kontinuierlich durch die *Wrapper* erfasst und liegen in großer Anzahl zur Auswertung vor.

Etablierte Methoden der Merkmalsreduktion mit Hilfe der RST verfolgen für das Extrahieren von Redukten die Strategien der *Forward Selection* oder der *Backward Elimination* [ZY04]. Eine weitere Möglichkeit wird von Hu et al. in [HLH03] beschrieben, bei der ausgehend vom *Core* nach Redukten gesucht wird (vgl. Kapitel 3.3.1). Dieser Abschnitt lehnt sich an die Ideen von Hu et al. in Kombination mit Definition 2.14 an, wobei zunächst mit relatationaler Algebra nach dem *Core* eines gegebenen IS gesucht wird. Im finalen Schritt werden dann weitere Attribute dahingehend überprüft, ob sie zusammen mit dem *Core* ein Redukt bilden. Der Algorithmus *core_search* aus Abbildung 4.11 beschreibt zunächst das Vorgehen zur Berechnung des *Core*.

Algorithmus $core\_search(\mathcal{R}, B \subseteq A_{\mathcal{R}})$:

$C = \emptyset$

foreach *Attribute* $a \in B$ **do**

  if $|represent_B(\mathcal{R})| \neq |represent_{B-\{a\}}(\mathcal{R})|$ **then**

    $C = C \cup \{a\}$

  end

end

return C

Abbildung 4.11: Algorithmus zur Bestimmung des *Core*

Die Signatur des Algorithmus besteht aus einer gegebenen Relation $\mathcal{R}$ und der zu untersuchenden Attributmenge $B \subseteq A_{\mathcal{R}}$. Aufgabe von *core_search* ist es nun jedes Attribut $a \in B$ darauf hin zu überprüfen, ob sein Entfernen einen Effekt auf die Anzahl der Äquivalenzklassen hat. Lässt sich a Entfernen ohne das sich die Struktur der Daten in $\mathcal{R}$ verändert, kann über a ausgesagt werden, dass a für eine Klassifikation nicht benötigt wird und verworfen werden kann. Alle anderen Attribute,

dessen Löschung die Struktur der Daten aus Sicht der Ununterscheidbarkeitsrelation verändert, sind Element des *Core*[2]. Der Algorithmus *reduct_search* aus Abbildung 4.12 basiert auf der resultierenden Menge von *core_search*. Im speziellen Falle, dass $B = C$ gilt, kann sofort geschlussfolgert werden, dass C das minimale Redukt aus $\mathcal{R}$ symbolisiert.

Algorithmus $reduct\_search(\mathcal{R}, B \subseteq A_{\mathcal{R}})$:

$C = core\_search(\mathcal{R}, B)$

if $B - C = \emptyset$ **then**

  return C

end

foreach *Attribute* $A \in \mathcal{P}(B - C) - \emptyset$ **do**

  if $|represent_B(\mathcal{R})| = |represent_{C \cup A}(\mathcal{R})| \wedge$
  $\forall a \in A : |represent_B(\mathcal{R})| \neq |represent_{C \cup A - \{a\}}(\mathcal{R})|$ **then**

    return $C \cup A$

  end

end

Abbildung 4.12: Algorithmus zur Bestimmung eines Redukt

reduct_search ist ein naiver Algorithmus. Er basiert im Wesentlichen auf der Idee, die verbleibenden Attribute $A \subseteq B - C$ daraufhin zu überprüfen, ob sie zusammen mit dem *Core* die Eigenschaft eines Redukts erfüllen. Im worst-case muss *reduct_search* $2^{|B-C|} - 1$ Attributkombinationen auf die Redukteigenschaft aus Definition 2.14 analysieren, was im Allgemeinen nur für eine moderate Anzahl an Attributen $|B-C|$, respektive für ein großes $|C|$ im Vergleich zu $|B|$ anwendbar ist. Für die Auswertung von CAM sei allerdings erwähnt, dass die Attributmenge zum aktuellen Zeitpunkt noch sehr klein ist. Dennoch ist absehbar, dass die Implementierung neuer *Wrapper* in weiteren Forschungsarbeiten zu einem Anwachsen der Attribute führt.

Beispiel 4.3. Finden eines Redukts mit *reduct_search*
Sei das IS $\mathcal{A} = (\mathbb{U}_{\mathcal{A}}, A_{\mathcal{A}})$ aus Abbildung 4.13 gegeben mit $\mathbb{U}_{\mathcal{A}} = \{x_1, ..., x_8\}$ und $A_{\mathcal{A}} = \{a, ..., f\}$. Damit ein Redukt $R \subseteq A_{\mathcal{A}}$ aus $\mathcal{A}$ extrahiert werden kann, wird zunächst der *Core* mit Hilfe des vorgeschlagenen Algorithmus *core_search* ermittelt. Dazu wird jedes Attribut $y \in A_{\mathcal{A}}$ auf die Bedingung $|represent_{A_{\mathcal{A}}}(\mathcal{A})| \neq$

[2] vgl. Beweisführung von Hu et al. in [HLH03]

-	a	b	c	d	e	f
x_1	1	3	1	6	1	1
x_2	1	3	1	6	1	1
x_3	1	4	1	6	1	1
x_4	1	4	1	5	2	1
x_5	2	3	1	5	2	1
x_6	2	3	1	5	2	1
x_7	2	4	1	6	1	1
x_8	2	3	1	5	2	1

Abbildung 4.13: Informationssystem zum Finden von Redukten

$|represent_{A_{\mathcal{A}}-\{y\}}(\mathcal{A})|$ überprüft. Daraus ergibt sich dann für die Attribute:

$$\begin{array}{lllll}
a: & |represent_{A_{\mathcal{A}}}(\mathcal{A})| = 5 & \neq & 4 = |represent_{A_{\mathcal{A}}-\{a\}}(\mathcal{A})| \\
b: & |represent_{A_{\mathcal{A}}}(\mathcal{A})| = 5 & \neq & 4 = |represent_{A_{\mathcal{A}}-\{b\}}(\mathcal{A})| \\
c: & |represent_{A_{\mathcal{A}}}(\mathcal{A})| = 5 & = & 5 = |represent_{A_{\mathcal{A}}-\{c\}}(\mathcal{A})| \\
d: & |represent_{A_{\mathcal{A}}}(\mathcal{A})| = 5 & = & 5 = |represent_{A_{\mathcal{A}}-\{d\}}(\mathcal{A})| \\
e: & |represent_{A_{\mathcal{A}}}(\mathcal{A})| = 5 & = & 5 = |represent_{A_{\mathcal{A}}-\{e\}}(\mathcal{A})| \\
f: & |represent_{A_{\mathcal{A}}}(\mathcal{A})| = 5 & = & 5 = |represent_{A_{\mathcal{A}}-\{f\}}(\mathcal{A})|
\end{array}$$

Der *Core* besteht somit aus den Attributen $C = \{a, b\}$. Es bleibt die Attributmengen, die aus $A_{\mathcal{A}} - C$ resultieren auf die Redukteigenschaften zu untersuchen. Die Anzahl der möglichen Attributkombinationen ergibt sich aus $|\mathcal{P}(A_{\mathcal{A}} - C) - \emptyset| \ll |\mathcal{P}(A_{\mathcal{A}}) - \emptyset|$. Beginnend bei den Teilmengen mit der Kardinalität 1, lässt sich durch *reduct_search* algorithmisch bestimmen, dass die Menge $R = C \cup \{d\}$ ein Redukt formt. Es gilt nämlich zum einen für R:

$$|represent_{A_{\mathcal{A}}}(\mathcal{A})| = |represent_R(\mathcal{A})| \tag{4.10}$$

und zum anderen lässt sich R nicht minimieren:

$$|represent_{A_{\mathcal{A}}}(\mathcal{A})| \neq |represent_{R-\{a\}}(\mathcal{A})|, \forall a \in R \tag{4.11}$$

4.4 Klassifikator auf Basis der Ununterscheidbarkeit

Dieser Abschnitt beschäftigt sich mit der Ununterscheidbarkeit als Instrument für die Klassifikation von CAM. Dazu wird zunächst mathematisch ein unscharfer Klassifikator formuliert, der unter Zuhilfenahme von externem Expertenwissen eine Hypothese auf den Daten erzeugt. Weiter wird ein Fehlermaß für den Klassifikator diskutiert, welches im Folgenden zur objektiven Evaluierung von generierten Hypothesen dienen soll. Letztlich wird in diesem Abschnitt anhand der parametrisierbaren Ununterscheidbarkeitsrelation herausgestellt, für welche Bereiche in der Benutzeranalyse des CAPLE-Projekts der formulierte Klassifikator anwendbar ist.

4.4.1 Formulierung des Klassifikators

Die Beschreibung der Ununterscheidbarkeitsrelation durch Erweiterungen der relationalen Algebra wurde bereits in Kapitel 4.3.1 erörtert. Dabei wird auf einem gegebenen IS $\mathcal{R} = (\mathbb{U}_\mathcal{R}, A_\mathcal{R})$ jede real induzierte Äquivalenzklasse $K_i \in IND_\mathcal{R}(A_\mathcal{R}), 1 \leq i \leq n \in \mathbb{N}$ durch ein Tupel $t_i \in \mathbb{U}_\mathcal{A}$ des IS $\mathcal{A} = (\mathbb{U}_\mathcal{A}, A_\mathcal{A})$ repräsentiert, welches aus der Anwendung von $ind_{A_\mathcal{R}}(\mathcal{R})$ resultiert. Die Kardinalität der Klassen $|K_i|$ wird dabei durch $card(t_i), card \in A_\mathcal{A}$ notiert (vgl. Ausdruck 4.5 und Beispiel 4.1). Grafisch kann das Ergebnis des deklarierten Ausdrucks $ind_{A_\mathcal{R}}(\mathcal{R})$ durch das folgende Histogramm schematisch illustriert werden:

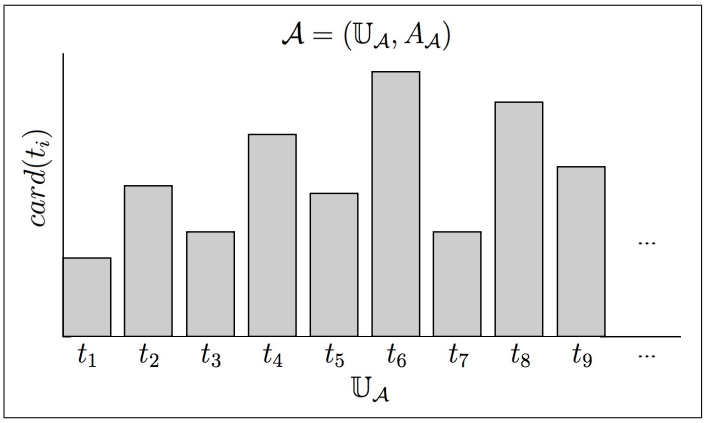


Abbildung 4.14: Ergebnisrelation von $ind_{A_\mathcal{R}}(\mathcal{R})$ als schematisches Histogramm

Diese Abbildung zeigt die absolute Häufigkeitsverteilung des gegebenen Datenraums aus $\mathcal{R}$ bezüglich der Attributmenge $A_\mathcal{R}$. Im Kontext des CAPLE-Projekts handelt

es sich um auftretende Aktivitäten von Benutzern im Umgang mit digitalen Inhalten in unterschiedlichen Anwendungen. Im speziellen Fall, dass die Daten aus $\mathcal{R}$ für einen bestimmten Benutzer in einem definierten Zeitintervall selektiert[3] vorliegen, können mittels $ind_{A_\mathcal{R}}(\mathcal{R})$ häufiger und weniger häufig auftretende Aktionen ermittelt werden. Tritt eine Aktion häufiger auf, kann daraus geschlossen werden, dass dieser Benutzer sich intensiver mit einem gewissen Sachverhalt auseinandersetzt als mit anderen. Wäre nun bekannt, ab welcher Schwelle bestimmte Aktivitäten im Interesse des Benutzers liegen und ab welcher Schwelle Aktionen definitiv aus dem Fokus fallen, lässt diese Überlegung eine grobe Hypothese für das Verhalten des Benutzers zu.

Sei $\mathcal{R} = ind_{A_\mathcal{T}}(\mathcal{T})$ mit $\mathbb{U}_\mathcal{R} \subseteq \mathbb{U}_\mathcal{T}, A_\mathcal{R} = \{card\} \cup A_\mathcal{T}$ das resultierende IS des algebraischen Ausdrucks 4.5, der auf das gegebenen IS $\mathcal{T} = (\mathbb{U}_\mathcal{T}, A_\mathcal{T})$ mit den Aktivitäten $\mathbb{U}_\mathcal{T} = \{a_1, ..., a_n\}, n \in \mathbb{N}$ angewandt wird. Mit Hilfe der zwei Schwellwerte $\alpha, \beta \in \mathbb{N}, \alpha > \beta$ lässt sich der Rough-Set-Klassifikator $h : \mathbb{U}_\mathcal{R} \times \mathbb{N}^2 \rightarrow \{1, ?, 0\}$ definieren durch:

$$h^\alpha_\beta(x) = \begin{cases} 1, & \text{falls } card(x) > \alpha \\ ?, & \text{falls } \beta \leq card(x) \leq \alpha \\ 0, & \text{sonst} \end{cases} \tag{4.12}$$

Es ist ersichtlich, dass anhand von h, der Ununterscheidbarkeitsrelation und dem Expertenwissen α und β, die Objekte $x \in \mathbb{U}_\mathcal{R}$ in die vordefinierten Klassen

$$\begin{aligned} K_{h=1} &= \{x \in \mathbb{U}_\mathcal{R} \mid h^\alpha_\beta(x) = 1\} \\ K_{h=?} &= \{x \in \mathbb{U}_\mathcal{R} \mid h^\alpha_\beta(x) = ?\} \\ K_{h=0} &= \{x \in \mathbb{U}_\mathcal{R} \mid h^\alpha_\beta(x) = 0\} \end{aligned} \tag{4.13}$$

eingeordnet werden können.

Im Bezug auf die Auswertung eines Benutzers können die Klassen folgendermaßen interpretiert werden: Aktivitäten in

$K_{h=1}$: fallen definitiv in das Interesse des Benutzers

$K_{h=?}$: können, müssen aber nicht im Interesse des Benutzers liegen

$K_{h=0}$: fallen definitiv aus dem Interesse des Benutzers

In Anlehnung an die RST lässt sich analog zu den erzeugbaren Klassen aus h die Approximation für das unbekannte Zielkonzept formulieren. Die B-untere Approxi-

[3] vgl. Selektionphase des KDD aus Kapitel 3.2.1

mation besteht aus der Klasse $K_{h=1}$. Die B-obere Approximation lässt sich aus der mengentheoretischen Vereinigung $K_{h=1} \cup K_{h=?}$ bilden und die B-Außenregion ergibt sich aus $K_{h=0}$.

Beispiel 4.4. Funktionsweise des Rough-Set-Klassifikators
Gegeben sei das IS $\mathcal{T}$, welches vorselektierte Aktivitäten eines Benutzers durch die Aufzeichnungen des *User Activity Logger Wrappers* bereit stellt. Durch Anwendung des algebraischen Ausdrucks $ind_{A_{\mathcal{T}}}(\mathcal{T})$ entsteht das IS $\mathcal{R}$ aus Abbildung 4.15[4]. Anhand der vorliegenden Daten in $\mathcal{R}$ lässt sich subjektiv bestimmen, dass sich der Benutzer *Max* zwischen den Tageszeiten *morning* und *afternoon* mit den Dokumenten *Worksheet* und *Todo* beschäftigt. Zusätzlich hat er in diesem Zeitraum mit der Applikation *Firefox* gearbeitet und ausschließlich zur Tageszeit *midday* Berechnungen mit *Calc* durchgeführt. Das Öffnen und Schließen von Applikationen ist ebenfalls in $\mathcal{R}$ dargestellt, mit dem Unterschied, dass diese Aktionen nicht wesentlicher Bestandteil seiner Arbeit sind.

-	*card*	*User*	*Application*	*Document*	*Type*	*Action*	*Time_of_Day*
x_1	4	Max	Winword	-	-	create	morning
x_2	24	Max	Winword	Worksheet	*.doc	activate	morning
x_3	11	Max	Winword	Worksheet	*.doc	activate	afternoon
x_4	17	Max	Wordpad	Todo	*.txt	activate	midday
x_5	3	Max	Wordpad	Todo	*.txt	activate	afternoon
x_6	7	Max	Firefox	-	-	create	morning
x_7	5	Max	Firefox	-	-	destroy	afternoon
x_8	12	Max	Firefox	?	?	activate	morning
x_9	19	Max	Firefox	?	?	activate	midday
x_{10}	6	Max	Firefox	?	?	activate	afternoon
x_{11}	9	Max	Calc	-	-	activate	midday

Abbildung 4.15: Beispielhafte Beobachtungen als Informationssystem

Ist nun durch einen Experten bekannt, dass $\alpha = 11$ und $\beta = 8$, lässt sich der Klassifikator h auf jedes Objekt in $\mathcal{R}$ anwenden. Abbildung 4.16 stellt die Klassifikation der Äquivalenzklassen $x_i, 1 \leq i \leq 11$ grafisch dar. Die erzeugbare Rough-Set-Klassifikation ergibt sich aus den Mengen:

$$\begin{aligned} K_{h=1} &= \{x_2, x_4, x_8, x_9\} \\ K_{h=?} &= \{x_3, x_{11}\} \\ K_{h=0} &= \{x_1, x_5, x_6, x_7, x_{10}\} \end{aligned}$$

[4] Durch geeignete Verwendung der Umbenennungsfunktion ρ sind die Attributnamen für den Leser vereinfacht worden.

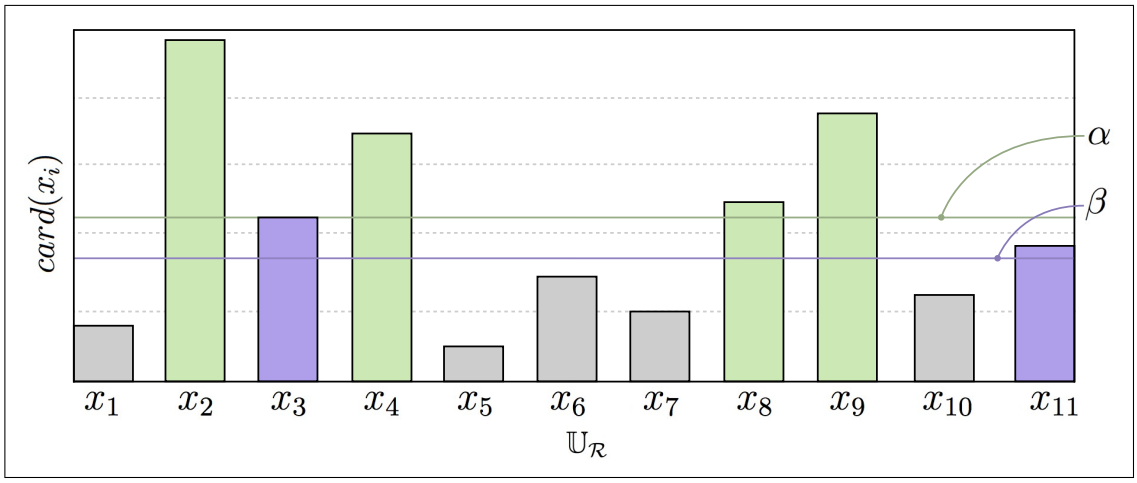


Abbildung 4.16: Illustration von h angewandt auf $\mathcal{R}$ mittels $ind_{A_{\mathcal{T}}}(\mathcal{T})$

Das bedeutet, dass anhand des Expertenwissens α und β die Aktivitäten $K_{h=1}$ vollständig im Zielkonzept enthalten sind, während die Aktivitäten $K_{h=?}$ nur partiell im Konzept liegen. Bei $K_{h=0}$ handelt es sich um Aktivitäten, welche zwar vom Benutzer *Max* durchgeführt worden sind, aber laut dem Experten nicht zum Zielkonzept gehören.

Anhand des Musters von h kann interpretiert werden, dass *Max* sich zur Tageszeit *morning* intensiv mit dem Dokument *Worksheet* und teilweise mit Inhalten von Webseiten beschäftigt. In der Zeit *midday* hingegen arbeitet er über die Applikation *Wordpad* am Dokument *Todo* und fokussiert sein Interesse auf Webseiten. Ebenfalls setzt *Max* sich über das Programm *Calc* mit Berechnungen auseinander, die möglicherweise in Beziehung mit den zuvor beschrieben Tätigkeiten stehen. Gleiches gilt für die Bearbeitung des Dokuments *Worksheet* zur Tageszeit *afternoon*. Alle weiteren Aktivitäten sind aufgrund des Vorwissens nicht Bestandteil des Konzept.

4.4.2 Güte der Klassifikation

Im Kontext des maschinellen Lernens lässt sich der in Kapitel 4.4.1 formulierte Klassifikator h auf seine Güte evaluieren. Das bedeutet, dass h darauf zu prüfen ist, ob durch die Klassifikation auch ungesehene Aktivitäten korrekt klassifiziert werden können. Das ML bietet hierfür verschiedene Möglichkeiten dies durchzuführen. Hier wird folgender Ansatz verfolgt: zunächst wird in einem ersten Schritt aus der zugrunde liegenden Datenmenge $\mathcal{A} = (\mathbb{U}_{\mathcal{A}}, A_{\mathcal{A}})$ eine Sample-Menge $\mathcal{S} = (\mathbb{U}_{\mathcal{S}}, A_{\mathcal{A}})$ entnommen mit $\mathbb{U}_{\mathcal{S}} \subseteq \mathbb{U}_{\mathcal{A}}$. Soll nun $\mathcal{S}$ auf Fehlklassifizierungen validiert werden, lässt sich $\mathcal{S}' = ind_{A_{\mathcal{A}}}(\mathcal{S})$ als Ausgangspunkt für die Erzeugung einer Hypothese h

berechnen. Den Fehler, den die Hypothese trotz des Expertenwissen α und β auf $\mathcal{S}'$ produziert, lässt sich dann durch folgende Menge bestimmen:

$$\begin{aligned} errSet_{\mathcal{S}'}(h) = \{x \in \mathbb{U}_{\mathcal{S}'} \mid \quad & (h^{\alpha}_{\beta}(x) = 0 \wedge |[x]_{IND_{\mathcal{A}}(A_{\mathcal{A}})}| \geq \beta) \vee \\ & (h^{\alpha}_{\beta}(x) = ? \wedge |[x]_{IND_{\mathcal{A}}(A_{\mathcal{A}})}| > \alpha) \ \} \end{aligned} \tag{4.14}$$

Die Fehlerrate, also der relative Fehler, den h auf $\mathcal{S}'$ macht, lässt sich durch den nachstehenden Quotienten ausdrücken:

$$errRate_{\mathcal{S}'}(h) = \frac{|errSet_{\mathcal{S}'}(h^{\alpha}_{\beta})|}{|\mathbb{U}_{\mathcal{S}'}|} \tag{4.15}$$

Es ist ersichtlich, dass der Fehler, den h auf $\mathcal{S}'$ generiert, stark von der Verteilung der Daten in $\mathcal{A}$ und somit letztlich von $\mathbb{U}_{\mathcal{S}} \subseteq \mathbb{U}_{\mathcal{A}}$ abhängt. Es lässt sich sogar vorstellen, dass $errRate_{\mathcal{S}'}(h) = 1$. Ein solch 100-prozentiger Fehler tritt genau dann auf, wenn die Sample-Menge $\mathcal{S}$ keine adäquate Repräsentation von $\mathcal{A}$ darstellt. Das bedeutet, dass $ind_{A_{\mathcal{A}}}(\mathcal{S})$ sehr stark zu $ind_{A_{\mathcal{A}}}(\mathcal{A})$ differiert. Berry und Linoff schlagen deshalb vor, stets eine Vorevaluierung der gegebenen Datenverteilung durchzuführen, um nützliche Ergebnisse aus der Analyse zu erhalten [BL04].

4.4.3 Variabilität durch die Granularität

Der Rough-Set-Klassifikator h, wurde in Kapitel 4.4.1 am Beispiel der Aktivitäten eines einzigen Benutzers eingeführt und interpretiert. Dies stellt eine Limitierung der Anwendungsszenarien für den Klassifikator dar, obwohl h durch die Ununterscheidbarkeitsrelation parametrisierbar ist. Im folgenden werden Attributmengen $B \subseteq A_{\mathcal{A}}$ vorgestellt, die im Hinblick auf h weitere Interpretationen bezüglich des Zielkonzepts liefern können. Ausgangspunkt für diese Betrachtungen ist ein IS $\mathcal{A} = (\mathbb{U}_{\mathcal{A}}, A_{\mathcal{A}})$, welches durch Anpassungen der algebraischen Ausdrücke 4.3 und 4.5 erzeugbar ist.

Applikationsexperten

Projiziert man mittels $ind_B(\mathcal{A}), B \subseteq A_{\mathcal{A}}$ das IS $\mathcal{A}$ auf die Attributmenge

$$B = \{uName, fName, aVersion\}$$

fallen alle Aktivitäten aus $\mathcal{A}$ in Klassen, die sich im Attribut $uName$ (aus Relation $User$) und $fName$ bzw. $aVersion$ (aus Relation *Application*) unterscheiden. Somit ist h in der Lage, die resultierende absolute Häufigkeitsverteilung von $\mathcal{A}$ bezüglich

B zu klassifizieren. Folglich fallen Benutzer, die sich häufiger mit bestimmten Applikationen auseinander setzen als andere, dadurch in die Konzeptapproximation von h und können interpretativ als Experten für verschiedene Anwendungen gesehen werden.

Stabile und bevorzugte Applikationen

Für die Projektion auf die Attributmenge

$$B = \{fName, aVersion\}$$

(aus Relation *Applikation*), enthält $ind_B(\mathcal{A})$ die Frequenz von genutzten Applikationen pro Version. Unter der Annahme, dass nur solche Applikationen von Benutzern gerne verwendet werden, die stabil auf den jeweiligen Computern funktionieren, erzeugt h eine grobe Hypothese, welche stabile Applikationen enthält.

Wichtige Dokumente

In einem vorselektierten Zeitraum sind für Benutzer manche Dokumente wichtiger als andere (vgl. Beispiel 4.4). Dies wird gerade dann deutlich, wenn verschiedene Benutzer bestimmte Dokumente gemeinsam nutzen. Mit der Attributmenge

$$B = \{fName, fLocation\}$$

lässt sich dies über $ind_B(\mathcal{A})$ und h ermitteln. B entstammt dabei ursprünglich aus der Relation *Document*.

Dateiformate und Applikationen

Soll eine Konzeptapproximation beschrieben werden, die Dokumenttypen und dazu häufig genutzte Applikationen in Beziehung setzt, kann dies über die Attributmenge

$$B = \{typeId, fName\}$$

erfolgen. $typeId$ und $fName$ sind hierbei Attribute der Relation *Document*.

4.5 Hierarchisches Clustering

Dieses Kapitel beschäftigt sich mit der Konzeption eines agglomerativen hierarchischen Clustering-Algorithmus, der in der Lage ist gesammelte Aktionen aus dem CAPLE-Projekt in eine Hierarchie von Gruppen zu überführen. Bevor jedoch ein Ähnlichkeitsmaß auf den Aktionen definiert und ein Verfahren angegeben werden kann, werden typische Abläufe und Strategien im Zusammenhang mit dem hierarchischen Clustering erläutert, die für die Konzeption des Verfahrens benötigt werden.

4.5.1 Typische Abläufe und Strategien

In Kapitel 3.2.2 wurde bereits eine kurze allgemeine Einführung über das Gruppieren von Daten gegeben. Der Prozess des agglomerative hierarchischen Clustering folgt dabei einem typischen Ablauf [BEPW08]:

(i) Initialisierung der Startpartitionierung

(ii) Berechnung der Distanzmatrix

(iii) Suche nach Clustern in der Matrix mit kleinstem Abstand

(iv) Zusammenführung der Cluster

(v) Aktualisierung der Distanzmatrix

(vi) Existieren Objekte ohne Cluster-Zugehörigkeit: gehe zu (iii)

Demnach muss ein Distanzmaß bzw. eine Distanzmetrik auf der Objektmenge definiert werden (vgl. Kapitel 3.2.2), welche es erlaubt, die Entfernung zwischen Objekten numerisch zu erfassen. Darauf basierend wird typischerweise eine Distanzmatrix angelegt, die die ermittelten Abstände zwischen allen Objekten enthält (vgl. vollständiger Graph). Nachdem zwei oder mehrere Objekte bzw. Cluster mit dem kleinsten Abstand gefunden und zusammengefasst wurden, muss für die nächste Iteration intuitiv die Matrix aktualisiert werden. Die Aktualisierungsphase beinhaltet das Löschen von Knoten und Abständen, die durch die Vereinigung zur neuen Gruppen nicht mehr benötigt werden. Abbildung 4.17 illustriert drei mögliche Strategien, den Abstand zwischen Clustern zu messen bzw. die neue Distanz nach einer Fusion von Clustern anzupassen. Beim sogenannten Single-Linkage wird die Distanz durch die Objekte zweier Cluster bestimmt, die die kleinste Entfernung aufweisen. Im Gegensatz dazu wird beim Complete-Linkage der Abstand zweier Cluster durch

die am weitesten entfernten Objekte beschrieben. Eine weitere Strategie verwendet keine absoluten Entfernungen zwischen Objekten sondern den durchschnittlichen Abstand der Objekte des einen Clusters zu Objekten im anderen Cluster [GM98].

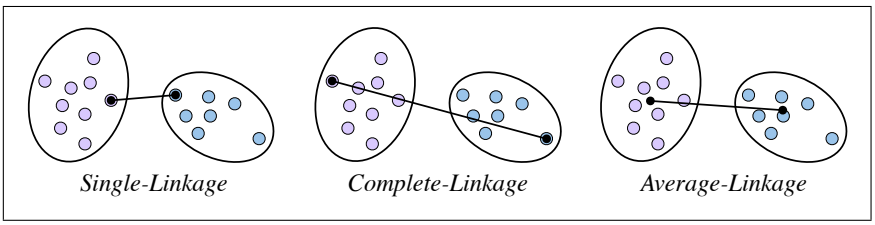


Abbildung 4.17: Messstrategien zur Ermittlung der Entfernung von Clustern

Das Ergebnis des agglomerativen hierarchischen Clusterings kann in einem so genannten Dendrogramm visualisiert werden. Ein Dendrogramm ist dabei eine baumähnliche Struktur, bestehend aus Blättern und einer Hierachie von Knoten. Die Blätter repräsentieren die initiale Partitionierung und die Knoten deren Zugehörigkeit - also die einzelnen Cluster. Der wesentliche Unterschied zwischen einer typischen Baumstruktur und einem Dendrogramm liegt in der Berücksichtigung des Abstands von Knoten bzw. Clustern durch die Länge der Kanten [Seg08]. Abbildung 4.22 (a) skizziert schematisch den Ablauf des agglomerativen hierarchischen Clusterings und Abbildung 4.22 (b) das Resultat in Form eines Dendrogramms:

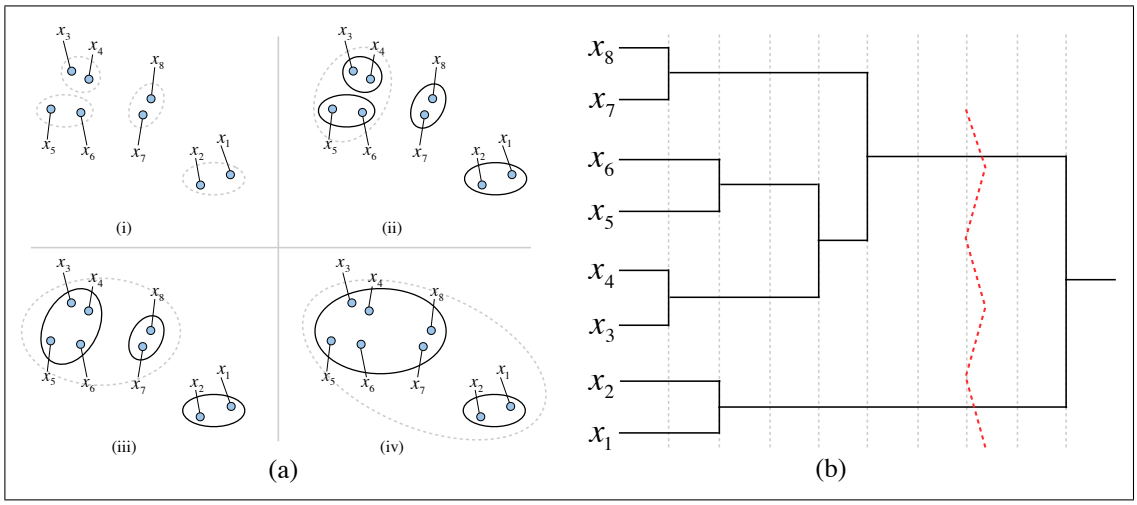


Abbildung 4.18: Prozess des agglomerativen hierarchischen Clustering

Typischerweise wird nach dem Clustering versucht, einen adäquaten Schnitt in der Hierarchie zu finden, der starke Cluster repräsentiert [GM98]. Solch eine mögliche Aufteilung der Partition ist durch die rote Linie in Abbildung 4.22 (b) dargestellt. Demnach ist entschieden, dass für weitere Analysen die Cluster $\{x_3, ..., x_8\}$ und $\{x_1, x_2\}$ zu betrachten sind.

4.5.2 Clustering mit der Ununterscheidbarkeit

Nachdem die wichtigsten Abläufe und Strategien im Umgang mit dem agglomerativen hierarchischen Clustering beschrieben wurden, folgt die schrittweise Konzeption eines Cluster-Algorithmus, der in der Lage ist, Aktionen von Benutzern am Computer zu gruppieren.

Distanzen zwischen Aktionen

Die vorliegenden Daten aus dem CAPLE-Projekt sind zu meist symbolischer Natur. Das bedeutet, dass die Merkmale von Aktionen häufiger aus kategorischen Werten bestehen als aus Zahlen (vgl. Abbildung 4.15). Um auf eine Diskretisierung der Daten weitestgehend zu verzichten, dient das folgende Maß $d : \mathcal{P}(A_{\mathcal{R}}) \times \mathbb{U}_{\mathcal{R}} \times \mathbb{U}_{\mathcal{R}} \rightarrow \mathbb{N}_0$ als Fundament für den Abstand zwischen Aktionen auf einem IS $\mathcal{R} = (\mathbb{U}_{\mathcal{R}}, A_{\mathcal{R}})$:

$$d_B(x, y) = |\{a \in B \mid a(x) \neq a(y)\}|, \forall a \in B \tag{4.16}$$

mit $B \subseteq A_{\mathcal{R}}, x, y \in \mathbb{U}_{\mathcal{R}}$. Die Distanzfunktion d bestimmt anhand einer zu analysierenden Attributmenge B die Anzahl der unterschiedlichen Werte zweier Objekte x und y. Bei genauerer Betrachtung wird deutlich, dass d sogar eine Metrik darstellt, denn es gilt nach Maimon und Rokach für d stets [MR05]:

(i) die Dreiecksungleichung: $d_B(x, y) \leq d_B(x, z) + d_B(y, z)$

(ii) $d_B(x, y) = 0 : x \equiv_{IND_B} y$

mit $B \subseteq A_{\mathcal{R}}, x, y, z \in \mathbb{U}_{\mathcal{R}}$. Intuitiv lässt sich also von einem natürlichen Abstand sprechen.

Der natürlichen Distanzfunktion liegt zugrunde, dass jedes Attribut die gleiche Bedeutung hat. In vielen Anwendungsfällen ist es jedoch möglich, dass manche Attribute wichtiger sind als andere, da sie möglicherweise von ihrer Information aussagekräftiger sind als andere. Aus diesem Grund sei ein Distanzmaß wd vorgestellt, welches eine Gewichtung der Attribute zulässt und auf der natürlichen Distanzfunktion d basiert:

$$wd_B^w(x, y) = d_B(x, y) \cdot (1 + \sum_{\forall a(x) \neq a(y)} w(a)), \forall a \in B \tag{4.17}$$

mit $B \subseteq A_{\mathcal{R}}, x, y \in \mathbb{U}_{\mathcal{R}}$ und der Attributgewichtungsfunktion $w : A_{\mathcal{R}} \rightarrow \mathbb{N}_0$, die jedem Attribut sein subjektives Gewicht zuordnet. wd summiert im Wesentlichen die Gewichtungen w der Attribute a, dessen Werte bezüglich der zwei zu analysierenden

Objekte ungleich sind und multipliziert diese mit dem Wert der natürlichen Distanz d. Durch den Summanden 1 in Formel 4.17 wird lediglich suggeriert, dass mit wd auch d simuliert[5] werden kann. wd stellt somit eine parametrisierbare Erweiterung zu d dar. Es sei an dieser Stelle erwähnt, dass anstatt w auch objektive Gewichtungen, erzeugt durch beispielsweise ein Entropiemaß, möglich sind. Eine Möglichkeit dazu wird in [Kan02] geschildert.

Beispiel 4.5. Anwendung des gewichteten Distanzmaß
Sei $\mathcal{R} = (\mathbb{U}_\mathcal{R}, A_\mathcal{R})$ mit $\mathbb{U}_\mathcal{R} = \{x_1, ..., x_9\}, A_\mathcal{R} = \{User, Application, App\_Version, Document, Time\_of\_Day\}$ gegeben durch Abbildung 4.19 und den subjektiven Gewichtungen $w(a), \forall a \in A_\mathcal{R}$.

-	*User*	*Application*	*App_Version*	*Document*	*Time_of_Day*
$w(a)$	1	3	0	6	2
x_1	Max	Winword	7	Worksheet	morning
x_2	Moritz	Winword	7	Worksheet	morning
x_3	Moritz	Winword	7	Worksheet	morning
x_4	Max	Firefox	2	?	midday
x_5	Max	Firefox	2	?	midday
x_6	Max	Firefox	2	?	midday
x_7	Moritz	Firefox	1	?	midday
x_8	Moritz	Winword	8	Todo	afternoon
x_9	Max	Calc	4	?	afternoon

Abbildung 4.19: Beobachtungen für das beispielhafte Clustering

Für ein Anwendungsszenario sind also Unterschiede in den Werten der Merkmale *Application* und *Document* sehr stark berücksichtigt, während vergleichsweise der Unterschied in den Attributwerten $User$ und $App\_Version$ eine nicht so große bis keine Rolle spielt. Die Distanzen für eine Teilmenge an Objektpaaren $(x, y) \subseteq \mathbb{U}_\mathcal{R} \times \mathbb{U}_\mathcal{R}$ und der Attributmenge $A_\mathcal{R}$ sind im Tableau von Abbildung 4.20 dargestellt.

x	y	$d_A(x,y)$	$wd_A^w(x,y)$
x_1	x_2	1	$1 \cdot (1 + 1) = 2$
x_1	x_4	4	$4 \cdot (1 + 3 + 0 + 6 + 2) = 48$
x_2	x_8	3	$3 \cdot (1 + 0 + 6 + 2) = 27$
x_4	x_9	3	$3 \cdot (1 + 3 + 0 + 2) = 18$
x_6	x_7	2	$2 \cdot (1 + 1 + 0) = 4$
x_7	x_9	4	$4 \cdot (1 + 1 + 3 + 0 + 2) = 28$

Abbildung 4.20: Natürliche und gewichtete Distanzen zwischen Aktionen

[5] Die Simulation lässt sich dann vorstellen, wenn $w(a) = 0, \forall a \in B$ gilt.

Die gewichtete Distanz wd zwischen Objekten ist nach Formel 4.17 dann groß, wenn sie sich in wichtigen Attributwerten unterscheiden, während der Abstand bei der Unterscheidung unwichtiger Merkmale nicht so hoch ausfällt. Dies lässt sich im Tableau nachvollziehen, denn es ist beispielsweise $wd^w_{A_\mathcal{R}}(x_2, x_8) > wd^w_{A_\mathcal{R}}(x_4, x_9)$, obwohl $d_{A_\mathcal{R}}(x_2, x_8) = d_{A_\mathcal{R}}(x_4, x_9)$. Dem gleichen Szenario unterliegen die Abstände der Objekte x_1, x_4 sowie x_7, x_9. Dadurch, dass $Document(x_1) \neq Document(x_4)$ liegen diese Objekte geographisch betrachtet im Raum viel weiter auseinander als die Objekte x_7, x_9.

Cluster-Algorithmus

Auf Basis der gewichteten Distanzfunktion wd aus Abschnitt 4.5.2 lässt sich nun ein Verfahren für das Gruppieren formulieren:

Algorithmus $clusterRel(\mathcal{R}, B \subseteq A_\mathcal{R}, w)$:

- Erzeuge initiale Partitionierung anhand von $ind_B(\mathcal{R})$ und füge sie in Ergebnisrelation $\mathcal{CT} = (\mathbb{U}_{\mathcal{CT}}, A_{\mathcal{CT}})$ ein
- Erzeuge Distanzmatrix $\mathcal{DM} = (\mathbb{U}_{\mathcal{DM}}, A_{\mathcal{DM}})$, resultierend aus initialen Partitionierung und w

while $|\mathbb{U}_{\mathcal{DM}}| > 0$ **do**

- Suche in $\mathbb{U}_{\mathcal{DM}}$ nach kleinster Distanz wd zwischen Aktion x, y. Sind weitere Aktionen betroffen, entscheidet die größere Kardinalität bzw. Münzwurf
- Fusioniere x, y und füge das neue Cluster in $\mathbb{U}_{\mathcal{CT}}$ ein
- Aktualisiere $\mathbb{U}_{\mathcal{DM}}$:
  - Lösche Distanz zwischen x, y
  - Aktualisiere alle weiteren Distanzen von Aktionen, die mit dem neuen Cluster in Beziehung stehen mittels Single- bzw. Complete-Linkage
  - Lösche unbenötigte Distanzen zwischen Aktionen und dem neuen Cluster

end

return $\mathcal{CT}$

Abbildung 4.21: Algorithmus für das agglomerative hierarchische Clustering

Die Funktionssignatur des Algorithmus *clusterRel* aus Abbildung 4.21 besteht aus einer gegebenen Relation $\mathcal{R} = (\mathbb{U}_\mathcal{R}, A_\mathcal{R})$, der zu analysierenden Attributmenge $B \subseteq A_\mathcal{R}$, sowie aus der Vorschrift $w : A_\mathcal{R} \rightarrow \mathbb{N}_0$, die jedem Attribut aus B eine numerische Gewichtung zuordnet. Dabei wird zunächst die Ununterscheidbarkeitsrelation als initiale Startpartitionierung genutzt, womit zu Beginn bereits eine Gruppierung auf Objektebene erwirkt wird. Auf Basis dieser disjunkten Teilmengen wird in einem weiteren Schritt die Distanzmatrix erzeugt, die alle nötigen gewichteten Abstände zwischen den Klassen und deren Kardinalität enthält. Schrittweise wird nun nach den Klassen mit dem kleinsten Abstand in der Matrix gesucht, welche dann fusioniert werden. Nach einer Aktualisierungsphase der Distanzmatrix wird solange mit der Gruppierung von Klassen fortgefahren, bis keine weiteren Einträge in der Matrix enthalten sind. Das Ergebnis von *clusterRel* ist eine hierarchische Struktur, repräsentiert durch eine Relation, die alle durchgeführten Cluster-Schritte beinhaltet.

Um die Funktionsweise des agglomerativen hierarchischen Cluster-Algorithmus zu veranschaulichen, ist im Folgenden ein Beispiel angegeben, welches grafisch das IS aus Abbildung 4.19 nach der Complete-Linkage-Strategie schrittweise clustert.

Beispiel 4.6. Illustration von Algorithmus *clusterRel*
Sei das IS mit den subjektiven Gewichtungen gegeben durch Abbildung 4.19. Der erste Schritt beim Clustern wird durch Abbildung 4.22 (a) illustriert.

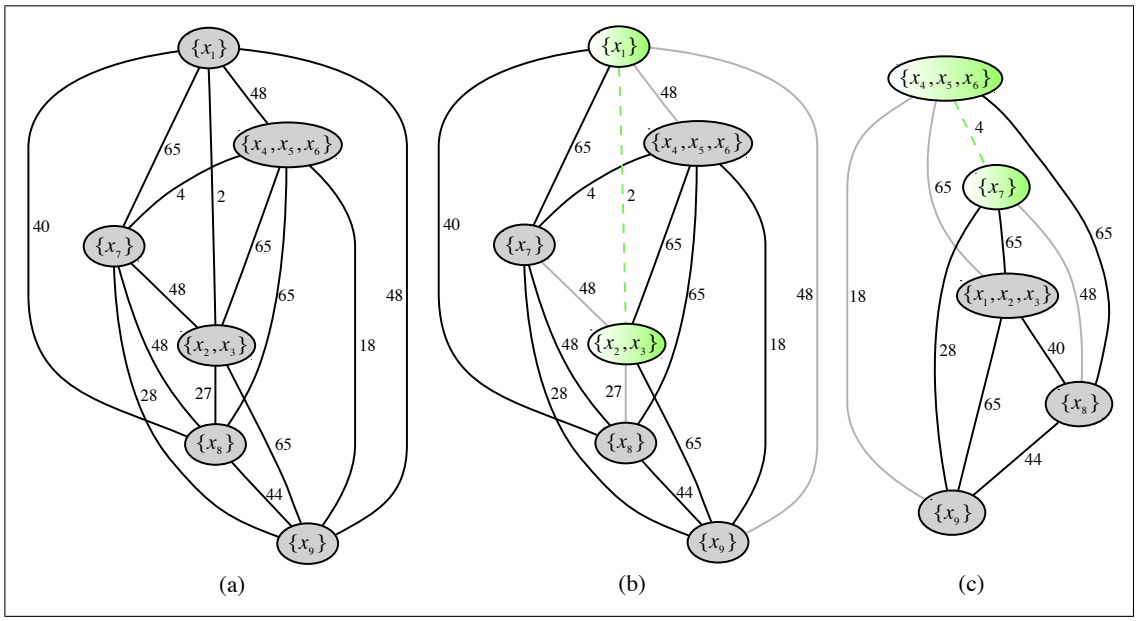


Abbildung 4.22: Initiale Partitionierung und iteratives Gruppieren

Diese Darstellung zeigt eine Datenreduktion[6] der Objekte in Form eines ungerichteten vollständigen Distanzgraphen, die mittels der Ununterscheidbarkeitsrelation erzielbar ist. Es ist ersichtlich, dass die Objekte $\{x_4, x_5, x_6\}$ und $\{x_2, x_3\}$ bereits eigenständige Gruppen bilden. Alle weiteren Teilmengen der initialen Partitionierung sind noch einelementig. Zu Beginn ergibt sich somit eine Anzahl von $\binom{|IND_{A_{\mathcal{R}}}|}{2}$ Einträgen[7] in der Distanzmatrix, welche für das weitere Vorgehen zu betrachten sind.

Nachdem die initiale Partitionierung erzeugt und die darauf basierende Distanzmatrix aufgestellt ist, kann die erste Iteration des eigentlichen Clustering beginnen. Sie ist durch Abbildung 4.22 (b) dargelegt. Da $wd^w_{A_{\mathcal{R}}}(x_1, x_2) = 2$ bzw. $wd^w_{A_{\mathcal{R}}}(x_1, x_3) = 2$ die kleinsten Distanzen im Graphen darstellt, werden diese Teilmengen zu $\{x_1, x_2, x_3\}$ vereinigt. Die grau hinterlegten Kanten symbolisieren dabei für die weiteren Iterationen unnötige Distanzen und müssen entfernt werden. Bedingt durch die Anwendung des Complete-Linkage sind, ausgehend von dem neuen Cluster, genau die Kanten zu Gruppen zu löschen, dessen Distanz die kleinste ist. Es ist beispielsweise $wd^w_{A_{\mathcal{R}}}(x_1, x_9) = 48$ und $wd^w_{A_{\mathcal{R}}}(x_2, x_9) = 65$. Da $\{x_1\}$ und $\{x_2, x_3\}$ in ein neues Cluster fallen, muss nun $wd^w_{A_{\mathcal{R}}}(x_1, x_9)$ aus der Matrix entfernt werden[8].

Abbildung 4.22 (c) illustriert den Ablauf der zweiten Iteration. Der kleinste Abstand zwischen den verbleibenden Gruppen lautet vier. Somit muss die Gruppen $\{x_4, x_5, x_6\}$ mit $\{x_7\}$ verschmolzen werden und der Graph adaptiert werden. In dieser Iteration sind genau vier Kanten davon betroffen und müssen entfernt werden.

Analog zu den vorangegangen Schritten von *clusterRel* veranschaulicht Abbildung 4.23 (a-c) die verbleibenden drei Schritte. Dabei werden in der dritte Iteration die Cluster $\{x_4, x_5, x_6, x_7\}$ und $\{x_9\}$ fusioniert, während im vorletzten Schritt die Gruppe Cluster $\{x_1, x_2, x_3, x_8\}$ entsteht. In der finalen Iteration existiert kein unbetroffenes Cluster mehr. Das bedeutet, dass die beiden Cluster $\{x_4, x_5, x_6, x_7, x_9\}$ und $\{x_1, x_2, x_3, x_8\}$ letztlich verschmolzen werden. Danach sind keine weiteren Knoten im Distanzgraphen respektive Einträge in der Matrix enthalten. Die Abbruchbedingung von *clusterRel* ist somit erfüllt und das Verfahren endet an dieser Stelle.

[6] Zur Nachvollziehbarkeit werden hier alle Klassen vollständig dargestellt.
[7] Bei $n > 1$ Knoten enthält ein vollständiger ungerichteter Graph immer $\binom{n}{2}$ Kanten [Tur04].
[8] Es sei angemerkt, dass beim Single-Linkage die Distanz $wd^w_{A_{\mathcal{R}}}(x_2, x_9)$ entfernt werden müsste.

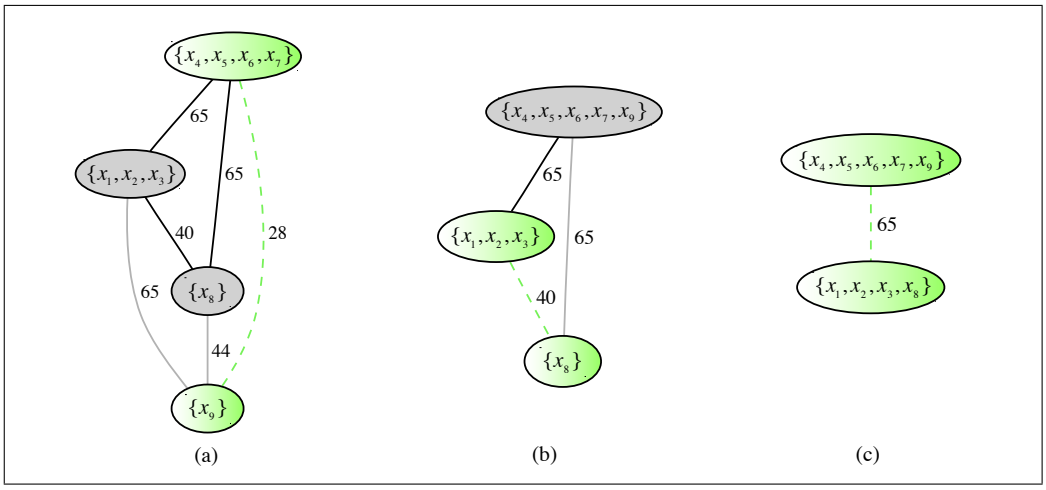


Abbildung 4.23: Verbleibende Iterationen beim hierarchischen Clustering

Damit unabhängig von der grafischen Repräsentation aus diesem Beispiel deutlich wird, welche Ausgabe der Cluster-Algorithmus liefert, sei zusätzlich Abbildung 4.24 betrachtet. Sie enthält eine Relation, die jeden Schritt des Algorithmus protokolliert:

it_No	*c_Id*	*c_1*	*c_2*	*values*	*card*	*wd*
0	A	⊥	⊥	{Max,Winword,7,Worksheet,morning}	1	⊥
0	B	⊥	⊥	{Moritz,Winword,7,Worksheet,morning}	2	⊥
0	C	⊥	⊥	{Max,Firefox,2,?,midday}	3	⊥
0	D	⊥	⊥	{Moritz,Firefox,1,?,midday}	1	⊥
0	E	⊥	⊥	{Moritz,Winword,8,Todo,afternoon}	1	⊥
0	F	⊥	⊥	{Max,Calc,4,?,afternoon}	1	⊥
1	G	A	B	⊥	3	2
2	H	C	D	⊥	4	4
3	I	H	F	⊥	5	28
4	J	G	E	⊥	4	40
5	K	I	J	⊥	9	65

Abbildung 4.24: Ergebnis des hierarchischen Clustering als mögliche Relation

4.6 Ermittlung ähnlicher Verhaltensmuster mit der Grenzregion

Die Thematik des kollaborativen Filterns behandelt im Wesentlichen die Aufgabe, Vorlieben verschiedener Individuen dazu zu verwenden, Empfehlungen für andere Benutzer auszusprechen. Solch eine typische Empfehlungs-Engine lässt sich auf

Online-Shops wie Amazon[9] nachvollziehen [Seg08]. Im Zusammenhang mit der RST konnten unlängst wichtige Erkenntnisse auf dem Forschungsgebiet des kollaborativen Filterns u.a. in [HQB08, TSWH08] erzielt werden. Dieses Kapitel zeigt eine weitere Idee auf, mit der es unter Zuhilfenahme der RST möglich ist, die Interessen verschiedener Benutzer quantitativ zu vergleichen und gegebenenfalls auf Basis dieser Kollation Empfehlungen zu generieren.

4.6.1 Grundlegende Idee

Damit es möglich ist, die Interessen zwischen Benutzern in einem vordefinierten Zeitraum zu vergleichen, sei zunächst ein IS $\mathcal{R} = (\mathbb{U}_{\mathcal{R}}, A_{\mathcal{R}})$ mit $A_{\mathcal{R}} = \{User, ...\}$ zu untersuchen, welches ausschließlich die Aktivitäten der zwei Benutzer *Max* und *Moritz* beinhaltet ($V_{User} = \{Max, Moritz\}$). Betrachtet man die gegebenen Objekte isoliert nach ihren Benutzern, lässt sich vorstellen, dass *Max* und *Moritz* in ihrer Tätigkeit einen gewissen Zweck verfolgen. In einigen Fällen mag sich der Zweck sogar ähneln, obwohl die Individuen unabhängig voneinander arbeiten. Das führt zu der Überlegung, dass nicht zwangsläufig die Aktivitäten beider Benutzer disjunkt sein müssen. Gerade im Bereich des *eLearning* sind solche Beziehungen zwischen verschiedenen Benutzern denkbar, wo Schüler oder Studenten autonom Aufgabenstellungen des Lehrers bearbeiten. Betrachtet man also bildlich die Schnittmenge der gegebenen Aktivitäten von *Max* und *Moritz*, lässt sich eine quantitative Aussage darüber machen, wie ähnlich das Interesse der Nutzer ist. Für das Erfassen von Ähnlichkeiten bietet die RST mit ihrer mengentheoretischen Sichtweise ein mögliches Werkzeug. Dazu lassen sich die unabhängigen Zielkonzepte $X \cap Y = \emptyset$ mit $X \cup Y = \mathbb{U}_{\mathcal{R}}$ anhand des Entscheidungsattributs $User$ leicht mit $IND_{\mathcal{R}}(\{User\})$ generieren. X steht dabei für das durch Max und Y für das durch $Moritz$ erzeugte Konzept. Das weitere Ausnutzen der Äquivalenzrelation $IND_{\mathcal{R}}(B)$ mit $B = A_{\mathcal{R}} - \{User\}$ kann nun in gewissen Fällen dazu führen, dass X respektive Y nur grob beschrieben werden kann, so dass sich lediglich für die Exaktheit der Konzeptapproximation $0 < \alpha_X^B, \alpha_Y^B < 1$ einstellt (vgl. Definition 2.9). Durch den Wegfall der Information $User$ gibt es nun Objekte, die nicht eindeutig X oder Y zugeordnet werden können. Allgemein betrachtet entsteht die Menge $BND_B(\{User\}) = \overline{\underline{X}}_B \cup \overline{\underline{Y}}_B$ (vgl. Definition 2.10), die Äquivalenzklassen solcher Aktivitäten enthält, die beide Benutzer durchgeführt haben. Doch im speziellen Fall, dass $V_{User} = 2$ reicht es, entweder $\overline{\underline{X}}_B$ oder $\overline{\underline{Y}}_B$ zu berechnen, denn es gilt $BND_B(\{User\}) = \overline{\underline{X}}_B = \overline{\underline{Y}}_B = \overline{X}_B \cap \overline{Y}_B$. Besitzt $BND_B(\{User\})$ ausreichend viele Objekte bzw. Äquivalenzklassen, weisen

[9] `http://amazon.com`, Stand: 05.10.09

Max und $Moritz$ zumindest partiell gemeinsame Präferenzen auf, so dass gegenseitig Empfehlungen ausgesprochen werden könnten. Eine solche Empfehlung[10] lässt sich beispielsweise so vorstellen, dass Max die Aktionen von $Moritz$ zur Verfügung gestellt bekommt, die er noch nicht von $Moritz$ kennt. Es lässt sich leicht sehen, dass diese Aktionen genau $\underline{Y}_B$ entsprechen. Zur Veranschaulichung dieser Idee sei folgende Abbildung 4.25 zu betrachten. Sie enthält eine beispielhafte Datenverteilung der Aktionen von Max und $Moritz$ und die jeweiligen Konzeptapproximationen der RST:

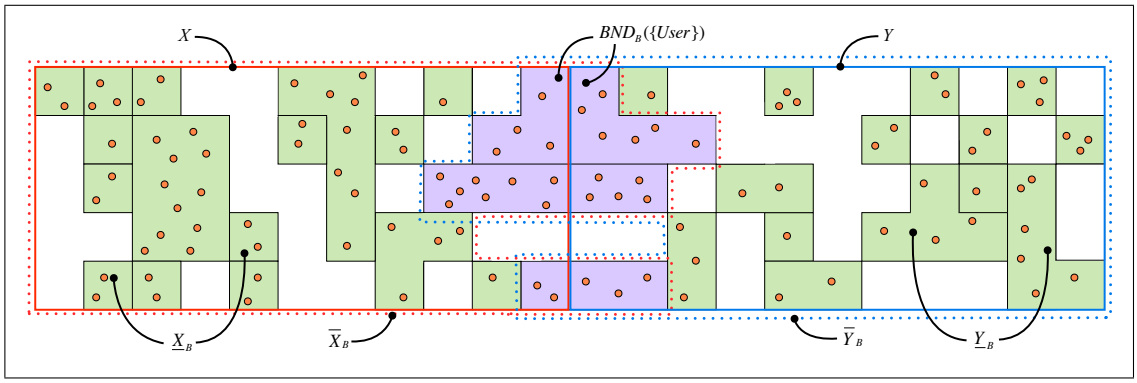


Abbildung 4.25: Schematische Darstellung für das Finden ähnlicher Interessen

4.6.2 Verfahren zum Finden ähnlicher Präferenzen

Ein Algorithmus zum Finden ähnlicher Benutzerinteressen benötigt nach den Ideen von Kapitel 4.6.1 als Eingabeparameter eine Relation $\mathcal{R}$ mit einem gegebenen Attribut d, welches die zwei zu vergleichenden Konzepte beinhaltet. Das bedeutet, dass insbesondere für den Wertebereich V_d von d mindestens $|V_d| \geq 2$ gelten muss. Zur Evaluierung der Ähnlichkeit lässt sich vereinfacht ein Schwellwert $\alpha \in \mathbb{N}$ vorstellen, anhand dessen das Verfahren entscheidet, ob ähnliche Präferenzen bestehen oder nicht. Dazu müssen zunächst mittels adäquater algebraischer Ausdrücke die Zielkonzepte aus $\mathcal{R}$ extrahiert werden. Durch die Vereinigung der zwei Zielkonzepte entsteht $\mathcal{A}$ mit $\mathbb{U}_\mathcal{A} \subseteq \mathbb{U}_\mathcal{R}$ und $A_\mathcal{A} = A_\mathcal{R}$. Auf $\mathcal{A}$ kann nun $BND_B(\{d\})$ mit $B = A_{\mathcal{A}-\{d\}}$ berechnet werden. Ist nun $|BND_B(\{d\})| \geq \alpha$, so liegen genügend Gemeinsamkeiten zwischen den Benutzern vor und der Algorithmus gibt $true$ aus. In anderen Fällen wird $false$ zurückgegeben und keine Ähnlichkeit ist gegeben. Abbildung 4.26 beschreibt dieses Vorgehen konzeptionell anhand des Algorithmus *equal_pref*.

[10] Im Zusammenhang mit dem *eLearning* wäre eine Empfehlung z.B. ein Vorschlag oder Tipp in Form einer Menge von Aktionen, die für die Lösung der Aufgabenstellung hilfreich sein können.

Algorithmus $equal\_pref(\mathcal{R}, d \in A_{\mathcal{R}}, \{s,t\} \subseteq V_d, \alpha \in \mathbb{N})$:

$\mathcal{A} = \sigma_{d(x)=s}(\mathcal{R}) \cup \sigma_{d(x)=t}(\mathcal{R}), x \in \mathbb{U}_{\mathcal{R}}$

$\mathcal{B} = boundary_{A_{\mathcal{A}}-\{d\}}(\mathcal{A}, \sigma_{d(x)=s}(\mathcal{A})), x \in \mathbb{U}_{\mathcal{A}}$

if $\gamma_{\mathtt{sum}(card);\emptyset}(\mathcal{B}) \geq \alpha$ **then**

  return $true$

end

return $false$

Abbildung 4.26: Finden von ähnlichen Verhaltensmustern

Beispiel 4.7. Anwendung von Algorithmus *equal_pref*
Es sei ein IS $\mathcal{R}$ gegeben durch Abbildung 4.27 mit dem Attribut $User$, den zu vergleichenden Nutzern $\{Max, Moritz\} \subseteq V_{User}$ und $\alpha = 4$:

-	*User*	*Application*	*Time_of_Day*
x_1	Max	Winword	morning
x_2	Moritz	Winword	morning
x_3	Peter	Winword	afternoon
x_4	Moritz	Winword	morning
x_5	Max	Firefox	midday
x_6	Max	Texshop	afternoon
x_7	Moritz	Firefox	midday
x_8	Moritz	Winword	afternoon
x_9	Max	Outlook	morning
x_{10}	Peter	Outlook	morning

Abbildung 4.27: Beobachtungen zum Finden ähnlicher Präferenzen

Durch Anwendung des relationalen Ausdrucks $\sigma_{User(x)=Max}(\mathcal{R}) \cup \sigma_{User(x)=Moritz}(\mathcal{R})$ werden nur solche Objekte x aus $\mathbb{U}_{\mathcal{R}}$ selektiert, die die Prädikate $User(x) = Max$ und $User(x) = Moritz$ erfüllen. Es entsteht $\mathcal{A} = (\mathbb{U}_{\mathcal{A}}, A_{\mathcal{A}})$ mit $\mathbb{U}_{\mathcal{A}} \subseteq \mathbb{U}_{\mathcal{R}}, A_{\mathcal{A}} = A_{\mathcal{R}}$ und dem Wertebereich $V_{User} = \{Max, Moritz\}$. Dadurch dass $|V_{User}| = 2$, reicht es nun aus $BND_{A_{\mathcal{A}}-\{User\}}$ beispielsweise durch $boundary_{A_{\mathcal{A}}-\{User\}}(\mathcal{A}, \sigma_{User(x)=Max}(\mathcal{A}))$ zu berechnen. Das Ergebnis ist ein IS mit folgender Gestalt:

Card	*Application*	*Time_of_Day*
3	Winword	morning
2	Firefox	midday

Abbildung 4.28: Berechnung der Grenzregion als quantitatives Maß für Ähnlichkeiten

Abbildung 4.28 enthält solche Aktionen, welche von den Benutzern Max und $Moritz$ gleichermaßen durchgeführt wurden. Die Summe über all diese Objekte beträgt $5 > \alpha$. Das bedeutet, dass sich $\frac{5}{8} \approx 62\%$ aller getätigten Aktionen der Benutzer ähneln. Dadurch würde $equal\_pref$ die Ausgabe $true$ gegeben und der Algorithmus endet.

5 Implementierung

Dieses Kapitel befasst sich mit der Umsetzung der entwickelten Entwürfe aus dem vorangegangenen Kapitel 4. Die dort formulierten und implementierungsunabhängigen Konzepte werden in diesem Kapitel im Hinblick auf die funktionale Erweiterung eines bestehenden Datenbanksystems implementiert. So kann ein objektrelationales DBS, angereichert mit Methoden der *Rough Set Theory*, entstehen, welches durch externe Software-Systeme[1] angesteuert werden kann. Ferner kann das erweiterte ORDBS auf direktem Wege als experimentelle Plattform für die wissenschaftliche Analyse von Benutzerbeobachtungen fungieren. Motiviert und diskutiert wird dabei zunächst die Integration der neuen Funktionalität in die bestehende Systemlandschaft des CAPLE-Projekts. Danach folgen in den einzelnen Unterkapiteln wichtige Aspekte aus der Konzeptionsphase, die im Kontext der Implementierung betrachtet und anhand von Code-Fragmenten konkretisiert werden.

5.1 Achitektur der funktionalen Erweiterungen

Dieser Abschnitt gibt einen grundlegenden Überblick über die in Kapitel 4 konzeptionierten Ideen in Form von Modulen und präsentiert die Einbettung dieser in die bestehende Systemlandschaft. Zusätzlich stellt sich diese Passage wichtigen technologischen Fragestellungen im Hinblick auf die Umsetzung der Module und arbeitet ein einheitliches Bild für die Verwendung der neuen Funktionalitäten heraus.

5.1.1 PL/pgSQL als Fundament der Umsetzung

PostgreSQL 8.3, als freiverfügbares und weitestgehend standardkonformes[2] DBS, soll im weiteren Verlauf dieser Arbeit den Grundstein für die Implementierung bilden.

[1] Insbesondere sind die *Analysis Tools* gemeint, die über das CAPLE-Framework auf diverse DBS zugreifen (vgl. Kapitel 4.1.1).

[2] Ein Überblick über die Konformitäten von PostgreSQL zu den SQL-Standards findet sich unter: `http://www.postgresql.org/docs/8.3/static/features.html`, Stand: 20.10.09

Neben den typischen OR-Konzepten liefert PostgreSQL standardmäßig die prozedurale Programmiersprache PL/pgSQL (Procedural Language/PostgreSQL SQL) mit. Dadurch wird es möglich, Kontrollstrukturen mit SQL-Befehlen so zu kombinieren, dass komplexe Berechnungen aus dem DBS heraus möglich sind. Ist eine solche Berechnungssequenz formuliert, kann sie als funktionale Erweiterung im DBMS gespeichert und wie eine Prozedur von externen Programmen aufgerufen werden[3]. In vielen Szenarien lässt diese Architektur zu, den Kommunikationsaufwand zwischen Client-Anwendungen und Datenbank-Server stark zu reduzieren [Eis03]. Dazu sei das typische Szenario betrachtet, in dem Client-Applikationen Datenbankanfragen stellen, auf die Antwort warten, lokale Berechnungen auf Basis der Antwort durchführen, um letztlich wieder Anfragen an das DBS zu übermitteln. Die Auslagerung der Programmlogik in eine oder mehrere server-seitige PL/pgSQL-Prozeduren würde aus Sicht der Client-Anwendung nur wenige anstatt vieler Anfragen an den Datenbank-Server zur Folge haben. In Abbildung 5.1 (a-b) wird diese Kommunikationseinsparung schematisch veranschaulicht.

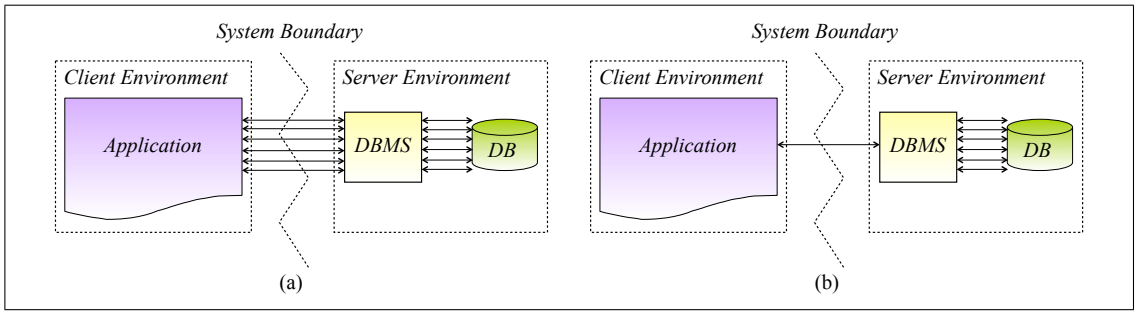


Abbildung 5.1: Mögliche Kommunikationseinsparung durch server-seitige Funktionen

Nachdem zwei wichtige Vorteile[4] im Umgang mit PL/pgSQL näher beleuchtet wurden, folgt nun zur Verdeutlichung der Syntax ein einfaches Programmbeispiel. Zu diesem Zweck sei die Aufgabe betrachtet, aus den CAM des spezifizierten Schemas aus Kapitel 4.1.2 die drei am längsten andauernden Aktivitäten eines gegebenen Benutzers zu finden und die involvierten Applikationsnamen als Multimenge zu extrahieren. Listing 5.1 beschreibt eine Lösungsmöglichkeit für dieses Vorhaben durch die Methode `threeMostDurableApps`:

```
/* Beispiel-Prozedur zur Erläuterung grundlegender PL/pgSQL-
   Syntax */
CREATE FUNCTION threeMostDurableApps(username TEXT)
  RETURNS SETOF TEXT AS /* Definition der Rückgabe */
```

[3] Eine Ausführung zu diesem Thema findet sich in Kapitel 5.1.3.
[4] Eine ausführliche Auflistung weiterer Vorteile im Bezug auf die Verwendung von prozeduralen Erweiterungssprachen in DBS findet sich z.B. in [SHS05].

```
$BODY$
DECLARE /* Variablendeklaration */
  userExists BOOLEAN := FALSE;
  counter INTEGER := 0;
  query TEXT :='';
  result RECORD;
BEGIN /* Beginn der Programmlogik */
  /* Prüfung, ob Benutzer überhaupt existiert */
  SELECT INTO userExists (R.card = 1)::BOOLEAN FROM (SELECT COUNT (*) AS card FROM
     "RCAM"."User" WHERE "uName" = $1) AS R;
  IF userExists != TRUE THEN
    RAISE EXCEPTION 'unique user: % could not be found.', $1;
  END IF;

  /* Hauptanfrage, über welche iteriert wird */
  FOR result IN SELECT App."fName" AS AppName from "Activity" Act
    JOIN "User" User ON Act."userId" = User."uId"
    JOIN "Application" App ON Act."appId" = App."fId"
    WHERE User."uName" = $1
    ORDER BY Act."aDuration" DESC LOOP

    /* Rückgabe des Applikationsnames */
    IF counter < 3 THEN
      RETURN NEXT result.AppName;
    END IF;
    counter := counter + 1;
  END LOOP;
END /* Ende der Programmlogik */
$BODY$
  LANGUAGE 'plpgsql' STABLE;
```

Listing 5.1: Beispielprozedur `threeMostDurableApps` mit PL/pgSQL

Der Ablauf dieser PL/pgSQL-Prozedur ist dabei wie folgt strukturiert: zunächst findet eine Überprüfung statt, ob der übergebene Parameter[5] `username` überhaupt in der Datenbank existiert. Es wird damit gerechnet, dass genau ein solcher Benutzer vorhanden ist. Wird kein oder mehrere Benutzer gefunden, wirft die Methode einen Ausnahmefehler (`RAISE EXCEPTION`) und die weitere Abarbeitung des Programms wird beendet. Ist diese Überprüfung abgeschlossen beginnt der Hauptteil der Methode. Es wird über eine Anfrage nach Aktionen von `username` mit den involvierten Applikationen gestellt. Das Resultat dieser Anfrage ist eine Relation, die absteigend nach dem Attribut `aDuration` sortiert ist. Das bedeutet, dass zur Lösung der eigentlichen Aufgabe nur die ersten drei Tupel relevant sind[6]. Es wird also über die Ergebnisrelation mit dem `FOR` ... `IN` ... Konstrukt iteriert, wobei nur die ersten drei Applikationsnamen (`RETURN NEXT result.AppName`) ausgegeben werden. `result` ist

[5] Parameter können in der Funktion selbst auch über die Schlüsselwörter `$1`, `$2`, ... angesprochen werden, wobei `$`i für die i-te übergebene Variable steht.

[6] Es sei erwähnt, dass die Aufgabe durch Anwendung des Schlüsselworts `LIMIT` in der Anfrage auf einem direkten Weg gelöst werden kann.

dabei eine Variable vom sogenannten Pseudotyp[7] `RECORD`, der in jeder Iteration als Platzhalter für ein untypisiertes Tupel fungiert. Die Rückgabeart selbst wurde bereits in der Funktionssignatur durch die Schlüsselwörter `SETOF TEXT` spezifiziert, welche anweisen, dass genau eine Multimenge von String-Werten[8] als Rückgabewert zu erwarten ist.

Für syntaktische Besonderheiten von PL/pgSQL als prozedurale Erweiterungssprache für PostgreSQL sei im weiteren Verlauf dieser Arbeit insbesondere auf die PostgreSQL-Dokumentation in [Eis03, Pos09] sowie auf die Fachbücher [Mom01, Sch09] verwiesen. Aufgrund der syntaktischen und semantischen Ähnlichkeit von PL/pgSQL zu PL/SQL[9] sei an dieser Stelle ebenfalls auf die gängige Literatur aus dem ORACLE-Umfeld referenziert: [UHMH05, FP05].

5.1.2 Überblick über die neue Systemlandschaft

Die Konsumenten des CAPLE-Frameworks sind *Wrapper* und *Analysis Tools*, die auf der einen Seite die Aktivitäten des Benutzer observieren und auf der anderen Seite die gespeicherten Beobachtungen für spezielle Auswertungen nutzen. Das CAPLE-Framework stellt somit den abstrahierten Client/Server-Betrieb zwischen den Konsumenten und den verwendeten Datenbanken sicher (vgl. Kapitel 4.1.1). In diesem Zusammenhang dient die Neustrukturierung des in Kapitel 4.1.2 vorgeschlagenen Schemas zum einen für die Persistierung von CAM in einem ORDBS, was zum anderen dazu ausgenutzt werden kann, die implementierten relationalen Operationen dieses ORDBS für die Auswertung der CAM zu verwenden. Die in den Kapiteln 4.3 bis 4.6 vorgestellten Ideen wurde im Hinblick auf diesen Gedankengang konzipiert, so dass nun mit Hilfe von PL/pgSQL die Integration der Ideen in PostgreSQL umgesetzt werden kann. Zusätzlich lassen sich zwei praxisrelevante und damit nicht unerhebliche Vorteile aus diesem Vorhaben ableiten:

(i) Aus Sicht von *Analysis Tools*, welche auf Basis der vorgeschlagenen Analysemethoden arbeiten, lässt sich der Kommunikationsaufwand zu PostgreSQL stark reduzieren (vgl. Kapitel 5.1.1).

[7] Weitere Informationen zu der Typdeklaration bzw. Row-Typen finden sich unter `http://www.postgresql.org/docs/8.3/static/plpgsql-declarations.html` bzw. `http://www.postgresql.org/docs/8.3/interactive/datatype-pseudo.html`, Stand 24.10.09

[8] Eine Auflistung der Datentypen von PostgreSQL ist unter: `http://www.postgresql.org/docs/8.3/interactive/datatype.html` beschrieben, Stand 24.10.09

[9] PL/SQL steht für Procedural Language/SQL und wurde ursprünglich für das DBS der Firma ORACLE entwickelt.

(ii) Auf (i) basierend ist es nicht notwendig wie bei anderen Implementierungen der RST die zu analysierenden Daten erst in den lokalen Speicher zu laden (vgl. Kapitel 3.3.3), sondern „direkt“ bzw. datennah auf diese zuzugreifen.

Die genannten Aspekte und die damit verbundenen Vorteile lassen sich zur Verdeutlichung in einem erweiterbaren Gesamtsystem veranschaulichen. Dieses System basiert auf PostgreSQL 8.3 und enthält damit standardmäßig durch SQL ein problemorientiertes Werkzeug für die Selektion und Transformation von Daten sowie Kernfunktionalitäten[10] für die Extraktion von Mustern. Dadurch können datennah die wesentlichen Phasen des *Knowledge Discovery in Databases* aus PostgreSQL heraus unterstützt werden. Die Integration des aus einem neuen Blickwinkel betrachteten DBS in die bestehende Client/Server-Architektur[11] lässt sich mit leichten Anpassungen auf Seiten des CAPLE-Frameworks realisieren. So kann eine neue Systemlandschaft für die wissenschaftliche Analyse von Daten im Kontext des CAPLE-Projekts entstehen, wie Abbildung 5.2 verdeutlicht:

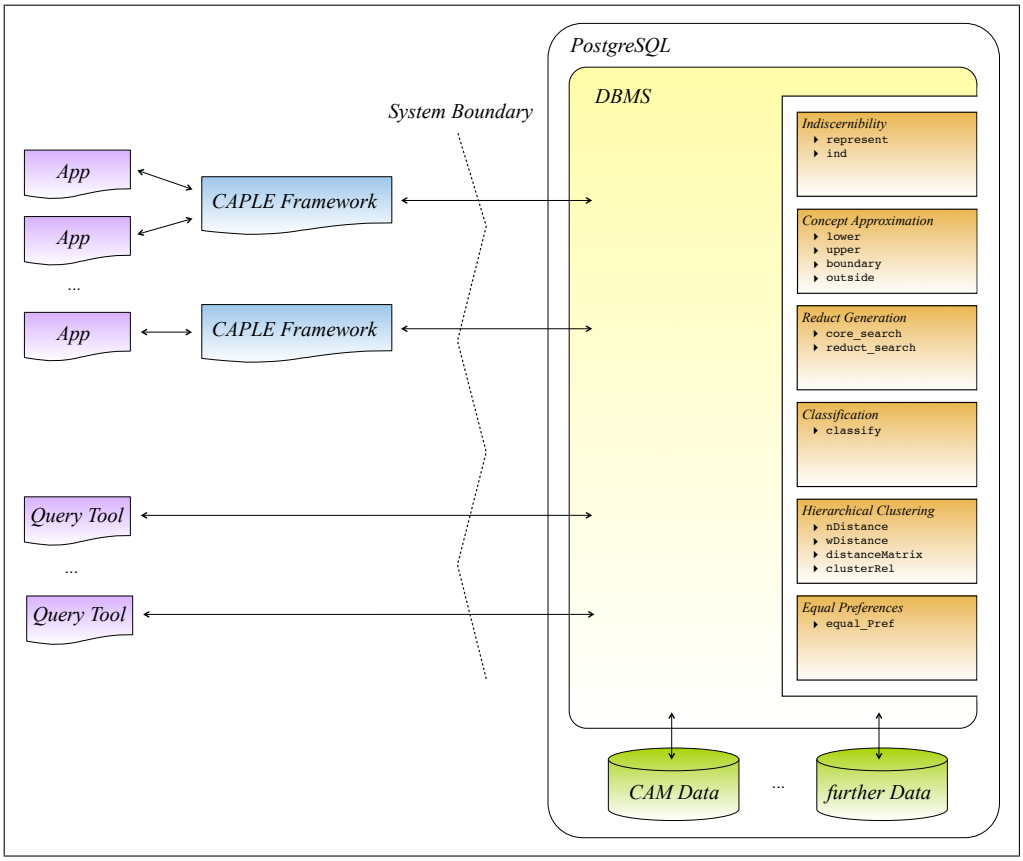


Abbildung 5.2: Integration neuer Funktionalität in die CAPLE-Systemlandschaft

[10] Die ausführliche Betrachtung dieser in Prozeduren gekapselten Funktionalitäten findet in den verbleibenden Abschnitten dieses Kapitel 5 statt.

[11] Hier ist die erläuterte Architektur aus Kapitel 4.1.1 gemeint.

5.1.3 Design der Funktionssignaturen

Abbildung 5.2 beschreibt nicht nur die adaptierte Software-Architektur des CAPLE-Projekts. Ebenfalls sind die vierzehn funktionalen Erweiterungen von PostgreSQL skizzieret. Sie sind thematisch in Blöcke unterteilt und verfolgen, wenn auch partiell aufeinander basierend, unterschiedliche Ansätze für die Extraktion von Mustern. Damit die einzelnen Prozeduren nicht nur für einen speziellen Anwendungsfall eingesetzt werden können, sind sie von ihrer Funktionssignatur mit einstellbaren Parametern versehen (vgl. Kapitel 4.3 bis 4.6). Dennoch muss sich im Hinblick auf die Implementierung der neuen Funktionalitäten darüber Gedanken gemacht werden, wie die Funktionssignatur in den einzelnen Fällen entworfen wird. Die folgende Aufzählung zeigt dazu wichtige Parameter aus der Konzeptionsphase, die mittels PL/pgSQL abgebildet werden müssen:

- Relationen, Tabellen bzw. Informationssysteme
- Einzelne Attribute bzw. Attributmengen
- Gewichtungsfunktionen für Attributmengen
- Numerische Parameter
- Formatierung der Rückgabe

Um die Prozeduren in ihrer Verwendung weitestgehend generisch zu halten, lassen sich Relationen wegen ihre Eindeutigkeit im Datenbank-Schema durch ihren Namen repräsentieren. Soll eine Auswertung auf einer nicht physisch vorhandenen Tabelle durchgeführt werden, können SQL-Statements gekapselt in Sichten (engl. *Views*) dafür genutzt werden logischen Tabellen zu erzeugen. Sie sind ebenfalls von ihrer Namensgebung in jedem Schema eindeutig und lassen sich wie reelle Relationen durch einen String-Wert (z.B. `relation TEXT :='rel'`), der den Namen beinhaltet, abbilden. Problematisch ist dieser eingeschlagene Weg bei der Abbildung von Attributmengen, denn die Ununterscheidbarkeitsrelation der RST arbeitet häufig nur auf einer Teilmenge der gesamten Attribute einer Relation. Da aber die Anzahl der Attribute im Vergleich zu den Tupeln einer Relation verhältnismäßig klein ausfällt, werden die Attribute als String-Feld (z.B. `attList TEXT[] := '{a1,a2,a3}'`) an die jeweilige Prozedur übergeben. Sofern die Prozedur ebenfalls eine numerische Gewichtung der Attribute erwartet, kann dies durch einen ganzzahligen Vektor (z.B. `attWeights INTEGER[] := '{5,3,1}'`) selber Länge realisiert werden. Einzelne Attribute und numerische Werte können direkt in der Parameterliste der Funktion aufgeführt werden. Insgesamt lassen sich dann beispielsweise die konzipierten PL/pgSQL-Prozeduren `clusterRel` und `classify` wie folgt aufrufen:

```
/* Typischer Aufruf von PL/pgSQL-Prozeduren */
SELECT * FROM "clusterRel"('rel','{a1,a2,a3}','{5,3,1}');
SELECT * FROM "classify"('rel','{a1,a2,a3}',5,11);
```

Listing 5.2: Aufruf von PL/pgSQL-Prozeduren für die Analyse von CAM

`'rel'` ist dabei der Name einer existenten Tabelle bzw. Sicht und `'{a1,a2,a3}'` die Liste der zu analysierenden Attribute aus dieser reellen oder virtuellen Relation. Der Vektor `'{5,3,1}'` repräsentiert die subjektiven Gewichtungen der Attribute, die für das hierarchische Clustering aus Kapitel 4.5.2 notwendig sind, während die Werte `5` und `11` das numerischen Vorwissen des Klassifikators h aus Kapitel 4.4.1 darstellen.

Die Ausgabe der konzipierten Algorithmen aus den Kapiteln 4.3 bis 4.6 ist darauf ausgelegt, eine neue Relation zu erzeugen, die bestimmte Muster der analysierten Daten enthält. Dies lässt sich in PL/pgSQL, wie im Programmbeispiel von Kapitel 5.1.1 beschrieben, durch das Schlüsselwort `SETOF` gefolgt von einem spezifizierten Datentyp[12] in der Funktionssignatur angeben. Eine detaillierte Schilderung der definierten Typen findet sich in den jeweiligen Implementierungskapiteln der Algorithmen.

5.2 Umsetzung von Basiskonzepten aus der Rough Set Theory

Dieses Kapitel beschäftigt sich mit der Umsetzung der Ununterscheidbarkeitsrelation als fundamentales Instrument für die vorgeschlagenen Analysealgorithmen dieser Arbeit. Zusätzlich werden in diesem Kapitel Möglichkeiten aufgezeigt, wie die Approximation von Zielkonzepten und das Finden von Redukten mit relationalen Operationen in PL/pgSQL implementiert werden können.

5.2.1 Repräsentation der Ununterscheidbarkeit mit PL/pgSQL

Der algebraische Ausdruck 4.4 aus Kapitel 4.3 wird dazu genutzt, jeweils einen Repräsentanten aus jeder Äquivalenzklasse der aufgespannten Partitionierung zu

[12] Es lassen sich auch neben den standardmäßigen Datentypen wie `INTEGER` oder `TEXT` auch komplexere benutzerdefinierte Typen oder Tabellen angeben.

extrahieren. Diese Sichtweise auf die Ununterscheidbarkeit von Objekten lässt sich in SQL durch eine einfache Duplikateneliminierung erwirken. Sie kann im Wesentlichen auf zwei Arten formuliert werden:

```
/* Konkrete Umsetzung des definierten Ausdrucks: represent */
/* Variante 1 mit DISTINCT: */
SELECT DISTINCT "a1","a2","a3" FROM "rel";
/* Variante 2 mit GROUP BY: */
SELECT "a1","a2","a3" FROM "rel" GROUP BY "a1","a2","a3";
```

Listing 5.3: Duplikateneliminierung für die Extraktion von Repräsentanten

Die zwei Statements aus Listing 5.3 sind von ihrer Semantik her identisch. Allgemein betrachtet entstammen jedoch beide Befehle unterschiedlichen Auffassungen. Während das Schlüsselwort `DISTINCT` ursprünglich für das Entfernen doppelter Tupel aus einer Relation eingeführt wurde, dient `GROUP BY` für die Gruppierung gleicher Tupel. Optional können bei `GROUP BY` Aggregierungsfunktionen auf die einzelnen Gruppen angewandt werden, was durch `DISTINCT` nicht unterstützt wird. Trotz ihrer syntaktischen Unterschiedlichkeit benötigen beide SQL-Statements für eine konkrete Ausführung den Namen einer Relation (hier `rel`) sowie die Attributmenge, auf die projiziert bzw. gruppiert werden soll (hier `a1`, `a2`, `a3`). Nach erfolgreicher Ausführung ist die Ausgabe dann eine Relation ohne Duplikate, reduziert auf die angegebenen Attribute `a1`, `a2`, `a3`. Problematisch ist dies zum einen, weil nichtbetrachtete Attributinformationen aus der originalen Relation `rel` verloren gehen und zum anderen die Information über die Anzahl gleicher Tupel nicht berücksichtigt wird. Während erstere Problematik nicht ohne weiteres mit SQL abbildbar ist, lässt sich der zweite Umstand durch folgendes Konstrukt lösen:

```
/* Konkrete Umsetzung des definierten Ausdrucks: ind */
SELECT count(*) AS card, "a1","a2","a3" FROM "rel"
  GROUP BY "a1","a2","a3";
```

Listing 5.4: Repräsentation der Ununterscheidbarkeit mit SQL

Hierbei findet eine Gruppierung der Tupel statt, die in den Attributen `a1`, `a2`, `a3` übereinstimmen. Die Anzahl der Tupel der einzelnen Gruppen wird dann mit der Aggregatfunktion `count(*)` ermittelt. Die Ergebnisrelation, die aus diesem Ausdruck resultiert, besteht genau aus einem Stellvertreter jeder Gruppe in Verbindung mit einem neuen Attribut (hier `card`), welches die Information über die Anzahl der Tupel der jeweiligen Gruppe enthält. Durch dieses SQL-Kommando wird die Semantik von Listing 5.3 mit der zusätzlichen Information der Kardinalität einer Gruppe erweitert.

Im Zuge einer generellen Anwendung der illustrierten SQL-Statements auf CAM-Daten ist ersichtlich, dass für beliebige Relationen die Ausdrücke aus Listing 5.3 und 5.4 im Hinblick auf die Anzahl und Wertigkeit der Attribute stets unterschiedliche Ausgaben generieren. Dieser Umstand ist zunächst kritsich für das Umsetzen der Befehle in PL/pgSQL-Prozeduren, denn jede Prozedur benötigt zur Compile-Zeit einen definierten Rückgabetyp (vgl. Kapitel 5.1.1), der nicht alleine durch die Befehle suggeriert werden kann. Betrachtet man allerdings die Ergebnisrelationen für die Kommandos aus Listing 5.3, fällt auf, dass jedes Tupel vereinfacht formuliert eine Liste von Attributwerten darstellt. Ähnliches gilt für die Ausgabe des Befehls aus Listing 5.4. Hier enthält jedes Tupel einen numerischen Wert, gefolgt von einer Liste von Attributwerten. Diese Erkenntnis lässt sich nutzen, um einheitliche benutzerdefinierte Typen für die jeweilige Prozedur zu konzipieren. Die folgende Anweisung zeigt dazu am Beispiel des Befehls aus Listing 5.4, wie ein solcher Typ aussehen kann:

```
/* Rückgabetyp für die Anwendung des GROUP BY-Befehls */
CREATE TYPE RS_TYPE AS
(
  card INTEGER,
  values TEXT[]
);
```

Listing 5.5: Rückgabetyp für die generische Verwendung des `GROUP BY`-Befehls

Es kann nun die Formulierung der generischen PL/pgSQL-Methoden `represent` und `ind` folgen. Die Prozedur `represent` kapselt hierbei einen der beiden Ausdrücke in Listing 5.3 und `ind` den Befehl aus 5.4. Da sich die Struktur der beiden Funktionen im Wesentlichen gleichen wird an dieser Stelle lediglich die Syntax von `ind` näher beschrieben:

```
/* PL/pgSQL-Implementierung für den algebraischen Ausdruck: ind */
CREATE FUNCTION ind(relation TEXT, attList TEXT[])
  RETURNS SETOF RS_TYPE AS
$BODY$
DECLARE
  attsConcat TEXT :='"';
  query TEXT :='';
  result RS_TYPE;
BEGIN
  /* Aufbereitung von attList */
  attsConcat := attsConcat || ARRAY_TO_STRING (attList, '","');
  attsConcat := attsConcat || '"';

  /* Zusammensetzung des Query-Strings */
  query := 'SELECT count(*) as card, ARRAY[' || attsConcat || '] AS values ' ||
     'FROM "' || relation || '" GROUP BY ' || attsConcat;
```

```
  /* Ausführung des zusammengesetzten Query-Strings */
  FOR result IN EXECUTE query LOOP
    RETURN NEXT result;
  END LOOP;
END
$BODY$
  LANGUAGE 'plpgsql' STABLE;
```

Listing 5.6: Kapselung des GROUP BY-Befehls in eine PL/pgSQL-Prozedur

Die Abarbeitung der PL/pgSQL-Methode aus Listing 5.6 ist dabei in drei Phasen eingeteilt. Zunächst wird die übergebene Attributliste `attList` in einen String umgewandelt. Ist beispielsweise `attList:='{a1,a2,a3}'`, so entsteht für die Generierung der eigentlichen Anfrage die neue Attributliste `attsConcat`, welche den String-Wert `'"a1","a2","a3"'` enthält. In der nächsten Phase wird über den Verknüpfungsoperator `||` und den gegebenen Parametern `relation` bzw. `attsConcat` das `GROUP BY`-Statement in Form eines Strings erzeugt. Im finalen Schritt wird letztlich die neue Anfrage ausgeführt, über das Ergebnis iteriert und dieses gekapselt im Typ `RS_TYPE` ausgegeben.

5.2.2 Grobe Darstellung von Konzepten mittels SQL

Um ein Zielkonzept ausschließlich mit den gegebenen Informationen einer Relation zu beschreiben, wird in der RST die Ununterscheidbarkeitsrelation verwendet. Im Folgenden wird darauf basierend die Umsetzung der untere und obere Approximation sowie der Grenz- und Außenregion durch geeignete SQL-Konstrukte erläutert. Konkret sei dazu eine Tabelle `rel` und eine Untermenge `rel_Concept` aus `rel` mit den jeweiligen Attributen `a1`, `a2`, `a3` betrachtet. Die konzipierten relationalen Ausdrücke 4.6, 4.7, 4.8 und 4.9 lassen sich dann wie folgt in SQL angeben:

```
/* Konkrete Umsetzung der Ausdrücke für die Konzeptapproximation aus der
   Rough Set Theory */
/* Untere Approximation: */
SELECT count(*) AS card, "a1","a2","a3" FROM "rel"
  GROUP BY "a1","a2","a3"
INTERSECT
SELECT count(*) AS card, "a1","a2","a3" FROM "rel_Concept"
  GROUP BY "a1","a2","a3";

/* Obere Approximation: */
SELECT count(*) AS card, q."a1",q."a2",q."a3" FROM
(
  SELECT "a1","a2","a3" FROM "rel_Concept"
    GROUP BY "a1","a2","a3"
) AS q
```

```
JOIN "rel" p ON p."a1" = q."a1" AND p."a2" = q."a2"
  AND p."a3" = q."a3"
GROUP BY q."a1",q."a2",q."a3";

/* Grenzregion: */
(
  SELECT count(*) AS card, q."a1",q."a2",q."a3" FROM
  (
    SELECT "a1","a2","a3" FROM "rel_Concept"
      GROUP BY "a1","a2","a3"
  ) AS q
  JOIN "rel" p ON p."a1" = q."a1" AND p."a2" = q."a2"
    AND p."a3" = q."a3"
  GROUP BY q."a1",q."a2",q."a3"
)
EXCEPT
(
  SELECT count(*) AS card, "a1","a2","a3" FROM "rel"
    GROUP BY "a1","a2","a3"
  INTERSECT
  SELECT count(*) AS card, "a1","a2","a3" FROM "rel_Concept"
    GROUP BY "a1","a2","a3"
);

/* Außenregion: */
SELECT count(*) AS card, q."a1",q."a2",q."a3" FROM
(
  SELECT "a1", "a2", "a3" FROM "rel" GROUP BY "a1", "a2", "a3"
  EXCEPT
  SELECT "a1", "a2", "a3" FROM "rel_Concept"
    GROUP BY "a1", "a2", "a3"
) AS q
JOIN "rel" p ON p."a1" = q."a1" AND p."a2" = q."a2" AND
  p."a3" = q."a3"
GROUP BY q."a1",q."a2",q."a3";
```

Listing 5.7: Abbildung der Konzeptapproximation mit konkreten SQL-Befehlen

Es ist leicht zu erkennen, dass diese konkret angegebenen Befehle sich analog zu der Prozedur `ind` in Listing 5.6 mit PL/pgSQL formulieren lassen. Es entstehen dann die Funktionen `lower`, `upper`, `boundary` und `outside`, welche in PostgreSQL aufrufbar sind (vgl. Abbildung 5.2). Die Rückgabe dieser vier Methoden ist dann stets der Art `SETOF RS_TYPE`.

Problematisch bei der Anwendung der gegebenen SQL-Statements aus Listing 5.7 ist jedoch der praktische Umgang mit Nullwerten. Sie werden von dieser Implementierung nicht direkt unterstützt, da zwei Nullmarken nicht ohne weiteres miteinander verglichen werden können[13]. Lassen sich dennoch solche Werte nicht vermeiden, muss eine Präprozessierung auf den Daten durchgeführt werden. Ein vorgeschlagener

[13] Dies wird in den Join-Bedingungen deutlich, denn zwei Objekte x, y für die gilt: $a(x) = \perp = a(y)$ muss nicht zwangsläufig $a(x) = a(y)$ gelten.

Lösungsansatz findet sich z.B. in Kapitel 4.2, wo Platzhaltersymbole für Nullmarken eingeführt werden. Andere Möglichkeiten zur Behandlung von Nullwerten lassen sich in [Kan02, GBGB05, HSS07] nachschlagen.

5.2.3 Algorithmus zum Finden von Redukten

Das Finden von Redukten wurde bereits mit dem Verfahren *reduct_search* aus Kapitel 4.3.3 beschrieben. In diesem Zusammenhang wurde die exponentielle Suche durch den Feature-Space dadurch motiviert, dass zunächst auf eine einfache Weise der *Core* anhand von Algorithmus *core_search* ermittelt wird und somit der Suchraum zum Finden von Redukten massiv eingeschränkt werden kann. Das folgende Listing 5.8 beschreibt dazu die Umsetzung von *core_search* in der PL/pgSQL-Methode `core_search`:

```
/* PL/pgSQL-Implementierung für den Algorithmus core_search */
CREATE FUNCTION core_search(relation TEXT, attList TEXT[])
  RETURNS SETOF TEXT AS
$BODY$
DECLARE
  attsConcatAll TEXT := '"'; actualAtts TEXT[] := attList;
  actualAttsConcat TEXT;
  attsSize INTEGER := array_upper (attList, 1); counter INTEGER := 1;
  ref INTEGER := 0; proof INTEGER := 0;
BEGIN
  /* Aufbereitung von attList */
  attsConcatAll := attsConcatAll ||
    ARRAY_TO_STRING (attList, '","');
  attsConcatAll := attsConcatAll || '"';

  /* Generierung und Ausführen der Referenz-Query */
  FOR ref IN EXECUTE 'SELECT COUNT(*) FROM "represent"(''' ||
    relation || ''',''{' || attsConcatAll || '}'')' LOOP
  END LOOP;

  /* Überprüfung aller Attribute*/
  WHILE counter <= attsSize LOOP
    /* Entfernen eines Attributs */
    actualAtts := attList[1:counter-1] ||
      attList[counter+1:attsSize];
    actualAttsConcat := '"';
    actualAttsConcat := actualAttsConcat ||
      ARRAY_TO_STRING (actualAtts, '","');
    actualAttsConcat := actualAttsConcat || '"';

    /* Generierung und Ausführen der zu prüfenden Query */
    FOR proof IN EXECUTE 'SELECT COUNT(*) FROM "represent"(''' ||
      relation || ''',''{' || actual_attsConcat || '}'')' LOOP
    END LOOP;

    /* Bedingungsüberprüfung auf Core-Attribut */
```

```
    IF ref != proof THEN
      RETURN NEXT attList[counter]; /* Ausgabe des Attributs */
    END IF;
    counter := counter + 1;
  END LOOP;
END
$BODY$
  LANGUAGE 'plpgsql' STABLE;
```

Listing 5.8: Ermittlung des *Core* mit PL/pgSQL

Der Basisalgorithmus für die Reduktsuche ist nun in PL/pgSQL formuliert. Dennoch werden der Verständlichkeit halber zwei weitere Hilfsfunktionen für die finale Betrachtung der Implementierung von *reduct_search* durch `reduct_search` benötigt. Sie lauten `pot` und `concat_to_text`. `pot` ist dabei eine Methode, die anhand eines übergebenen String-Felds die Potenzmenge in Form einer Relation (`SETOF TEXT []`) berechnet. Auf der anderen Seite wird mit `concat_to_text` ein übergebenes String-Feld durch bestimmte String-Verknüpfungen in einen normalen String umgewandelt. Nachdem alle Komponenten für `reduct_search` bekannt sind, folgt eine Betrachtung dessen in Form ausgewählter Code-Segmente:

```
/* PL/pgSQL-Implementierung für den Algorithmus reduct_search */
CREATE FUNCTION reduct_search(relation TEXT, attList TEXT[])
  RETURNS SETOF TEXT[] AS
$BODY$
DECLARE
  /* Variablendeklaration für die Attributrepräsentation */
  allAttsSize INT := array_upper (attList, 1);
  allAttsConcat TEXT := '';
  coreAtts TEXT[] := '{}'; coreAttsConcat TEXT := '';
  diffAtts TEXT[] := '{}'; diffAttsConcat TEXT := '';
  potAtts TEXT[] := '{}'; potAttsConcat TEXT := '';
  mininalAtts TEXT[] := '{}'; minimalAttsConcat TEXT := '';

  /* Weitere Variablendeklarationen */
  attResult TEXT := ''; tmpTable TEXT := ''; reductAtts TEXT[];
  ref INT := 0; proof INT := 0; minimalProof INT := 0;
  counter INT := 1; reductCounter INT := 0;
BEGIN
  /* Erstellung zweier temporärer Tabellen bestehend aus einer
     Spalte mit dem Typ TEXT. In die eine Tabelle werden dann die
     zu analysierenden Attribute und in die andere Tabelle der
     berechnete Core gespeichert. Die Namen der Tabellen ergeben
     sich aus einer Zufallszahl tmpTable gefolgt von dem Postfix
     _All respektive _Core. Zusätzlich liegen die Attribute
     jeweils redundant in den String-Feldern allAtts bzw. coreAtts
     vor. */
  ...

  /* Differenzbildung zwischen den Mengen All und Core */
  FOR attResult IN EXECUTE 'SELECT * FROM "' || tmpTable ||
    '_All" EXCEPT SELECT * FROM "' || tmpTable || '_Core"' LOOP
```

```
    diffAtts[counter] := attResult;
    counter := counter + 1;
  END LOOP;

  /* Prüfung, ob es sich beim Core schon um ein Redukt handelt */
  IF array_upper (diffAtts, 1) is NULL THEN
    coreAttsConcat := "concat_to_text"(coreAtts);
    FOR reductAtts IN EXECUTE 'SELECT ''{' || coreAttsConcat ||
      '}''::TEXT[]' LOOP
    END LOOP;
    RETURN NEXT reductAtts; /* Ausgabe des Core als Redukt */
    RETURN; /* Vorläufige Beendigung der Abarbeitung */
  END IF;
  diffAttsConcat := "concat_to_text"(diffAtts);

  /* Berechnung der Anzahl der Äquivalenzklassen mit allen Attributen */
  FOR ref IN EXECUTE 'SELECT COUNT(*) FROM "represent"(''' || relation ||
     ''',''{' || allAttsConcat || '}'')' LOOP
  END LOOP;

  /* Exponentielle Suche nach Redukt auf der Menge: Core + pot(diffAtts) */
  FOR potAtts IN EXECUTE 'SELECT * FROM "pot"(''{' ||
    diffAttsConcat || '}'')' LOOP
    potAttsConcat := "concat_to_text"(potAtts || coreAtts);
    reductCounter := 0;

    FOR proof IN EXECUTE 'SELECT COUNT(*) FROM "represent"(''' || relation ||
       ''',''{' || potAttsConcat || '}'')' LOOP
    END LOOP;

    /* Prüfung, ob Core + pot(diffAtts) ein Redukt sein kann */
    IF ref = proof THEN
      /* In diesem Abschnitt wird geprüft, ob eines der Attribute in potAtts
         entfernt werden kann. Ist dies der Fall wird die Abarbeitung dieser
         Iteration beendet. Andernfalls wird die Variable reductCounter
         inkrementiert. */
      ...

      /* Überprüfung, ob reductCounter der Kardinalität von potAtts entspricht */
      IF reductCounter = array_upper(potAtts, 1) THEN
        minimalAttsConcat := "concat_to_text"(potAtts ||coreAtts);
        FOR reductAtts IN EXECUTE 'SELECT ''{' || minimalAttsConcat ||
           '}''::TEXT[]' LOOP
        END LOOP;
        RETURN NEXT reductAtts; /* Ausgabe des Redukts */
        RETURN; /* Durch Löschen dieser Zeile werden alle existenten Redukte
                   ausgegeben => Beachte: Rechenaufwand! */
      END IF;
    END IF;
  END LOOP;
  RETURN;
END
$BODY$
  LANGUAGE 'plpgsql' VOLATILE; /* Muss gesetzt werden, weil diese
                                  Methode Tabellen erzeugt. */
```

Listing 5.9: Bestimmung von Redukten mit PL/pgSQL

5.3 Labeln von Relationen auf Basis des groben Klassifikators

Das Vorgehen, das Benutzeraktionen in Klassen einordnet, wurde in Kapitel 4.4 durch den Rough-Set-Klassifikator h erläutert. Dabei findet die Zuordnung auf Basis des Vorwisses $\alpha, \beta \in \mathbb{N}$ in die vordefinierten Klassen $K_{h=1}, K_{h=?}, K_{h=0}$ statt, die sich zusätzlich als Approximation des unbekannten Zielkonzepts in den Daten interpretieren lassen. Im Hinblick auf die Implementierung dieser Klassifikation durch die Prozedur `classify` lässt sich die Eigenschaft nutzen, dass $K_{h=1}, K_{h=?}$ und $K_{h=0}$ eine Partition $\{K_{h=1}, K_{h=?}, K_{h=0}\}$ über den zu klassifizierenden Daten aufgespannt, die wiederum in einer Relation darstellbar ist. Dazu kann man sich vorstellen, dass der benutzerdefinierte Typ `LABELED_RS_TYPE` in Listing 5.10 jeweils ein Tupel dieser Relation repräsentieren kann:

```
/* Rückgabetyp für Zuordnung von Aktionen */
CREATE TYPE LABELED_RS_TYPE AS
(
  card INTEGER,
  values TEXT[],
  tag TEXT
);
```

Listing 5.10: Typ für die Zuordnung von Repräsentanten in Klassen

Der Typ `LABELED_RS_TYPE` stellt eine Spezialisierung von `RS_Type` aus Listing 5.5 dar. Begründung findet diese Konstellation in der Tatsache, dass die grobe Klassifikation aus der Entwurfsphase auf dem algebraischen Ausdruck 4.5 basiert und somit die bereits vorgestellte PL/pgSQL-Prozedur `ind` aus Listing 5.6 als Grundlage für die praktische Umsetzung der Klassifizierung fungieren kann. Zusätzlich kapselt `LABELED_RS_TYPE` ein neues Attribut `tag`, welches die Klassenzugehörigkeit h eines jeden Repräsentanten aus $K_{h=1} \cup K_{h=?} \cup K_{h=0}$ symbolisiert.

Nachdem nun der Typ `LABELED_RS_TYPE` als mögliche Rückgabe für die Methode `classifiy` diskutiert wurde, kann nun das eigentliche Vorgehen der Umsetzung erläutert werden. Die Grundidee dazu ist das Labeln[14] von `ind`, angewandt auf eine Ausgangsrelation `relation` mit den Attributen `attList` und den zwei numerischen Parametern `alpha` und `beta`. Zu diesem Zweck wird zunächst eine Anfrage für die generische Ausführung der Prozedur `ind` erzeugt. Mittels des `FOR` ... `IN EXECUTE` ...

[14] Damit ist gemeint, dass jedes Tupel mit einer Zuordnungsinformation (hier $1, ?$ oder 0) angereichert wird.

Konstrukts wird dann über das Ergebnis der Anfrage iteriert und für jedes Tupel anhand des Vorwissens entschieden, in welche Klasse es einzuordnen ist. Nachvollzogen werden kann dieses Verfahren anhand von Listing 5.11:

```
/* PL/pgSQL-Implementierung für die grobe Klassifikation */
CREATE FUNCTION classify(relation TEXT, attList TEXT[],
  beta INTEGER, alpha INTEGER)
  RETURNS SETOF LABELED_RS_TYPE AS
$BODY$
DECLARE
  attListConcat TEXT; result LABELED_RS_TYPE;
BEGIN
  /* Prüfung, ob alpha größer als beta ist */
  IF beta >= alpha THEN
    RAISE EXCEPTION 'The parameter alpha (%) must be greater than
      beta (%)',alpha,beta;
  END IF;
  attListConcat := "concat_to_text"(AttList);

  /* Iteration über Ergebnisrelation */
  FOR result IN EXECUTE 'SELECT * FROM "ind"(''' || relation || ''',''{' ||
     attListConcat || '}'')' LOOP
    /* Zuordnen bzw. Labeln der Äquivalenzklassen */
    IF (result.card > alpha) THEN
      result.tag := '1';
    ELSIF (result.card < beta) THEN
      result.tag := '0';
    ELSE
      result.tag := '?';
    END IF;
    RETURN NEXT result; /* Ausgabe der gelabelten Klassen */
  END LOOP;
END
$BODY$
  LANGUAGE 'plpgsql' STABLE;
```

Listing 5.11: Zuordnung von äquivalenten Objekten in drei Klassen

Es sei darauf hingewiesen, dass das Klassifizieren bzw. Labeln von Repräsentanten aus der an `classify` übergebenen Tabelle (z.B. `relation := 'toTest'`) mit dem gegebenen Vorwissen und der Ununterscheidbarkeit lokal gesehen keinen Fehler besitzt. Ist die Tabelle `toTest` aus globaler Perspektive jedoch nur eine Teilmenge eines größeren Datenbestandes (z.B. Tabelle `global`), so können Fehlklassifizierungen durch `classify` auftreten (vgl. Kapitel 4.4.2). Der Fehler, also die Menge an Repräsentanten, die `classify` nicht korrekt labelt, lässt sich dann quantitativ durch folgende Berechnung angeben, wenn `a1`, `a2`, `a2` die zu analysierenden Attribute aus `toTest` bzw. `global` und `alpha := 5`, `beta := 3` das Vorwissen bilden:

```
/* Fehler der Klassifikation durch Differenzbildung */
SELECT R.card, R.values FROM
```

```
(
  SELECT * FROM "classify"('toTest','{a1,a2,a3}',3,5)
  EXCEPT
  SELECT * FROM "classify"('global', '{a1,a2,a3}',3,5)
) AS errSet
JOIN "classify"('toTest','{a1,a2,a3}',3,5) R ON
  R.values = errSet.values AND R.tag = errSet.tag
```

Listing 5.12: Tatsächlicher Fehler von `classify` am konkreten Beispiel

Der Fehler lässt sich mengentheoretisch so interpretieren, dass im Schnitt der beiden gelabelten Relationen `classify('toTest', ... )` und `classify('global', ... )` genau solche Tupel enthalten sind, die korrekt von `classify('toTest', ... )` zugeordnet wurden. Dies trifft zu, da `classify('global', ... )` als globale und damit korrekte Referenz für `classify('toTest', ... )` fungiert. Demzufolge müssen sich in der Differenzmenge von `classify('toTest', ... )` und `classify('global', ... )` alle gelabelten Tupel befinden, die fehlerhaft eingeschätzt wurden, was exakt der Semantik von Listing 5.12 entspricht. Die Join-Operation wird zusätzlich angewandt, damit die Information der Kardinalität nach der Differenzbildung nicht verloren geht. Somit stellt Listing 5.12 die praktische Umsetzung der Menge *errSet* aus Formel 4.14 in Kapitel 4.4.2 für ein konkretes Beispiel dar.

5.4 Clustering von Benutzeraktivitäten

In diesem Abschnitt werden wichtige Aspekte des hierarchischen Clustering aus Kapitel 4.5.2 im Hinblick auf die Erstellung der Prozedur `clusterRel` konkretisiert. `clusterRel` ist dabei die PL/pgSQL-Implementierung des konzipierten Algorithmus *clusterRel*. Die Berechnung des gewichteten Distanzmaßes wird dazu in diesem Kapitel durch die Prozedur `wDistance` erläutert. Gleichermaßen ist die Distanzmatrix und deren Aktualisierung im Verlaufe von `clusterRel` wichtig. Ihr Aufbau wird anhand der Methode `distanceMatrix` beleuchtet, während die Aktualisierungsphase der Matrix durch ausgewählte Code-Fragmente aus `clusterRel` illustriert wird.

5.4.1 Attributgewichtete Berechnung von Distanzen

Der vorgeschlagene agglomerative Ansatz geht von der initialen Partitionierung durch die Ununterscheidbarkeitsrelation aus. Das bedeutet, dass `ind` als mögliche PL/pgSQL-Implementierung der Ununterscheidbarkeit (vgl. Kapitel 5.2.1) die Basis für das verschachtelte Gruppieren von Aktivitäten liefern kann. Dies eröffnet

zusätzlich die Möglichkeit den Rückgabetyp `rs_type` von `ind` für die Berechnungen der konzipierten Distanzen aus Kapitel 4.5.2 zwischen Aktivitäten zu nutzen. Veranschaulicht werden kann dies anhand des Attribut `values` für jedes Tupel aus `ind`, denn zur Ermittlung einer Distanz müssen lediglich alle Attributwerte der einen Aktivität mit denen der Anderen verglichen werden. Das folgende Listing 5.13 zeigt dazu eine Umsetzungsmöglichkeit am Beispiel des gewichteten Abstands in Form der Prozedur `wDistance`. Als Eingabeparameter erwartet `wDistance` die Feld-Variablen `valuesClass_1` und `valuesClass_2`, welche die Attributwerte zweier zu vergleichender Tupel aus `ind` symbolisieren sowie `attWeights`, dem Gewichtungsvektor, der jedem Attribut sein subjektives Gewicht zuordnet:

```
/* PL/pgSQL-Implementierung der gewichteten Distanz: wd */
CREATE FUNCTION wDistance(valuesClass_1 TEXT[],
  valuesClass_2 TEXT[], attWeights INTEGER[])
  RETURNS INTEGER AS
$BODY$
DECLARE
  /* Anzahl der Attribute in den zwei übergebenen Tupel */
  countValues1 INTEGER := array_upper (valuesClass_1, 1);
  countValues2 INTEGER := array_upper (valuesClass_2, 1);
  counter INTEGER := 1; /* Iterationszähler */
  weightSum INTEGER := 0; /* gewichtete Summe */
  nDistance INTEGER := 0; /* natürliche Distanz */
BEGIN
  /* An dieser Stelle wird überprüft, ob die zu vergleichenden
     Tupel valuesClass_1 und valuesClass_1 die gleiche Anzahl an
     Attributen besitzen. Ist dies nicht der Fall, so wird die
     Abarbeitung beendet. Gleiches wird mit der Länge von
     attWeights durchgeführt. */
  ...

  /* Vergleich der Attributwerte */
  WHILE counter <= countValues1 LOOP
    /* Berechnung der natürlichen Distanz und Summierung von
       Gewichten */
    IF valuesClass_1[counter] != valuesClass_2[counter] THEN
      nDistance := nDistance + 1;
      weightSum := weightSum + attWeights[counter];
    END IF;
    counter := counter + 1;
  END LOOP;
  RETURN nDistance * (1 + weightSum); /* Ausgabe der Distanz */
END
$BODY$
  LANGUAGE 'plpgsql' STABLE;
```

Listing 5.13: Gewichtete Distanz zwischen Aktivitäten in PL/pgSQL

5.4.2 Erzeugung der Distanzmatrix

Wendet man nun die Prozedur `wDistance` auf alle sinnvollen Tupelkombinationen aus `ind` an, so entsteht eine Distanzmatrix[15]. Mit sinnvoll ist hier gemeint, dass es ausreicht, die Distanz zwischen zwei Aktionen nur einmal zu berechnen. Dies lässt sich mit zwei ineinander geschachtelten Schleifen realisieren. Die äußere Schleife iteriert dabei über `ind` und speichert jedes Tupel in einer temporären Relation, während die innere Schleife die Differenzmenge zwischen `ind` und der temporären Relation durchläuft. Es werden dann die gewichteten Distanzen zwischen Tupeln der inneren und äußeren Schleife gebildet und als Ergebnis ausgegeben. Listing 5.14 beschreibt dazu das Vorgehen in der Methode `distanceMatrix`:

```
/* PL/pgSQL-Implementierung der Distanzmatrix */
CREATE FUNCTION distanceMatrix(relation TEXT,
  attList TEXT[], attWeights INTEGER[])
  RETURNS SETOF DM_TYPE AS
$BODY$
DECLARE
  tmpTable TEXT := '';
  recOuter RS_TYPE;   recInner RS_TYPE;
  result DM_TYPE;
BEGIN
  /* An dieser Stelle werden zwei leere temporäre Tabellen
     erstellt. Die Namen der Tabellen ergeben sich aus einer
     Zufallszahl tmpTable gefolgt von dem Postfix _Ignore und
     _ToScan. Sie halten Daten vom Typ RS_TYPE. */
  ...

  /* Speichern von ind in der Menge _ToScan */
  FOR recInner IN EXECUTE 'SELECT * FROM "ind"(''' || relation ||
    ''',''{' || "concat_to_text"(attList) || '}'')' LOOP
    EXECUTE 'INSERT INTO "' || tmpTable || '_ToScan" VALUES (' || recInner.card ||
       ',''{' || "concat_to_text"(recInner.values) || '}'')';
  END LOOP;

  /* Erstellung der Distanzmatrix */
  FOR recOuter IN EXECUTE 'SELECT * FROM "' || tmpTable || '_ToScan"' LOOP
    EXECUTE 'INSERT INTO "' || tmpTable || '_Ignore" VALUES (' || recOuter.card ||
       ',''{' || "concat_to_text"(recOuter.values) || '}'')';
    FOR recInner IN EXECUTE 'SELECT * FROM "' || tmpTable ||
       '_ToScan"' || ' EXCEPT ' || 'SELECT * FROM "' || tmpTable ||
       '_Ignore"' LOOP

      /* Kapselung der Ausgabe im Typ DM_TYPE */
      result.c1 := recOuter.values;
      result.c2 := recInner.values;
      result.cardC1 := recOuter.card;
      result.cardC2 := recInner.card;
      result.wd := "w_distance"(recOuter.values, recInner.values, attWeights);
```

[15] Grafisch betrachtet, repräsentiert die Distanzmatrix die Kanten des vollständigen Distanzgraphen (vgl. Beispiel 4.6).

```
      RETURN NEXT result; /* Ausgabe */
    END LOOP;
  END LOOP;
END
$BODY$
  LANGUAGE 'plpgsql' VOLATILE;
```

Listing 5.14: Generierung der Distanzmatrix als Relation

Die Ausgabe von `distanceMatrix` ist eine Menge vom Typ `DM_TYPE`. Dieser Typ besteht aus zwei String-Feldern `c1` und `c2`, welche jeweils zwei voneinander unterschiedliche Benutzeraktivitäten repräsentieren[16], deren Kardinalität `cardC1`, `cardC2` sowie die gewichtete Distanz `wd` zwischen `c1` und `c2`.

5.4.3 Aktualisierung der Distanzmatrix

In jeder Iteration von `clusterRel`, werden die zwei Cluster mit dem kleinsten gewichteten Abstand verschmolzen. Dieser Vorgang impliziert eine Aktualisierung der Distanzmatrix, denn das gefundene Tupel muss aus der Matrix entfernt werden und alle anderen Tupel, die in Korrelation mit einem der zusammengefügten Cluster stehen, müssen eine neue Zuordnung erhalten. U.a. können aus dieser Aktualisierung auch Duplikate resultieren, die ebenfalls entfernt werden müssen. Das beschriebene Vorgehen wird nun innerhalb von `clusterRel` am Beispiel einer Iteration erläutert.

Ausgangssituation ist dabei die von `clusterRel` bzw. `distanceMatrix` erzeugte Distanzmatrix, aus der bereits das Tupel `minRec` mit dem kleinsten gewichteten Abstand `wd` gefunden und entfernt wurde. Es muss nun die Teilmenge der Einträge aus der Matrix aktualisiert werden, die im Attribut `c1` oder `c2` mit `minRec.c1` bzw. `minRec.c2` übereinstimmt. Dies kann adäquat durch folgende Schleife realisiert werden:

```
...
  /* Iteration über Tupel aus der Distanzmatrix, die von der
     Fusion betroffen sind. Es werden numerische Platzhalter an
     den Stellen eingefügt, wo einer der Werte in c1 oder c2 dem
     gelöschten Tupel ursprünglich entsprach und die Kardinali-
     täten aktualisiert. */
  FOR dmRec IN EXECUTE 'SELECT * FROM "' || tmpTable || '_DM" WHERE c1 = ''' ||
     minRec.c1 || ''' OR c2 = ''' || minRec.c1 || ''' OR c1 = ''' || minRec.c2 ||
     ''' OR c2 = ''' || minRec.c2 || '''' LOOP
```

[16] vgl. Variable `values` aus der Rückgabe von `ind` in Listing 5.5

```
    /* Aktualisierung */
    IF dmRec.c1 = minRec.c1 OR dmRec.c1 = minRec.c2 THEN
      EXECUTE 'UPDATE "' || tmpTable || '_DM" SET c1 = ''' || clusterId ||
         ''', cardC1 = ' || minRec.cardC1 + minRec.cardC2 || ' WHERE c1 = ''' ||
         dmRec.c1 || ''' AND c2 = ''' || dmRec.c2 || '''';
    ELSIF dmRec.c2 = minRec.c1 OR dmRec.c2 = minRec.c2 THEN
      EXECUTE 'UPDATE "' || tmpTable || '_DM" SET c2 = ''' || clusterId ||
         ''', cardC2 = ' || minRec.cardC1 + minRec.cardC2 || ' WHERE c1 = ''' ||
         dmRec.c1 || ''' AND c2 = ''' || dmRec.c2 || '''';
    END IF;
  END LOOP;
...
```

Listing 5.15: Erster Aktualisierungsschritt der Distanzmatrix in `clusterRel`

Die Aktualisierung der Matrix ist nach dem Ausführen von Listing 5.15 jedoch noch nicht abgeschlossen, da noch redundante Verbindungen zu dem neu erstellten Cluster existieren, die entfernt werden müssen. Erschwerend bei der Entdeckung solcher Tupel ist, dass die durch Listing 5.15 betroffenen Tupel entweder in `c1` oder `c2` mit dem numerischen Platzhalter `clusterId`, als Repräsentant für das neue Cluster, versehen wurden. Somit können Verbindungen in der Distanzmatrix auftreten, welche die selbe Semantik besitzen, allerdings nicht ohne eine Normierung algebraisch entdeckt werden können. Zur Verdeutlichung seien zwei Tupel `x` und `y` aus der Matrix betrachtet mit $\texttt{x.c1} \neq \texttt{y.c1}$ und $\texttt{x.c2} \neq \texttt{y.c2}$, aber $\texttt{x.c1} = \texttt{y.c2}$ und $\texttt{x.c2} = \texttt{y.c1}$. `x` und `y` gleichen sich von ihrer Bedeutung, aber aus relationaler Pespektive sind sie unterschiedlich. Durch die in Listing 5.16 beschriebene Normalisierung werden potenziell gleiche Tupel vereinheitlicht:

```
...
  /* Normalisieren von ähnlichen Tupel in der Distanzmatrix */
  FOR dmRec IN EXECUTE 'SELECT * FROM "' || tmpTable || '_DM" ' ||
     'WHERE state = ''todo''' LOOP
      /* Markierung der Tupel, die nicht weiter betrachtet werden
         müssen. */
      EXECUTE 'UPDATE "' || tmpTable || '_DM" SET state = ''done'' ' ||
         'WHERE c1 = ''' || dmRec.c1 || ''' AND ' || 'c2 = ''' || dmRec.c2 ||
         ''' AND state = ''todo''';
      /* Vertauschen von c1 u. c2 bei den Tupel, die die selbe
         Semantik besitzen. */
      EXECUTE 'UPDATE "' || tmpTable || '_DM" SET c1 = ''' || dmRec.c1 ||
         ''', c2 = ''' || dmRec.c2 || ''', state =''cardToChange'' WHERE ' ||
         'c1 = ''' || dmRec.c2 || ''' AND c2 =''' || dmRec.c1 ||
         ''' AND state =''todo''';
  END LOOP;

  /* Vertauschen der Kardinalitätsinformationen bei den Tupel,
     die zuvor geändert wurden. */
  FOR dmRec IN EXECUTE 'SELECT * FROM "' || tmpTable || '_DM" ' ||
     'WHERE state = ''cardToChange''' LOOP
    IF dmRec.state = 'cardToChange' THEN
```

```
      EXECUTE 'UPDATE "' || tmpTable || '_DM" SET state = ''done'', cardC1 = ' ||
         dmRec.cardC2 || ', cardC2 = ' || dmRec.cardC1 || ' WHERE c1 = ''' ||
         dmRec.c1 || ''' AND c2 = ''' || dmRec.c2 || ''' AND state = ' ||
         '''cardToChange''';
    END IF;
  END LOOP;
...
```

Listing 5.16: Zweiter Aktualisierungsschritt der Distanzmatrix in `clusterRel`

Da nun die Tupel der Distanzmatrix vereinheitlicht vorliegen, können die Tupel nach `c1` und `c2` gruppiert und somit überflüssige Verbindungen aus der Matrix eliminiert werden. Durch Anwendung der Aggregatfunktion `max(wd)` auf jede Gruppe wird die Complete-Linkage- bzw. durch `min(wd)` die Single-Linkage-Strategie des Clustering erwirkt (vgl. Abbildung 4.17).

5.5 Implementierungsvarianten für Interessenvergleiche von Benutzern

Der in Kapitel 4.6 formulierte Ansatz basiert auf der Grenzregion der RST. Er geht davon aus, dass sich ähnliche Interessen zwischen Benutzern quantitativ durch Objekte in der Grenzregion aufdecken lassen und somit dem einen oder dem anderen Benutzer Empfehlungen ausgesprochen werden können, die sich bei der Arbeit am Computer positiv auswirken.

Aus technischer Sicht erinnert diese Idee sehr stark an die Schnittmengenbildung, weil ähnliche Aktionen, die beide Benutzer tätigen, in diese mengentheoretische Region zu fallen scheinen. Dennoch reicht der Schnittmengenoperator `INTERSECT` zur Beschreibung der Grenzregion in PostgreSQL nicht vollständig aus, denn im Gegensatz zum Schnitt enthält die Grenzregion alle Äquivalenzklassen, die die beiden aufgespannten Konzepte der Benutzer schneiden[17]. Auf der anderen Seite spiegelt der Schnitt, bildlich gesprochen, lediglich die Grenzregion mit Duplikateneliminierung wider.

Dennoch kann die Schnittmengenbildung dazu genutzt werden um den Algorithmus *equal_pref* aus Abbildung 4.26 mit SQL konkret zu beschreiben, wenn nach dem Schnitt alle ähnlichen Aktivitäten durch eine Join-Operation geladen werden und die Kardinalität des Ergebnis bestimmt wird. Dazu sei eine Relation `rel` mit den

[17] Genauer gesagt enthält die Grenzregion nur Klassen mit dieser Eigenschaft (vgl. Definition 2.10).

Attributen `user`, `a1`, `a2` zu betrachten, wobei die Größe der Grenzregion der beiden Konzepte `SELECT * FROM "rel" WHERE "user" = 'Max'` bzw. `SELECT * FROM "rel" WHERE "user" = 'Moritz'` anhand des Werts `5` zu evaluieren ist. Mit den zusammengesetzten SQL-Befehlen aus Listing 5.17 lässt sich dann *equal_pref* umsetzen:

```
/* SQL-Implementierung von equal_pref */
SELECT (5 <= count(*)) AS equal_pref FROM
(
  SELECT * FROM
  (
    SELECT "a1", "a2" FROM "rel" WHERE "user" = 'Max'
    INTERSECT
    SELECT "a1", "a2" FROM "rel" WHERE "user" = 'Moritz'
  ) AS r
  JOIN (SELECT * FROM "rel" WHERE "user" = 'Max' OR
    "user" = 'Moritz') s ON s."a1" = r."a1" AND s."a2" = r."a2"
) AS t
```

Listing 5.17: Erste Umsetzungsvariante für *equal_pref*

Die Ausgabe des Kommandos aus Listing 5.17 ist ein bool'scher Wert, der eine Aussage darüber trifft, ob die Kardinalität der Grenzregion einen gegebenen Wert (hier `5`) übersteigt.

Eine weitaus intuitivere Variante für die Implementierung von *equal_pref* bietet die PL/pgSQL-Prozedur `boundary` aus Kapitel 5.2.2. Aus Sicht der RST müssen hier nicht beide Konzepte separat betrachtet und geschnitten werden, sondern es reicht aus, nur eines zu betrachten (vgl. Kapitel 4.6). Allerdings darf für diesen Spezialfall zum einen die Tabelle `rel` nur Aktivitäten von `Max` oder `Moritz` umfassen und zum anderen akzeptiert die Funktionssignatur von `boundary` nur Relationsnamen und keine komplexen SQL-Anweisungen, was die flexible Verwendung der Methode in diesem Zusammenhang einschränkt. Das bedeutet, dass für die Implementierung von *equal_pref* durch `boundary` Tabellen oder Sichten im Vorhinein erzeugt werden müssen, auf die `boundary` zugreifen kann. Dies ist mit folgenden Anweisungen möglich:

```
/* Vorbereitung für die Anwendung von boundary */
CREATE VIEW relPartition AS
  SELECT "user", "a1", "a2" FROM "rel" WHERE "user" = 'Max' OR "user" = 'Moritz';

CREATE VIEW relConcept AS
  SELECT "user", "a1", "a2" FROM "rel" WHERE "user" = 'Max';
```

Listing 5.18: Vorverarbeitung für die Verwendung von `boundary`

Nachdem nun die zwei virtuellen Tabellen `relPartition` und `relConcept` durch die Ausdrücke aus Listing 5.18 vorliegen, kann `boundary` für die Umsetzung von *equal_pref* angewandt werden. Dazu sei Listing 5.19 betrachtet, welches von der Semantik mit Listing 5.17 übereinstimmt:

```
/* Nutzen der Prozedur boundary zur Bestimmung von equal_pref */
SELECT (5 <= sum(card)) AS equal_pref FROM "boundary"
  ('relPartition','{a1,a2}','relConcept');
```

Listing 5.19: Zweite Umsetzungsvariante für *equal_pref*

6 Evaluierung

Nach der Ausstattung von PostgreSQL mit Werkzeugen für die Musterextraktion, beschäftigt sich dieses Kapitel mit der praktischen Anwendung dieser Ansätze. Zu diesem Zweck findet zunächst eine empirische Auswertung über die erzielbaren Laufzeiten der konzipierten Ideen in einer gegebenen Testumgebung statt. Daraus kann sich dann ein Bild gemacht werden, in welchem Rahmen die einzelnen Methoden in der Praxis des CAPLE-Projekts Einsatz finden können. Der zweite Abschnitt dieses Kapitels widmet sich der Anwendung des Gesamtsystems auf konkrete CAM. So kann einerseits evaluiert werden, in wie weit die einzelnen Ansätze Muster aus den Daten erkennen, und andererseits ein erster Eindruck über die Justierung der zu setzenden Parameter in den einzelnen Algorithmen gewonnen werden.

6.1 Testumgebung

Als Testumgebung für die Laufzeitanalyse und das Parametrisieren der in Kapitel 4 eingeführten Methoden wird ein Apple iMac mit PostgreSQL 8.3 verwendet. Die wesentlichen Eigenschaften des Systems sowie wichtige Datenbankparameter sind durch das Datenblatt[1] aus Abbildung 6.1 gegeben. Dennoch bedarf es für das weitere Vorgehen jeweils einer kurzen Erläuterung der modifizierten Konfigurationsparameter in PostgreSQL:

- `block_size`: Dieser Wert beschreibt die physische Größe einer Datenbankseite auf der Festplatte.
- `effective_cache_size`: Der Wert, der dem Planer von PostgreSQL einen Hinweis darüber gibt, ob benötigte Daten möglicherweise bereits im Hauptspeicher vorliegen.

[1] Alle weiteren, nicht gelisteten, Datenbankkonfigurationsparameter sind standardmäßig unverändert geblieben.

- `shared_buffer`: Die Angabe dieses Werts symbolisiert PostgreSQL die Größe des Shared-Buffer-Pools, also die Größe des insgesamt verfügbaren Hauptspeichers für alle PostgreSQL-Prozesse.
- `work_mem`: Die Größe des Hauptspeichers, welcher PostgreSQL für wichtige Operationen potenziell zur Verfügung steht. Unter diese Operationen fallen z.B. sortierungsbasierte Algorithmen wie `DISTINCT`, `ORDER BY` oder Merge-Joins. Ebenfalls hash-basierte Ansätze wie Hash-Aggregationen bzw. Hash-Joins profitieren von einem großen Wert, so dass die Hash-Tabellen im Speicher gehalten werden können.

System Umgebung (iMac 4.1)	
Betriebsystem:	Mac OSX 10.4.11 (Darwin 8.11.1)
Prozessor:	2 GHz (Intel Core Duo) / L2-Cache 2 MByte
Speicher:	2 GByte DDR2 SDRAM (667 MHz)
Festplatte:	250 GByte (Maxtor 6L250M0)
Datenbanksystem:	PostgreSQL 8.3.7 / GCC 4.0.1
PostgreSQL spezifische Parameter	
block_size:	8 KByte
effective_cache_size:	128 MByte
shared_buffer:	24 MByte
work_mem:	256 MByte

Abbildung 6.1: Datenblatt der Testumgebung

Eine detaillierte Beschreibung der gelisteten und weiteren Parameter findet sich in [Pos09, EH09, Sch09]. Für die Planungsoptimierung von Anfragen sei insbesondere auf die zwei letztgenannten Literaturangaben verwiesen.

6.2 Empirische Laufzeitanalyse

Dieser Abschnitt beschäftigt sich mit der experimentellen Laufzeitanalyse von ausgewählten PL/pgSQL-Prozeduren (`represent`, `ind`, `lower`, `upper`, `boundary`, `outside`, `clusterRel`), welche im Hinblick auf die Auswertung von Benutzeraktivitäten im CAPLE-Projekt konzipiert wurden. Bevor jedoch auf die Ergebnisse der Analyse näher eingegangen werden kann, wird der Aufbau der Experimente näher skizziert.

Experimenteller Aufbau

Die Aufzeichnung von Benutzerdaten im CAPLE-Projekt besteht zum Zeitpunkt der Evaluierung aus Sicht des *User Activity Logger Wrapper* im Wesentlichen aus

vier zu nennenden Attributen. Zukünftig werden durch weitere Wrapper-Implementierungen mehr Attribute für die Auswertung von Benutzeraktivitäten zur Verfügung stehen. Um diesen Aspekt in der Laufzeitanalyse zu berücksichtigen, wird daher die Auswertung auf fiktiven textbasierten Datensätzen mit bis zu zehn Attributen durchgeführt. So können vom Datenvolumen her die gesammelten Informationen mehrerer *Wrapper* über einen gewissen Zeitraum durch die Analyse abgedeckt werden. Zur Verdeutlichung der charakteristischen Strukturen, auf der die Experimente durchgeführt werden, sei die folgende Abbildung 6.2 betrachtet:

Basiseigenschaften der Datentabelle	
Name:	TBTable
Attributmenge:	11
Größe:	26 MByte - 1,2 GByte
Tupel:	$2 \cdot 10^5$ - $9,8 \cdot 10^6$
Durchschnittl. Tupelgröße:	124 Byte
RST-spezifische Informationen	
Analysierte Attribute:	10
Analysierte Tupel:	$2 \cdot 10^5$ - $9,8 \cdot 10^6$
Äquivalenzklassen:	790
Zielkonzept:	15%
Obere Approximation:	18,2%
Untere Approximation:	13,7%
Grenzregion:	4,5%
Außenregion:	81,8%
Cluster-spezifische Informationen	
Analysierte Attribute:	1 - 10
Analysierte Tupel:	$1,2 \cdot 10^6$
Initiale Cluster:	55 - 713

Abbildung 6.2: Charakteristische Struktur des Experiments

Die Basis der Experimente ist eine Tabelle `TBTable`, welche die textbasierten Daten enthält. Für die Laufzeitanalyse der rein RST-gestützten Methoden sind zu Beginn $2 \cdot 10^5$ Tupel ($\approx$ 26 MByte) in `TBTable` enthalten. Die benötigte Zeit zum Abarbeiten des Datenvolumens wird dann erfasst und protokolliert. In jeder Folgeiteration wird an die Tabelle stets die gleiche Menge an Tupeln angefügt, so dass die Tabelle jeweils linear wächst. Nach 35 bis 50 Durchläufen enden die Messungen und es befinden sich $7 \cdot 10^6$ Tupel ($\approx$ 925 MByte), respektive $9,8 \cdot 10^6$ Tupel ($\approx$ 1,2 GByte) in der Tabelle. Die Analyse für das hierarchische Clustering hingegen basiert auf einer konstanten Tabellengröße ($1,4 \cdot 10^6$ Tupel $\approx$ 186 MByte). Hier wird zu Beginn des Experiments ein Attribut berücksichtigt, so dass 33 initiale Cluster durch den Algorithmus weiter zu gruppieren sind. Die verbrauchte Zeit in den einzelnen Code-Segmenten wird

dabei gemessen. Für folgende Iterationen werden sukzessive weitere Attribute in die Analyse mit einbezogen, so dass sich die Anzahl der initialen Cluster jeweils um ca. 50 Klassen erhöht. Nach dem Gruppieren von insgesamt 713 unterschiedlichen Objekten aus `TBTable` endet das Experiment. Eine chronologische Abfolge für die Tätigkeiten in einer Iteration beider Experimentarten liefert folgende Auflistung:

(i) Freigabe von unbenutztem Arbeitsspeicher und Aktualisierung von Statistiken der Tabelle (evtl. Restrukturierung von Indizes bzw. Clustern der Tabelle)

(ii) Zeitmessung für die zu analysierende(n) PL/pgSQL-Prozedur(en)

(iii) Protokollierung des Messergebnisses mit allen relevanten Informationen, des Datenbestandes und den Strukturen auf denen das Messergebnis basiert

(iv) Erweiterung des Datenbestandes um $2 \cdot 10^5$ Tupel bzw. Betrachtung weiterer Attribute für die nächste Zeitmessung

Ununterscheidbarkeit

Bei dieser Messreihe sind die Funktionen `represent` und `ind` mit jeweils zwei unterschiedlichen Ausführungsplänen hinsichtlich ihrer benötigten Zeit untersucht worden. Für `represent` wurden dabei die zwei gekapselten SQL-Ausdrücke aus Listing 5.3 verwendet. Auf der anderen Seite wurde die Prozedur `ind` aus Listing 5.6 mit der Option `SET enable_hashagg = 0` bzw. `SET enable_hashagg = 1` ausgeführt. Das Ergebnis dieses Experiments ist in Abbildung 6.3 durch die abgebildeten Trendkurven skizziert:

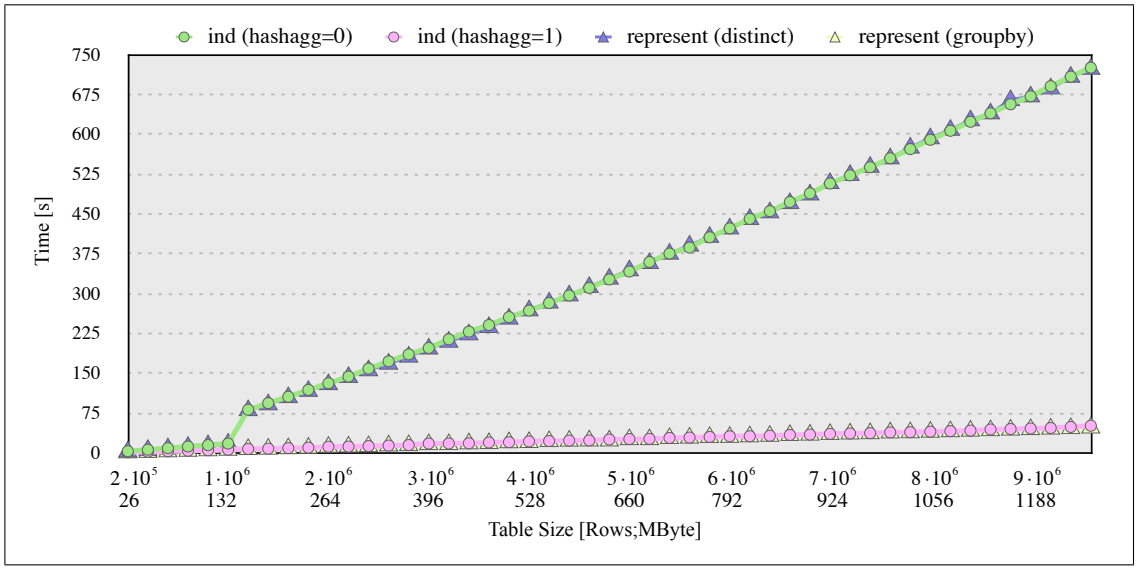


Abbildung 6.3: Experimentelle Laufzeit für `represent` und `ind`

Es ist ersichtlich, dass die Methode `represent` mit dem Schlüsselwort `DISTINCT` und die Funktion `ind` mit dem Planerhinweis `SET enable_hashagg = 0` im Wesentlichen das gleiche Laufzeitverhalten aufweisen. Charakteristisch für beide Kurven ist der massive Zeitanstieg bei $1,4 \cdot 10^6$ Tupel ($\approx$ 186 MByte), welcher den vorangegangenen Steigungstrend ($20 \cdot 10^5$ bis $1,2 \cdot 10^6$ Tupel) nicht fortsetzt. Der Grund für das veränderte Kurvenverhalten lässt sich am Beispiel des Ausführungsplans von `ind` mit `SET enable_hashagg = 0` verdeutlichen:

```
/* Ausführungsplan von ind (hashagg=0) bei 1200000 Tupeln */
Group  (cost=143341.87..173472.55 rows=109567 width=101) (actual time
    =15314.517..17907.004 rows=790 loops=1)
  ->  Sort  (cost=143341.87..146081.02 rows=1095661 width=101) (actual time
      =15314.509..15637.285 rows=1200000 loops=1)
        Sort Key: a0, a1, a2, a3, a4, a5, a6, a7, a8, a9
        Sort Method:  quicksort  Memory: 241156kB
        ->  Seq Scan on "TBTable"  (cost=0.00..33428.61 rows=1095661 width=101) (
            actual time=0.013..1383.586 rows=1200000 loops=1)
Total runtime: 17943.114 ms

/* Ausführungsplan von ind (hashagg=0) bei 1400000 Tupeln */
Group  (cost=181829.74..220605.79 rows=141004 width=100) (actual time
    =77228.730..81110.398 rows=790 loops=1)
  ->  Sort  (cost=181829.74..185354.84 rows=1410038 width=100) (actual time
      =77228.722..78480.708 rows=1400000 loops=1)
        Sort Key: a0, a1, a2, a3, a4, a5, a6, a7, a8, a9
        Sort Method:  external merge  Disk: 162472kB
        ->  Seq Scan on "TBTable"  (cost=0.00..37813.38 rows=1410038 width=100) (
            actual time=0.037..1603.019 rows=1400000 loops=1)
Total runtime: 81182.557 ms
```

Listing 6.1: Ausführungspläne von ind(hashagg=0)

Beide Ausführungspläne aus Listing 6.1 eliminieren Duplikate aus `TBTable` auf Basis einer Sortierung, wobei die Sortierung der Relation bei $1,2 \cdot 10^6$ Tupel noch im reservierten Hauptspeicher stattfinden kann ($\approx$ 241 MByte). Bei der Abarbeitung der $1,4 \cdot 10^6$ Tupel kann dies mit den gesetzten Parameter `work_mem` nicht mehr gewährleistet werden und PostgreSQL muss auf eine teure Sortierung[2] im Sekundärspeicher ausweichen. Dies gilt auch für alle weiteren Zeitmessungen ($> 1,4 \cdot 10^6$ Tupel) der Methoden `represent` mit `DISTINCT` und `ind` mit `SET enable_hashagg = 0`. Es ergibt sich somit insgesamt für die Berechnung von $9,8 \cdot 10^6$ Tupel ($\approx 1,2$ GByte) ein zeitlicher Aufwand von 725 Sekunden für das einmalige Ausführen der jeweiligen Methode.

[2] Setzt man einen linearen Zusammenhang zwischen benötigten Speicher und Tupelmenge für die Sortierung voraus, so müssten für die Sortierung im Hauptspeicher von $9,8 \cdot 10^6$ Tupel aus der Tabelle `TBTable` ca. 1,6 bis 2 GByte auf diesem reserviert sein.

Eine andere Algorithmik legt PostgreSQL mit der Hash-Aggregation für die Duplikateneliminierung bzw. das Bilden von Gruppen zugrunde (`SET enable_hashagg = 1`). Hierbei wird über die Relation iteriert und jedes Tupel anhand einer gegebenen Hash-Funktion in eine Hash-Tabelle eingeordnet. Äquivalente Tupel der Relation befinden sich dann optimalerweise nach dem Lesen der Relation in einem *Bucket* der Hash-Tabelle [RG02]. Der Vorteil, dass PostgreSQL für diesen Ausführungsplan keine Sortierung benötigt wird durch das Laufzeitverhalten der Methode `represent` in der `GROUP BY`-Variante sowie der Prozedur `ind` mit `SET enable_hashagg = 1` deutlich. Beide Kurven verlaufen im Vergleich zu den sortierungsbasierten Ansätzen in der Testumgebung mit einem deutlich geringeren Steigungsgrad. Durch die Verwendung der Hash-Aggregation ergibt sich insgesamt ein 14-mal besseres Laufzeitverhalten, als bei der sortierungsbasierten Variante, so dass sich bei $9,8 \cdot 10^6$ Tupel eine Laufzeit von unter 50 Sekunden einstellt. Aus praktischer Sicht bildet die Hash-Aggregation somit ein gutes Fundament für die weiteren zu analysierenden Prozeduren. Es sei allerdings erwähnt, dass für das Hashen genügend Primärspeicher für PostgreSQL zur Verfügung stehen muss, damit die Hash-Tabellen im Speicher gehalten werden können (vgl. Kapitel 6.1). Dies ist über alle Testläufe hinweg stets gewährleistet. Kann dies allerdings nicht gewährleistet werden, so verändert der Planer trotz der Option `SET enable_hashagg = 1` seinen Ausführungsplan zu der sortierungsbasierten Variante.

Konzeptapproximation

Dieses Experiment erfasst die Laufzeit der Konzeptapproximation jeweils mit und ohne Unterstützung von Indizes. Das zu approximierende Zielkonzept `TB_X_Table` ist dabei eine Teilmenge aus der Datentabelle `TBTable` in Form einer Sicht. Es umfasst für jeden Testlauf ca. 15% der Daten aus `TBTable`. Die untere und obere Approximation bezüglich der zehn zu analysierende Attribute entspricht $13,7\%$ respektive $18,2\%$ aus `TBTable` (vgl. Abbildung 6.2).

Beginnend mit der Analyse ohne jegliche Indexunterstützung ergibt sich ein ähnliches Bild wie bei der Untersuchung der Laufzeiten der sortierungsbasierten Ausführungspläne für die Prozeduren `represent` bzw. `ind`. Ab einer Tupelanzahl von $1,4 \cdot 10^6$ ist ein deutlicher Anstieg der Abarbeitungsdauer für die Funktionen `upper`, `boundary` und `outside` bemerkbar. Dies liegt jedoch nicht daran, dass die Option für Hash-Aggregationen deaktiviert ist, sondern daran, dass sich der Planer ohne Index in allen drei Fällen für Merge-Joins[3] entscheidet. Dadurch muss zunächst die gesamte Ta-

[3] Zu weiteren Ausführungen über Join-Varianten sei auf weiterführende Literatur wie z.B. [EN05, SHS05] verwiesen.

belle `TBTable` anhand der Join-Attribute sortiert werden, um letztlich die gewählte Verschmelzungsoperation durchzuführen. Nachdem jedoch für alle Folgemessungen ($> 1,4 \cdot 10^6$ Tupel) die Sortierung auf den Sekundärspeicher verlagert wird, besitzen alle drei Prozeduren ein nahezu lineares Verhalten in der gegebenen Testumgebung, so dass sich die zuletzt gemessene Zeit für die Extraktion des Zielkonzepts aus $7 \cdot 10^6$ Tupeln ($\approx$ 925 MByte) noch unter 570 Sekunden bewegt. Die Funktion `lower` ohne Indizierung auf `TBTable` zeigt insgesamt eine viel geringere Steigung. Dies findet Begründung darin, dass zwar für die Schnittmengenbildung in PostgreSQL typischerweise eine Sortierung durchgeführt wird, dies im Falle von `lower` aber erst nach der Ausführung zweier Hash-Aggregationen zur Berechnung von `ind` für `TBTable` und `TB_X_Table` geschieht. Die zu sortierenden Datenmengen sind dann im Testszenario so klein, dass sie im Hauptspeicher gehalten werden können. Daraus ergibt sich ein maximaler Zeitaufwand zur Bestimmung der unteren Approximation von unter 50 Sekunden, was im Vergleich zu den Funktionen `upper`, `boundary` und `outside` um den Faktor 10 schneller ist. Das gesamte Ergebnis ist durch Abbildung 6.4 grafisch dargelegt:

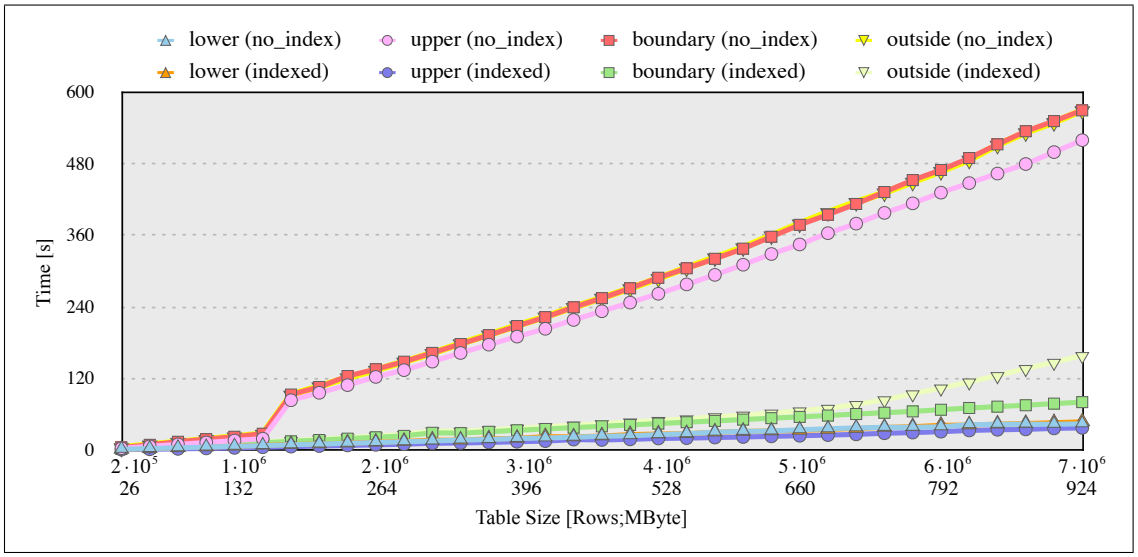


Abbildung 6.4: Experimentelle Laufzeit für die Konzeptapproximation (I)

Die indexgestütze Variante zur Berechnung der Konzeptapproximation ist ebenfalls in Abbildung 6.4 skizziert. Die Untersuchung fand für alle zehn Attribute auf Basis einer zusammengesetzten balancierten Indexstruktur statt. Der Planer verwendet dabei für die Methoden `upper`, `boundary` und `outside` einen herkömmlichen Nested-Loop-Join in Verbindung mit einem Index-Scan auf der Tabelle `TBTable`. Dies führt im untersuchten Datenbestand für alle drei Methoden zu einer deutlich geringeren Laufzeit als ohne Indexunterstützung. Die benötigten Zeiten sind im Schnitt um

den Faktor 6 besser. Es ist jedoch ersichtlich, dass die Prozedur `outside` bei steigendem Datenvolumen ($> 5 \cdot 10^6$ Tupel $\approx$ 660 MByte) schneller wächst als `upper` und `boundary`. Dies lässt sich im Wesentlichen darauf zurückführen, dass zwar ein Index-Scan für die Verschmelzungsoperation in der Methode `outside` zum Einsatz kommt, jedoch viel mehr Traversierungen durch die Indexstruktur nötig sind als beispielsweise bei der Funktion `upper` (vgl. Formel 4.7 und 4.9). So stellt sich der Nested-Loop-Join in Verbindung mit dem Index-Scan für `outside` als der dominierende Kostenfaktor dar. Ein anderes Phänomen tritt bei genauerer Betrachtung von Abbildung 6.4 im Laufzeitverhalten der Prozedur `lower` auf. Die Zeiten für die Berechnung der unteren Approximation stimmen mit denen von `lower` ohne Indexunterstützung überein. Dies ist deshalb der Fall, weil trotz des existenten Indexes der Ausführungsplan keine Indexnutzung vorsieht und sich beide Pläne somit gleichen.

Eine weitere Verbesserung des Laufzeitverhaltens bei der Approximation des Zielkonzepts lässt sich in PostgreSQL durch das Konzept des Clusterns[4] erwirken. Dabei wird ein Datenbestand nach einer gegebenen Indexstruktur neu angeordnet, so dass ähnliche Tupel auch physisch auf dem Sekundärspeicher nahe beieinander liegen. Die erfassten Laufzeiten der einzelnen Prozeduren zur Berechnung der Konzeptapproximation finden sich in Abbildung 6.5:

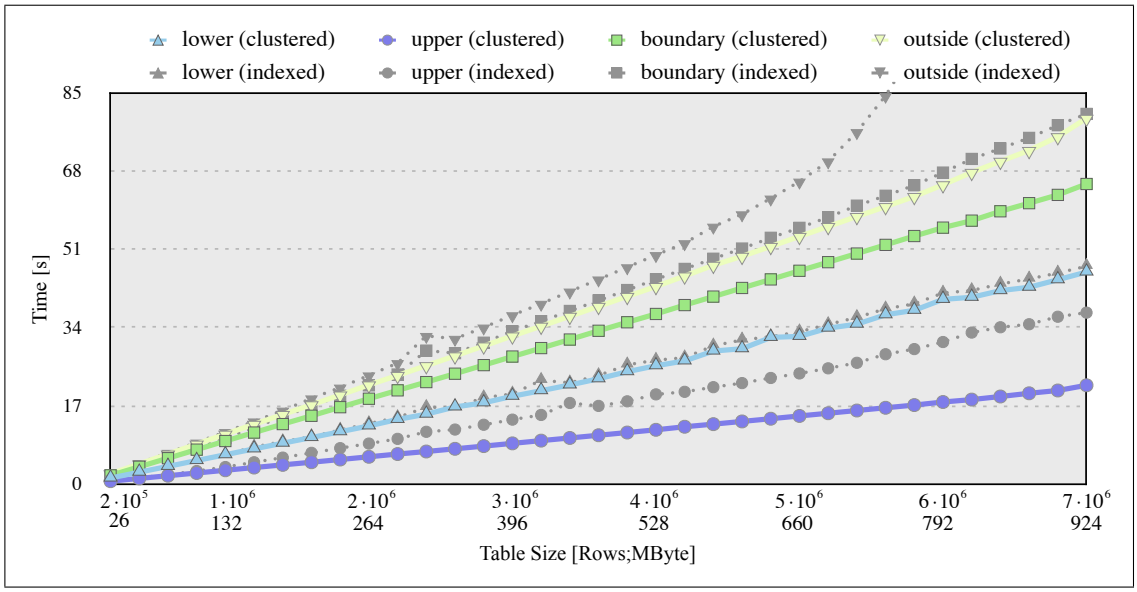


Abbildung 6.5: Experimentelle Laufzeit für die Konzeptapproximation (II)

[4] Details zu diesem Thema finden sich unter `http://www.postgresql.org/docs/8.3/static/sql-cluster.html`, Stand 18.11.09

Das Clustern der Tabelle `TBTable` basiert auf dem selben zusammengesetzten Index, wie die bereits zuvor getätigten Messungen. Aus diesem Grund sind zu Vergleichszwecken auch noch einmal die Ergebnisse ohne Clustern des Datenbestands (jedoch mit Indizierung) in Abbildung 6.5 grau hinterlegt eingefügt. Im direkten Vergleich treten die wesentlichen Verbesserungen in der Methode `outside` und `upper` auf. Ebenfalls partiell auf `upper` basierend besitzt die Funktion `boundary` eine geringere Laufzeit, als die gleiche Funktion ohne physische Umstrukturierung von `TBTable`. Insgesamt kann die Leistung aller drei Prozeduren bezogen auf den jeweiligen zeitlichen Aufwand um den Faktor 1,5 bis 2 verbessert werden.

Es sei abschließend für die Laufzeitanalyse der indexgestützten Konzeptapproximation erwähnt, dass die gute Leistung, welche mit der Indizierung erreicht werden kann, im Kontrast zu der benötigten Speicherkapazität liegt. Die Testergebnisse haben gezeigt, dass die Größe des Indexes in der gleichen Größenordnung liegt, wie die Datentabelle. Es muss also bei der Verwendung der erläuterten Methoden stets mit den doppelten Speicherkosten gerechnet werden, was für manchen Anwendungsfall nicht akzeptabel ist.

Attributgewichtetes hierarchisches Clustering

Anders als bei den vorangegangenen Analysen ist beim hierarchischen Cluster-Algorithmus `clusterRel` nicht etwa das steigende Datenvolumen der dominierende Kostenfaktor. Selbst beim kontinuierlichen Vergrößern der Tabelle `TBTable` würde `clusterRel` nur unwesentlich mehr Zeit benötigen, um eine Hierarchie von Gruppen auf den Daten zu erzeugen. Dies ist auf die guten Leistungen der Prozedur `ind`, auch bei großen Datenmengen, zurück zu führen (vgl. Abbildung 6.3). `ind` wird in `clusterRel` vor dem Clustering dazu verwendet, um eine Reduktion auf die wesentlichen Strukturen in den Daten zu erwirken und federt somit den Zeitaufwand für das Clustern großer Mengen ab. Kritisch für das eigentliche Clustering ist vielmehr die Anzahl der initialen Cluster[5], die aus `ind` resultieren. Von dieser Anzahl ist zum einen die Größe der Distanzmatrix abhängig und zum anderen die Anzahl der Iterationen, die `clusterRel` durchlaufen muss. Deshalb basiert diese Analyse auf einer konstanten Datenmenge von $1,4 \cdot 10^6$ Tupeln ($\approx$ 186 MByte) mit steigender Anzahl an initialen Clustern. Zusätzlich wurde für die Laufzeitanalyse die Prozedur `clusterRel` in logische Code-Blöcke eingeteilt, so dass zeitaufwändige Operationen beim Durchlaufen des Algorithmus detektiert und weiter analysiert werden können.

[5] Im Kontext der RST sind dies die unterschiedlichen Objekte, welche die Äquivalenzklassen der Ununterscheidbarkeitsrelation repräsentieren.

Das Flächendiagramm aus Abbildung 6.6 enthält das aufgeschlüsselte Laufzeitergebnis nach der jeweiligen Verweildauer in den einzelen Blöcken bei der initialen Cluster-Anzahl 55 bis 713:

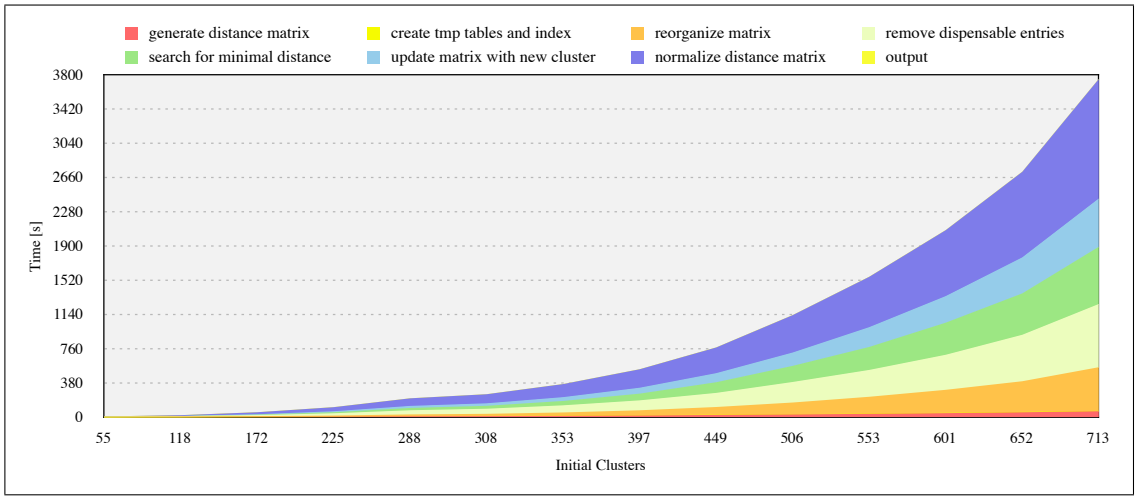


Abbildung 6.6: Experimentelle Laufzeit für das Clustern von Aktivitäten

Es wird deutlich, dass `clusterRel` ein sehr stark wachsender Algorithmus ist, dessen Hauptaufwand in jeder Iteration aus der Aktualisierung der Distanzmatrix resultiert. Insbesondere das Vereinheitlichen der Einträge in der Matrix, die vom Clustering in jedem Schritt betroffen sind, nimmt einen Großteil der Zeit ein, welche insgesamt für die Abarbeitung benötigt wird. Selbst die Verwendung eines intern angelegten Indexes auf der Distanzmatrix kann die Verweilzeit in diesem logischen Block über die gesamte Laufzeit des Algorithmus nicht mindern. Begründet werden kann dies durch das ständige Modifizieren der Matrix pro Iteration, so dass die Indexstruktur schnell eine veraltete Sicht auf die zu gruppierenden Klassen erlangt. Um diese Situation zu vermeiden, wurden auch einige Restrukturierungen des Indexes durchgeführt. Vergleichstests haben jedoch gezeigt, dass diese Aktualisierungsstrategie das Gesamtergebnis in der gegebenen Testumgebung mehr verschlechtert als ohne Reorganisation des Indexes.

Dennoch wird bei einer detaillierten Code-Betrachung von `clusterRel` aus Kapitel 5.4 abschließend deutlich, dass diese Implementierung noch weiteres Optimierungspotenzial besitzt, so dass die Aufwandskurve auch für größere Mengen an initialen Klassen anwendbar wäre.

Anmerkungen zu den Laufzeiten weiterer Algorithmen

Das Laufzeitverhalten der PL/pgSQL-Prozedur `classify` und des aus SQL-Befehlen zusammensetzbaren Verfahrens *equal_pref* werden an dieser Stelle nicht weiter ver-

tieft. Sie lassen sich im Wesentlichen aus den Laufzeiten von `ind` bzw. `boundary` ableiten. `ind` benötigt für die Abarbeitung von $7 \cdot 10^6$ Tupeln 34 Sekunden (vgl. Abbildung 6.3), während `boundary` für die gleiche Menge an Daten ca. doppelt soviel Zeit in Anspruch nimmt (vgl. Abbildung 6.5).

6.3 Strategien und Parameter in der Praxis

Da die entworfenen Algorithmen weitestgehend semi-automatischer Natur sind, befasst sich dieser Teil der Evaluierung mit der Feinjustierung der zu setzenden Parameter, um letztlich qualitative Muster zu erhalten. Für dieses Vorhaben liegen Produktivdaten[6] von Mitarbeitern aus dem CAPLE-Projekt vor, die alltägliche Tätigkeiten am Arbeitsplatz widerspiegeln. Sie wurden über einen Zeitraum von zwei Monaten durch den *User Activity Logger Wrapper* und das CAPLE-Framework aufgenommen. Unabhängig davon haben die Mitarbeiter für jeden Tag Protokolle über die Themen und Aufgaben angefertigt, an denen sie tatsächlich gearbeitet haben. Dadurch ist zumindest partiell eine Gegenüberstellung zwischen den Ergebnissen der Algorithmen und dem subjektiven Empfinden der Benutzer möglich.

Grobe Rekonstruktion von Tätigkeiten

Besteht ein Anwendungsfall darin, die gespeicherten Aktivitäten eines Benutzers über einen bestimmten Tag hinweg zu reproduzieren, ist dies mit der Prozedur `classify`, unter spezieller Anpassung der Parameter `alpha` und `beta` im Wesentlichen möglich. Im Sinne einer automatischen Anwendung von `classify` auf beliebige Tage, durch universelle Schwellwerte, haben Tests auf den Produktivdaten gezeigt, dass häufig ungültige Konzeptapproximationen entstehen. Der Grund für die schlechten Ergebnisse liegt darin, dass nicht nur die Aktivitäten eines Benutzers über den Tag gesehen einer unregelmäßigen Verteilung unterliegen, sondern auch manche Tätigkeiten auf ihre zeitliche Dichte bezogen näher aneinander liegen als andere. Es ist ersichtlich, dass beide Phänomene kritisch für das Finden von qualitativ guten Mustern unter einem universellen Einsatz von `classify` sind. Eine mögliche Herangehensweise zur Lösung dieses Problems wäre es, den gegebenen Datenraum über die Tageszeiten zu partitionieren, um dann auf jede Teilmenge `classify` mit einem festen `alpha` und `beta` anzuwenden. Die Ergebnisse aus den Teilberechnungen könnten dann zu einem Gesamtergebnis zusammengefasst werden. Dies würde lokal gesehen

[6] Die vollständigen Daten liegen zur Reproduzierbarkeit am Fraunhofer-Institut FIT bereit. Der Ansprechpartner für die Ausgabe ist Herr Dr. M. Wolpers.

gute Ergebnisse erzielen, hat jedoch zum einen den Nachteil, dass globale Fehlinterpretationen auftreten können und zum anderen höchst wahrscheinlich Konflikte bei der Zusammenführung der Teilergebnisse entstehen, welche es zu behandeln gilt. Ein anderer Lösungsansatz wird durch Abbildung 6.7 dargelegt. Sie beinhaltet die Ergebnisse[7] der Anwendung von `classify` auf die Aktivitäten von Mitarbeitern an verschiedenen Tagen $D_1, ..., D_{15}$ mit der Attributmenge $\{fName, tName\}$, die bezogen auf die beschriebene Aktivitätsdichte variieren.

Tag	$\|\mathbb{U}\|$	α	β	$\|K_{h=1}\|[\%]$	$\|K_{h=?}\|[\%]$	$\|K_{h=0}\|[\%]$	α [%]	β [%]
D_1	394	66	17	75,8	18,2	6,0	16,8	4,3
D_2	700	103	37	62,3	22,4	15,3	14,7	5,3
D_3	859	158	62	52,2	16,0	31,8	18,4	7,2
D_4	1393	224	53	64,5	23,4	12,1	16,1	3,8
D_5	1768	283	30	68,6	21,9	9,5	16,0	1,7
D_6	2232	335	83	45,7	38,8	15,5	15,0	3,7
D_7	2713	387	131	59,8	13,6	26,6	14,3	4,8
D_8	3348	700	172	47,7	32,3	20,0	20,9	5,1
D_9	3948	630	129	82,0	10,4	7,6	16,0	3,3
D_{10}	4584	708	192	43,2	43,2	13,6	15,4	4,2
D_{11}	4819	1095	235	23,6	47,6	28,8	22,7	4,9
D_{12}	5647	1305	308	60,6	32,6	6,8	23,1	5,5
D_{13}	10168	1759	344	69,2	20,9	9,9	17,3	3,4
D_{14}	13824	2519	756	48,8	43,2	8,0	18,2	5,5
D_{15}	16652	2649	875	36,5	22,4	41,1	15,9	5,3
						$\overline{x}_{arith} \approx$	17,4	4,5
						$\sigma^2 \approx$	$7,3$	$1,5$

Abbildung 6.7: `classify` angewandt auf verschiedene Arbeitstage von Benutzern

Für jeden analysierten Tage wurden die Parameter `alpha` und `beta` so gesetzt, dass sie die erstellten Aktivitätsprotokolle der Mitarbeiter im Wesentlichen abdecken. So konnten zwar gute Ergebnisse für die Reproduktion der alltäglichen Tätigkeiten erzeugt werden, aber die schwankenden Absolutwerte ließen auf den ersten Blick keinerlei Rückschlüsse auf ein Muster zu. Bezogen auf die Gesamtzahl aller Aktivitäten hat die relative Betrachtung von `alpha` und `beta` allerdings recht stabile Werte mit variabler Aktivitätsdichte für jeden analysierten Tag hervorgebracht. Somit wird neben dem eigentlichen Ergebnis der Klassifikation aus Abbildung 6.7 zusätzlich deutlich, dass im Schnitt die Äquivalenzklassen, welche einen Anteil von 17,4% aller Aktionen eines Tages ausmachen, sich sicher zu den Tätigkeiten des Benutzers zählen lassen. Des Weiteren hat sich aus den durchgeführten Experimenten erge-

[7] Die folgende Tabelle behandelt nur einen Ausschnitt der evaluierten Tagesaktivitäten. Die vollständige Auswertungstabelle befindet auf der CD im Verzeichnis `/Classify_Analysis/`.

ben, dass durchschnittlich solche Aktionen mit einem kleineren Anteil als 4,5% pro Tag definitiv nicht zu den Tätigkeiten gehören, während über die Mengen zwischen den beiden relativen Schwellwerten keine genaue Aussage getroffen werden kann. Aufgrund der geringen Streuung bei den durchgeführten und weiteren Tests über das weitestgehend vollständige Spektrum an Aktivitätsdichten, ist mit dieser Konfiguration von `classify` zu erwarten, dass die relativen Parameter auch Bestand für zukünftige Experimente im CAPLE-Projekt haben. Da es sich dennoch bei den beobachteten Werten um Durchschnittswerte über mehrere Testreihen handelt, sei angemerkt, dass diese nur als Richtwerte zu sehen sind.

Während die zuvor erläuterten Ergebnisse aus Abbildung 6.7 auf den Attributen *fName* und *tName* beruhen, haben praktische Tests mit mehreren Attributen gezeigt, dass die Suche nach stabilen Schwellwerten im gesamten Spektrum der analysierten Tage für automatisierte Auswertungen immer schwieriger wird. Begründung findet dies in der Tatsache, dass durch die Vergrößerung der Attributmenge mit sinnvollen Attributen zwar mehr Information in die Verteilung einfließt, jedoch aus diesem Grund auch feinere Partitionierungen auf den Datenräumen erwirkt werden. Die Schwellwerte müssen demnach stark verkleinert werden. Dies birgt auf der einen Seite das Risiko sehr grobe Zielkonzepte zu generieren, was wiederum auf der anderen Seite zu einem hohen Justierungsaufwand der Parameter `alpha` und `beta` führt. Es ist also nachvollziehbar, dass beide Gesichtspunkte den automatischen Betrieb von `classify` gefährden. Abbildung 6.8 zeigt am Beispiel eines Tages mit ca. 4000 Aktivitäten die Entwicklung der Datenverteilung für diese Problematik bei steigender Attributanzahl.

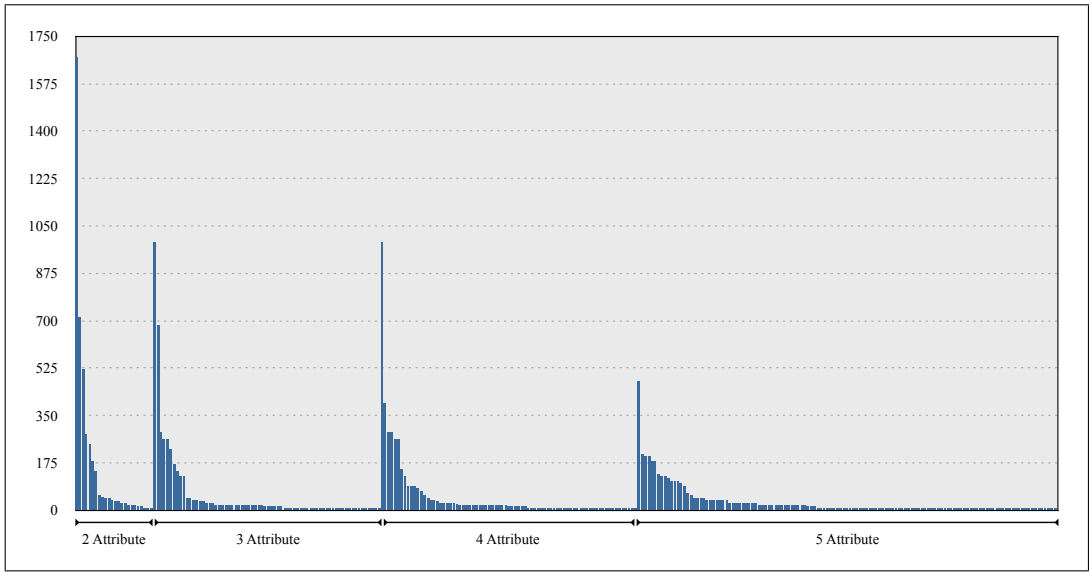


Abbildung 6.8: Entwicklung der absoluten Häufigkeitsverteilung bei mehr Information

Hierarchien und Klassen aus Benutzeraktivitäten

An dieser Stelle soll ein subjektiver Eindruck davon vermittelt werden, in wie weit der implementierte Cluster-Algorithmus `clusterRel` im Bezug auf seine konzeptionellen Ideen gute und aussagekräftige Hierarchien von Aktivitäten liefern kann. Dabei wurde zunächst in Experimenten das Clustering von realen Benutzeraktivitäten mittels der natürlichen Distanz analysiert. In weiteren Untersuchungen wurden dann die Ergebnishierarchien auf Basis der gewichteten Distanz mit wechselnder Parametrierung des Gewichtungsvektors ausgewertet. Der Vergleichbarkeit halber wurden die Tests stets auf den selben CAM und Attributmengen durchgeführt. $dayTime$ ist dabei das diskretisierte Attribut, welches die Tageszeiten in sieben kategorische Zeitintervalle einteilt, während die restlichen Attribute $alAction$ (Aktivitätstyp) , $alwDescription$ (Beschreibung der Aktivität), $tName$ (Dokumententyp) und $fName$ (Anwendungsname) aus Kapitel 4.1.2 bekannt sind.

Die resultierende Hierarchie von `clusterRel` in Verbindung mit der natürlichen Distanz und der Complete-Linkage-Strategie hat im Wesentlichen zwei sehr starke Gruppen von Aktivitäten hervorgebracht. Die erste dieser beiden Cluster enthält dabei Aktivitäten zu beliebigen Tageszeiten in diversen Anwendungen mit der zusätzlichen Eigenschaft, dass immer ein Dokument in die jeweilige Tätigkeit involviert ist. Die Aktivitäten der zweiten Gruppe besitzen diese Eigenschaft nicht. Die nähere Betrachtung dieses zweiten Clusters hat jedoch ergeben, dass sich aus ihr weitere vier Segmente extrahieren lassen. Das erste davon enthält solche Aktivitäten, welche zu beliebigen Tageszeiten den Ruhezustand des Computers symbolisieren, während die zweite große Subkategorie von ihrer Semantik her ganz andere Benutzeraktionen umschließt. Sie besteht aus den Aktivitäten, die das Öffnen oder Beenden von Anwendungen repräsentieren. Die letzten beiden Cluster umfassen zunächst alltägliche Aktivitäten unter Verwendung diverser Applikationen. Beide zeichnen sich jedoch durch einen hohen Anteil an Aktitivitäten auf speziellen Anwendungen aus, was letztlich die Separation dieser beiden Gruppen durch den Algorithmus begründet. Der direkte Vergleich zum Single-Linkage hat auf den gleichen Daten ein anderen Effekt der Segmentierung. Hier lässt sich zwar auch in zwei Arten von Aktivitäten unterscheiden, jedoch nicht so offensichtlich, wie es beim Complete-Linkage der Fall ist. Die erste Gruppe enthält dabei alle Aktivitäten, die den Ruhezustand des Computers beschreiben, während das andere Cluster die restlichen Benutzeraktionen umfasst. Es enthält im Wesentlichen viele kleine, aber dafür starke Gruppen, die thematisch ähnliche Aktivitäten auf Applikationen mit partiell involvierten Dokumenten widerspiegeln. Drei Segmente dieses großen Clusters zeichnen sich durch die hohe Anzahl vieler ähnlicher Aktionen aus. Sie bilden mit gewissen Unregelmäßigkeiten

eine ähnliche Struktur wie die drei letzt beschriebenen Gruppen aus der Anwendung von `clusterRel` mit der Complete-Linkage-Strategie. Zur Veranschaulichung der beschriebenen Cluster sei auf die folgende Abbildung verwiesen. Abbildung 6.9 (a) enthält dabei die subjektiv extrahierten Cluster aus dem resultierenden Dendrogramm[8] der Complete-Linkage-Variante und Abbildung 6.9 (b) die Gruppen, welche mit dem Single-Linkage gefunden wurden.

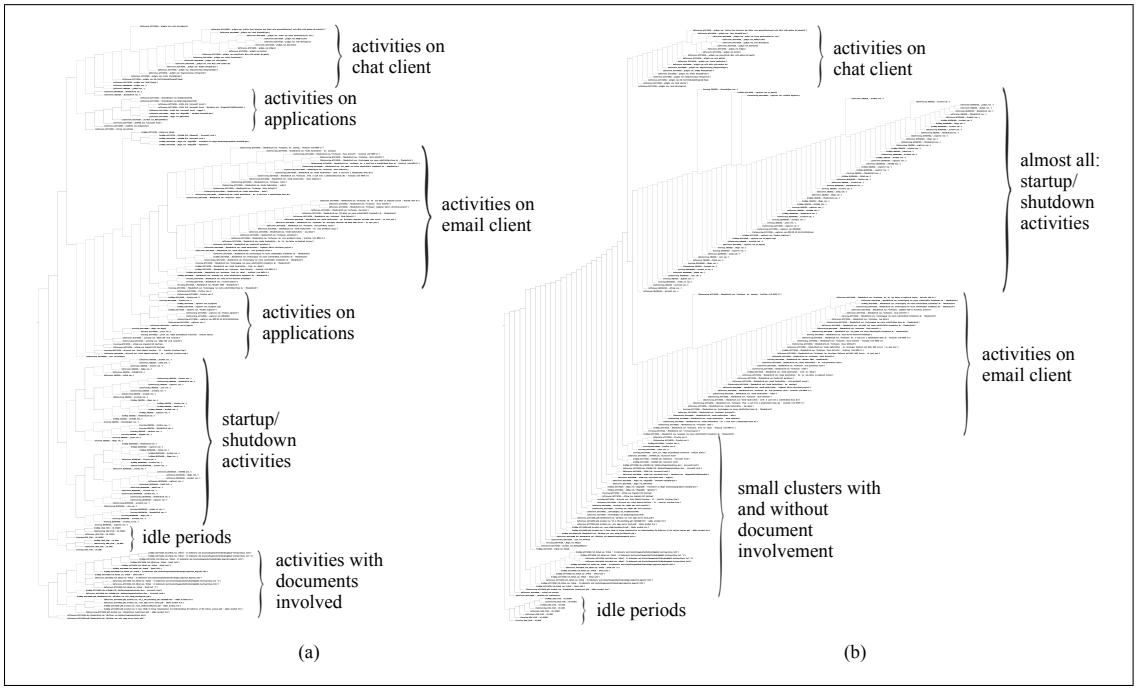


Abbildung 6.9: Ergebnisse des hierarchischen Clusterings ohne Gewichtung

Die Auswertung der vorliegenden Benutzerdaten hat gezeigt, dass mit `clusterRel` ohne den optionalen Gewichtungsvektor bereits Gruppen von Daten mit thematischer Ähnlichkeit extrahiert werden können. Tests auf weiteren Datensätzen haben im Wesentlichen zu gleichen Cluster-Strukturen geführt, so dass davon ausgegangen werden kann, dass das Clustern auf CAM mit diesem Algorithmus und dem gegebenen Distanzmaß prinzipiell möglich ist. Dennoch ist zu sagen, dass einige wenige Unregelmäßigkeiten mit der angewandten Metrik und den Strategien auf den CAM zu verzeichnen sind.

Um eine stärkere Kategorisierung der gegebenen Daten zu erwirken, lassen sich Gewichte auf die einzelnen Attribute verteilen. Damit kann erreicht werden, dass Aktivitäten, die sich in einem hoch gewichteten Attribut unterscheiden, auch aus geo-

[8] Die Dendrogramme wurden mit einer angepassten Variante des in [Seg08] beschrieben Python-Scripts generiert. Es befindet sich auf der beigefügten CD im Verzeichnis `/Cluster_Analysis/Python_Script/`.

grafischer Perspektive weit auseinander liegen und von `clusterRel` somit auch erst spät gruppiert werden. Die Analyse der generierten Hierarchien von `clusterRel` hat mit den Gewichtungen $w(dayTime) = 1$, $w(alAction) = 1$, $w(alwDescription) = 5$, $w(tName) = 15$, $w(fName) = 15$ und dem Complete-Linkage in fast allen Fällen eine Seperation in starke und homogene Klassen hervorgebracht. Wie beim Clustering ohne Gewichtung und dem Complete-Linkage lassen sich die Aktivitäten zunächst in Gruppen mit und ohne Dokumentenzugehörigkeit unterteilen, bedingt durch das hohe Gewicht auf dem Attribut $tName$. Betrachtet man allerdings das zugehörige Dendrogramm eine Hierarchiestufe tiefer, so wird ersichtlich, dass sich die Aktionen mit involvierten Dokumenten in drei weitere Cluster einteilen lassen. Sie sind nach den Dokumenttypen `PDF`, `TXT`, `DOC` und den zugehörigen Anwendungen zu beliebigen Tageszeiten segmentiert. Die andere große Gruppe von Aktivitäten enthält 14 starke Subklassen, die die restlichen Aktivitäten nach verwendeter Applikationen kategorisiert. Auf der anderen Seite hat das Clustering mit der Single-Linkage-Strategie bei gleicher Gewichtung ähnliche Segmentierungsergebnisse erzielt wie das Gruppieren mit dem Complete-Linkage. Die Reihenfolge der einzelnen Cluster hat sich jedoch so verschoben, dass nicht zwischen Aktionen mit und ohne Dokumentenzugehörigkeit unterschieden werden kann, sondern in Aktivitäten auf Applikationen zu beliebigen Tageszeiten und solchen Aktivitäten, die den Ruhezustand des Computers beschreiben. Das große Cluster, welches die alltäglichen Aktivitäten beschreibt, lässt sich dabei in acht weitere Gruppen unterteilen. Während sieben dieser Klassen im Wesentlichen aus Aktionen bestehen, in die ein Dokument involviert war, besteht das achte Cluster aus Aktivitäten, kategorisiert nach der jeweiligen Anwendung. Aufgrund der niedrigen Gewichtung der Attribute $dayTime$, $alAction$ und $alwDescription$, war keine Segmentierung nach diesen Merkmalen zu erkennen. Eine grafische Aufbereitung zu den beschriebenen Clustern mit der subjektiven Gewichtung lässt sich aus Abbildung 6.10 (a) für die Complete-Linkage-Strategie und aus Abbildung 6.10 (b) für das Single-Linkage entnehmen.

Das beschriebene Gruppierungsergebnis in Verbindung mit Ergebnissen anderer Datensätze hat die Erkenntnis gebracht, dass `clusterRel` durch diese Parametrisierung subjektiv beeinflusst bzw. gesteuert werden kann. Eine vergleichsweise hohe Gewichtung auf dem Attribut $dayTime$ würde z.B. zur Folge haben, dass Aktivitäten zunächst nach ihrem tageszeitlichen Auftreten gruppiert werden, während die Erhöhung des Gewichts für $alAction$ beispielsweise dazu führt, Aktivitäten nach ihrem Typ zu segmentieren.

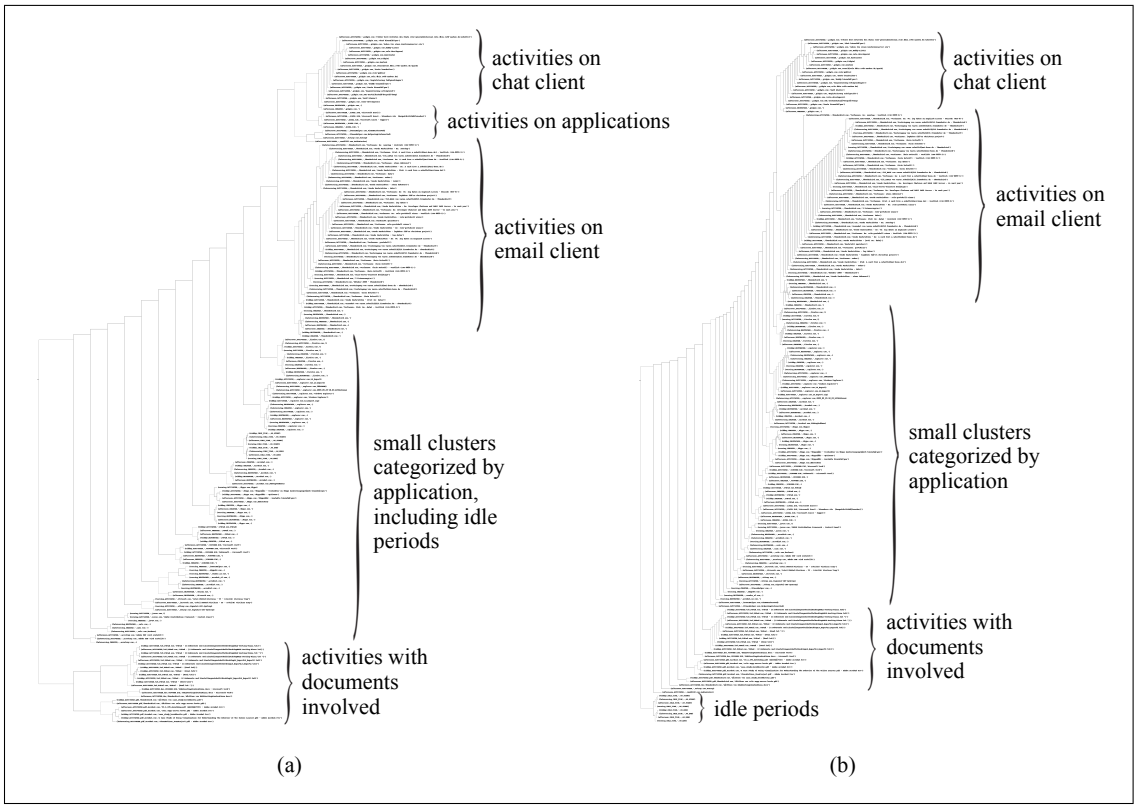


Abbildung 6.10: Ergebnisse des hierarchischen Clusterings mit Gewichtung

Finden ähnlicher Verhaltensweisen

Die Aufgabe des Algorithmus *equal_pref* ist es, zwei Datenmengen unterschiedlicher Benutzer auf Basis der Grenzregion miteinander zu Vergleichen, um letztlich zu entscheiden, ob ähnliche Verhaltensmuster in den zwei Mengen enthalten sind oder nicht. Die praktische Anwendung dieses Verfahrens auf die gegebenen Produktivdaten hat dabei gezeigt, dass dieses Vorhaben prinzipiell durch *equal_pref* abgedeckt wird und solche Muster somit extrahierbar sind. Dennoch konnten im Rahmen der praktischen Analyse zwei wesentliche Schwächen des Algorithmus aufgedeckt werden: Zum einen ist aufgefallen, dass *equal_pref* bei manchen Experimenten zwar nach dem Expertenwissen α richtige Ausgaben liefert, diese jedoch nicht mit den realen Präferenzen der Benutzer im analysierten Datenintervall korrespondieren. Der Grund für solche Fehlinterpretationen liegt an teilweise ungünstigen Datenverteilungen in der Grenzregion, so dass der Anteil an ähnlichen Aktivitäten des einen Nutzers sehr viel kleiner sein kann als die äquivalenten Aktionen des anderen. Reale Messungen unter diesen Bedingungen haben einen relativen Anteil von 7% der Aktivitäten des einen Benutzers und 93% der Aktivitäten des anderen Benutzers in der Grenzregion ergeben, so dass nicht mehr von einem ähnlichen Verhalten gespro-

chen werden kann, obwohl in Summe die Grenzregion den Schwellwert α übersteigt. Die zweite grundliegende Schwäche von *equal_pref* liegt an der Repräsentation des Expertenwissens. Der Parameter α musste für fast alle Experimente individuell angepasst werden, so dass kein stabiler Schwellwert aus der Analyse dieses Algorithmus mit CAM hervor geht.

6.4 Bewertung

Für die entwickelten Verfahren aus Kapitel 4 und 5 hat sich im Bezug auf die Performanz PostgreSQL 8.3 als relationales Datenbanksystem bewehrt. Mit den von ihm zur Verfügung gestellten Basisalgorithmen war es durch die generischen Implementierungen der Methoden *represent* und *ind* möglich die grundlegenden Strukturen großer Datenbankrelationen in weniger als einer Minute zu extrahieren. Dies ist im Hinblick auf das hohe analysierte Datenvolumen von $9,8 \cdot 10^6$ Tupeln ($\approx$ 1,2 GByte) und der zur Verfügung stehenden Hardware als effizient zu bewerten. Diese Ergebnisse waren allerdings nur dadurch erzielbar, dass die für die Berechnungen genutzte Hash-Tabelle stets im Hauptspeicher gehalten werden konnte. In anderen Fällen griff PostgreSQL auf weniger schnelle Basisalgorithmen zurück, deren Hauptidee die Sortierung zugrunde liegt. Alle weiteren Implementierungen der konzipierten und evaluierten Verfahren *upper*, *boundary*, *outside*, *equal_pref* sowie der Klassifikator h konnten teilweise mit und teilweise ohne Indexunterstützung im Wesentlichen an das gute Laufzeitverhalten von *represent* bzw. *ind* anknüpfen. Die Ausnahme stellt dabei die Umsetzung des hierarchischen Cluster-Algorithmus *clusterRel* dar. Trotz einiger Anstrebungen, die Performanz des Verfahrens zu verbessern hat sich unter den Bedingungen der empirischen Laufzeitverhalten ergeben, dass dieser Algorithmus für schnelle praktische Auswertungen ungeeignet ist. Somit kann die aktuelle Implementierung nur für langfristige Analysen im Sinne des *Knowledge Discovery in Databases* für das CAPLE-Projekt Verwendung finden, obwohl mit dedizierter Hardware und deutlich mehr Hauptspeicher zu erwarten ist, bessere Ergebnisse zu erhalten.

Die detaillierte Analyse der Umsetzung des Klassifikators h hat im Bezug auf die Qualität der Klassifikation gezeigt, dass mit groben Äquivalenzklassen Reproduktionen von Tagesaktivitäten verschiedener Benutzer aus den gegebenen Produktivdaten extrahierbar sind. Bei steigender Attributgröße hat sich jedoch zunehmend bestätigt, dass die resultierenden Abstufungen in den Klassen der ursprünglichen Partition dazu führen kann, sehr unsichere Muster zu extrahieren. Zusätzlich ist

bei Testläufen zum Finden guter Schwellwerte α und β ihr relativer Bezug zu der Gesamtanzahl der analysierten Aktivitäten aufgefallen. Mit schwacher Varianz lagen die durchschnittlichen Werte für α und β bei 17,4% respektive 4,5% für die gegebene Attributmenge $\{fName, tName\}$ auf den Produktivdaten. Daraus lässt sich ableiten, dass eine relative Betrachtung dieser Werte als sinnvoll zu erachten ist. Dieser Qualitätsgewinn würde aber den Mehraufwand zur Folge haben, den zu analysierenden Datenbestand vor der Ausführung des Algorithmus mindestens einmal zu durchlaufen, was sich bei großen Relationen als sehr kostenintensiv erweisen kann.

Die Analyse der Umsetzung des agglomerativen hierarchischen Cluster-Verfahrens *clusterRel* hat auf den Produktivdaten mit der natürlichen Distanz gezeigt, dass gerade im Zusammenhang mit der Single-Linkage-Strategie zunächst unähnliche Gruppen von Aktivitäten in Folgeiterationen schneller in ein neues Cluster fallen als beim Complete-Linkage. Dieser sogenannte *Chaining Effect* hat zur Folge, dass heterogene Gruppen von Aktivitäten entstehen [GRS98]. Im direkten Vergleich hat das kompetitivere Complete-Linkage auf der natürlichen Distanz bessere Ergebnisse erzielt, so dass weitestgehend homogene Gruppierungen entstanden sind. Insgesamt waren die Abstände zwischen starken Gruppen von Aktivitäten für beide verwendeten Strategien jedoch sehr klein. Das führt letztlich zu dem Schluss, dass das hierarchische Clustering, mit dem eingeführten natürlichen Distanzmaß auf den gegebenen CAM, als instabil zu bewerten ist. Im Kontrast zu dieser Einschätzung steht die Verwendung der Umsetzung von *clusterRel* mit subjektiv gewichteten Attributen. Mit diesem Konzept hat sich gezeigt, dass nicht nur die zuvor dargelegte Schwäche beim Gruppieren weitestgehend überwunden werden, sondern auch das Clustering je nach Gewichtung gesteuert werden kann. So können beispielsweise Aktivitäten nach Tageszeiten, verwendeten Applikationen oder Dokumenten zusammengefasst werden.

Der konzipierte Algorithmus *equal_pref* geht zum Finden ähnlicher Benutzerinteressen von einer Gleichverteilung der Daten in der Grenzregion aus. Ist diese gewährleistet, haben sich in der Praxis gute Ergebnisse bestätigt. In ungünstigen Fällen, bei denen sich die Verteilung in der Grenzregion zweier Benutzer als einseitig beschreiben lässt, hat *equal_pref* jedoch beliebig schlechte Ausgaben geliefert. Die zweite Schwäche des Verfahrens ist der automatisierte Betrieb. Mit dem derzeitig verwendeten Absolutwert α lässt sich nur schwer ermitteln, ab welcher Schwelle zwei Benutzerinteressen als ähnlich zu bewerten sind. Aus diesem und dem zuvor genannten Grund sind weitere Modifikationen und Analysen für *equal_pref* außerhalb dieser Arbeit notwendig.

7 Schlusswort und Ausblick

Diese Arbeit wurde von der Idee angetrieben, die grundlegenden Konzepte der *Rough Set Theory* mit den effizienten Basisalgorithmen praxiserprobter objektrelationaler Datenbanksysteme zu vereinen. Darauf aufbauend wurden weitere Verfahren entwickelt, um wissenschaftliche Auswertungen von Benutzeraktivitäten im Rahmen des Forschungsprojekts *Context and Attention in Personalized Learning Environments* am Fraunhofer-Institut für Angewandte Informationstechnik FIT zu unterstützen. Diese Analysealgorithmen wurden letztlich als server-seitige Prozeduren in das konkrete Datenbanksystem PostgreSQL integriert. So konnte ein erweiterbares objektrelationales Gesamtsystem entstehen, welches als datennahe Auswertungsplattform im Sinne des *Knowledge Discovery in Databases* Verwendung finden kann.

Die Evaluierung hat im Detail ergeben, dass der mengentheoretische Entwurf der Analysemethoden im Bezug auf ihre praktische Effizienz weitestgehend positiv zu beurteilen ist. Somit haben sich die Erwartungen in diesem Zusammenhang bestätigt. Hinsichtlich der Qualität der extrahierten Muster konnten kleinere Schwächen aufgedeckt werden, die ihren Ursprung in der Entwurfsphase haben und leicht zu beheben sind. Größtenteils konnten jedoch gute Ergebnisse - insbesondere mit den Konzepten der Ununterscheidbarkeit und der Unsicherheit aus der RST - erzielt werden, so dass insgesamt nützliche Werkzeuge für das Data-Mining im Kontext des CAPLE-Projekts entstanden sind.

Das gesamte Potential dieses Systems ist jedoch nicht vollständig ausgeschöpft. Es lässt sich beispielsweise auf Methodenebene ein erweiternder mengenorientierter Klassifikator vorstellen, der in einem ersten Schritt verallgemeinernde Hypothesen generiert. In einer zweiten Ausbaustufe können dann ergänzende Methoden aus dem maschinellen Lernen wie *Bagging* oder *Boosting* zum Einsatz kommen, so dass noch stärkere Voraussagen bezüglich zukünftiger Aktivitäten eines Benutzers durchgeführt werden können. Zusätzlich lässt sich für den alltäglichen Umgang mit den vom System zur Verfügung gestellten Analysemethoden ein nahtloserer Übergang zu SQL überlegen, mit dem es erlaubt ist, flexibel und komfortabel SQL-Befehle an die Prozeduren zu übergeben. Im Hinblick auf die Performanz kann das System noch

weiteren Verbesserungen unterzogen werden. Zum Beispiel lassen sich auf konzeptioneller Ebene Bestrebungen dahingehend unternehmen, die algebraischen Ausdrücke so zu optimieren, dass weniger kostenintensive Join-Operationen zur Anwendung kommen. Für Methoden, die aus theoretischer Perspektive kein weiteres Verbesserungspotenzial besitzen, aber parallelisierbar sind, kann versucht werden Berechnungen adäquat auf ein *Cloud* zu verteilen. Damit stünde enorme Rechenleistung für die Bearbeitung von Anfragen zur Verfügung, was jedoch den Nachteil einer redundanten Datenspeicherung zur Folge hätte. Die frei verfügbare Middleware-Software *pgpool-II* liefert eine solche Funktionalität speziell für PostgreSQL. Eine andere praktische Möglichkeit zur Verbesserung der Performanz bietet das Vorberechnen von Aggregationen durch *Trigger*. Auf diese Weise kann gewährleistet werden, dass bestimmte Berechnungen für die Benutzeranalysen ganz wegfallen. Der letztgenannte Punkt weist insbesondere in Richtung Data-Warehousing.

Aufgrund des aufgezeigten Potenzials, qualitativere und noch effizientere Auswertungen durchführen zu können, ist zu erwarten, dass im CAPLE-Projekt weitere wissenschaftliche Arbeiten an die Ergebnisse dieser Master-Thesis anknüpfen, um letztlich den gesamten Rahmen an Möglichkeiten im Kontext der wissenschaftlichen Analyse von Benutzeraktivitäten auszuschöpfen.

Abkürzungs- und Symbolverzeichnis

ACID	Atomicity Consistency Isolation Durability
ASCII	American Standard Code for Information Interchange
BLOB	Binary Large Object
CAM	Contextualized Attention Metadata
CAMS	Contextualized Attention Metadata Schema
CAPLE	Context and Attention in Personalized Learning Environments
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
DM	Data-Mining
DS	Decision-System
ERD	Entity-Relationship-Diagramm
FIT	Fraunhofer-Institut für Angewandte Informationstechnik
GUID	Globally Unique Identifier
IS	Informationssystem
KDD	Knowledge Discovery in Databases
ML	Maschinelles Lernen
NIST	National Institute of Standards and Technology
NP	Nichtdeterministisch polynomiell
OO	Objektorientierung
OR	Objektrelational
ORM	Objektrelationales-Mapping
PL/pgSQL	Procedural Language/PostgreSQL SQL
PL/SQL	Procedural Language/SQL
RDBS	Relationales Datenbanksystem
ROSE	Rough Set Data Explorer
RSDM	Rough Set Data Miner
RSES	Rough Set Exploration System
RSL	Rough Set Library
RST	Rough Set Theory
SQL	Structured Query Language

UML	Unified Modeling Language
VPRS	Variable Precision Rough Sets
XML	Extensible Markup Language

$\mathbb{B}$	Menge der Bool'schen Werte $\{0, 1\}$
$\mathbb{N}$	Menge der Natürlichen Zahlen $\{1, 2, 3, ...\}$
$\mathbb{N}_0$	Menge der Natürlichen Zahlen mit der Null $\{0, 1, 2, ...\}$
$\emptyset, \{\}$	Leere Menge
$\mathcal{P}(M)$	Potenzmenge der Menge M
$\forall$	Allquantor
$\exists$	Existenzquantor
$\wedge$	Logische Und-Verknüpfung
$\vee$	Logische Oder-Verknüpfung
$\perp$	Undefiniertheitssymbol, Nullwert, -marke
$\|M\|$	Kardinalität der Menge M
$f : M \rightarrow V$	Abbildungsvorschrift f aus der Menge M in V
$\sim$	Äquivalenzrelation
$[x]_\sim$	Äquivalenzklasse von x bez. $\sim$
$U/\sim$	Partition, Menge aller Äquivalenzklassen bez. U und $\sim$
$x \in M$	x ist ein Element der Menge M
$x \equiv_\sim y$	Äquivalenz der Objekte x, y bez. $\sim$
$N \subseteq M$	N ist eine Teilmenge von M
$N \subset M$	N ist eine echte Teilmenge von M
$N \cup M$	Mengentheoretische Vereinigung der Mengen N und M
$N \cap M$	Mengentheoretischer Schnitt der Mengen N und M
$M - N$	Mengentheoretische Differenz der Mengen M und N
$M \times N$	Kartesisches Produkt der Mengen M und N
$R \subseteq M \times N$	Relation zwischen den Mengen M und N
σ^2	Varianz, Streuung
$\overline{x}_{arith}$	arithmetischer Mittelwert, Durchschnitt
$\mathbb{U}$	Universum, Menge von Objekten in einem Informationssystem
X	Zielkonzept, $X \subseteq \mathbb{U}$
A	Attributmenge eines Informationssystems
$\mathcal{A} = (\mathbb{U}, A)$	Informationssystem mit $\mathbb{U}$ und A
$\mathcal{A}_d$	Entscheidungssystem mit $\mathbb{U}, A$ und Entscheidungsattributen
$IND_\mathcal{A}(B)$	Ununterscheidbarkeitsrelation bez. eines $\mathcal{A} = (\mathbb{U}, A)$ und $B \subseteq A$
$\underline{X}_B$	B-untere Approximation von X bez. $IND_\mathcal{A}(B)$
$\overline{X}_B$	B-obere Approximation von X bez. $IND_\mathcal{A}(B)$

$\overline{\underline{X}}_B$	B-Grenzregion von X bez. $IND_{\mathcal{A}}(B)$
$\mu_X^B(x)$	Unexakte Zugehörigkeitsfunktion bez. $X \subseteq \mathbb{U}, x \in \mathbb{U}, B \subseteq A, \mathcal{A}$
α_X^B	Exaktheit von $\underline{X}_B$ und $\overline{X}_B$ bez. $X \subseteq \mathbb{U}, B \subseteq A, \mathcal{A}$
$POS_B(E)$	Positive-Region, Vereinigung aller unteren Approximationen
$BND_B(E)$	Boundary-Region, Vereinigung aller Grenzregionen
$NEG_B(E)$	Negative-Region
$\gamma_B(E)$	Grad der Abhängigkeit zwischen Attributmenge B und E
$\sigma_B^E(a)$	Signifikanz des Attributs $a \in B$ bez. Attributmenge B und E
$\mathcal{R}$	Datenbankrelation bzw. Datentabelle
$\sigma_\phi(\mathcal{R})$	Selektion mit dem Prädikat ϕ auf der Relation $\mathcal{R}$
$\Pi_B(\mathcal{R})$	Projektion der Relation $\mathcal{R}$ auf die Attribute B
$\gamma_{F;B}(\mathcal{R})$	Gruppierung von $\mathcal{R}$ nach den Attributen B unter Anwendung von F
$\mathcal{R} \bowtie \mathcal{R}'$	Natürlicher Verbund von $\mathcal{R}$ und $\mathcal{R}'$
$\mathcal{R} \bowtie_\phi \mathcal{R}'$	Verschmelzung von $\mathcal{R}$ und $\mathcal{R}'$ auf Basis des Prädikats ϕ
$\mathcal{R} ⟕_\phi \mathcal{R}'$	Linker Verbund von $\mathcal{R}$ und $\mathcal{R}'$ auf Basis des Prädikats ϕ
$\mathcal{R} ⟖_\phi \mathcal{R}'$	Rechter Verbund von $\mathcal{R}$ und $\mathcal{R}'$ auf Basis des Prädikats ϕ
$\rho_{b \leftarrow a}(\mathcal{R})$	Umbenennungsfunktion, Umbenennung des Attributs a in b

Abbildungsverzeichnis

Listings

Literaturverzeichnis

[Alp08] Alpaydin, E. : *Maschinelles Lernen.* Oldenbourg Verlag, 2008

[Alv98] Alvarez, E. : *Rough Analysis.* `http://www.lsi.upc.edu/~ealvarez/rough.html`. Version: 1998, Abruf: 19.08.2009

[Bac96] Bacher, J. : *Clusteranalyse. Anwendungsorientierte Einführung.* 2. Auflage. Oldenbourg Verlag, 1996

[BC03] Begg, C. ; Connolly, T. : *Database Systems: A Practical Approach to Design, Implementation and Management.* 3. Auflage. Pearson Studium, 2003

[BC06] Beekmann, F. ; Chamoni, P. : Verfahren des Data Mining. In: Chamoni, P. ; Gluchowski, P. : *Analytische Informationssysteme.* 3. Auflage. Springer Verlag, 2006, S. 263–282

[Bee09] Beer, F. : Entwicklung eines Kommunikationsframeworks im Rahmen des CAPLE-Projekts / Fraunhofer-Institut für Angewandte Informationstechnik FIT, Forschungsgruppe: CAPLE. 2009. – Technical Report (unveröffentlicht)

[BEPW08] Backhaus, K. ; Erichson, B. ; Plinke, W. ; Weiber, R. : *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung.* 12. Auflage. Springer Verlag, 2008

[Beu01] Beutelspacher, A. : *Lineare Algebra.* 5. Auflage. Vieweg Verlag, 2001

[Bjo97] Bjorvand, A. : Rough Enough - Software Demonstration. In: Sydow, A. : *Proceedings of the Fifteenth IMACS World Congress on Artificial Intelligence and Computer Science* Bd. 4, Wissenschaft & Technik Verlag, 1997, 607-612

[BK08] Bean, C. ; Kambhampati, C. : Autonomous Clustering using Rough Set Theory. In: *International Journal of Automation and Computing* 5 (2008), Nr. 1, S. 90–102

[BL04] Berry, M. ; Linoff, G. : *Data Mining Techniques.* 2. Auflage. Wiley & Sons Verlag, 2004

[BMA06] Banerjee, M. ; Mitra, S. ; Anand, A. : Feature Selection using Rough Sets. In: *Studies in Computational Intelligence* 16 (2006), S. 3–20

[Bri05] Brill, M. : *Mathematik für Informatiker.* 2. Auflage. Hanser Verlag, 2005

[BS05] Bazan, J. ; Szczuka, M. : The Rough Set Exploration System. In: Peters, J. ; Skowron, A. ; Albaba, D. van: *Transactions on Rough Sets III* Bd. 3400. Springer Verlag, 2005, S. 37–56

[CABG07] Caballero, Y. ; Alvarez, D. ; Bello, R. ; Garcia, M. : Feature Selection Algorithms using Rough Set Theory. In: *Seventh International Conference on Intelligent Systems Design and Applications*, 2007, S. 407–411

[CCF+09] Chakrabarti, S. ; Cox, E. ; Frank, E. ; Güting, R. ; Han, J. ; Jiang, X. ; Kamber, M. ; Lightstone, S. ; Nadeau, T. ; Neapolitan, R. ; Pyle, D. : *Data Mining: Know it All.* Morgan Kaufmann Verlag, 2009

[Chi68] Childs, D. L.: Feasibility of a set-theoretical Data Structure. In: *Proceeding of the Fall Joint Computer Conference*, 1968, S. 557–564

[Cod70] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks. In: *Communications of the ACM* Bd. 13, 1970, S. 377–387

[Cod79] Codd, E. F.: Extending the Database Relational Model to Capture More Meaning. In: *ACM Transactions on Database Systems* Bd. 4, 1979, S. 397–434

[CPC08] Chiu, D. ; Pan, Y. ; Chang, W. : Using Rough Set Theory to Construct e-Learning FAQ Retrieval Infrastructure. In: *The First IEEE International Conference on Ubi-Media Computing and Workshops*, IEEE Computer Society, 2008

[Dat05] Date, C. : *Database in Depth - Relational Theory for Practitioners.* O'Reilly Verlag, 2005

[DG97] Düntsch, I. ; Gediga, G. : The Rough Set Engine GROBIAN. In: Sydow, A. : *Proceedings of the Fifteenth IMACS World Congress on Artificial Intelligence and Computer Science* Bd. 4, Wissenschaft & Technik Verlag, 1997, S. 613–618

[Düs06] Düsing, R. : Knowledge Discovery in Databases. In: Chamoni, P. ; Gluchowski, P. : *Analytische Informationssysteme.* 3. Auflage. Springer Verlag, 2006, S. 241–262

[EH09] Eisentraut, P. ; Helmle, B. : *PostgreSQL - Administration.* O'Reilly Verlag, 2009

[Eis03] Eisentraut, P. : *PostgreSQL - Das offizielle Handbuch.* Mitp-Verlag, 2003

[EN05] Elmasri, R. ; Navathe, S. : *Grundlagen von Datenbanksystemen.* 3. Auflage. Addison-Wesley, 2005

[ES00] Ester, M. ; Sander, J. : *Knowledge Discovery in Databases.* Springer Verlag, 2000

[FBRS96] Fernandez-Baizán, M. C. ; Ruiz, E. M. ; Sánchez, J. M. P.: Integrating RDMS and Data Mining Capabilities using Rough Sets. In: *Proceedings of the Sixth International Conference. Information Processing Management of Uncertainty in Knowledge-Based Systems* (1996), S. 1439–1445

[FBRSP98] Fernandez-Baizán, M. C. ; Ruiz, E. M. ; Sánchez, J. M. P. ; Pastrana, B. P.: RSDM System. In: *Bulletin of International Rough Set Society* 2 (1998), S. 21–24

[FBRW98] Fernandez-Baizán, M. C. ; Ruiz, E. M. ; Wasilewska, A. : A Model of RSDM Implementation. In: Polkowski, L. ; Skowron, A. : *International Conference on Rough Sets and Current Trends in Computing*, Springer Verlag, 1998, S. 186–193

[FHS96] Fayyad, U. ; Haussler, D. ; Stolorz, P. : KDD for Science Data Analysis: Issues and Examples. In: *Proceedings of Second International Conference on Knowledge Discovery and Data Mining*, AAAI Press, 1996, S. 50–56

[FP05] Feuerstein, S. ; Pribyl, B. : *Oracle PL/SQL Programming.* 4. Auflage. O'Reilly Verlag, 2005

[FPSS96] Fayyad, U. ; Piatetsky-Shapiro, G. ; Smyth, P. : From Data Mining to Knowledge Discovery in Databases. In: *AI Magazine* 17 (1996), S. 37–54

[FR98] Fraley, C. ; Raftery, A. : How Many Clusters? Which Clustering Methods? Answers Via Model-Based Cluster Analysis / Department of Statistics - University of Washington. 1998. – Technical Report No. 329

[GB05] Grzymala-Busse, J. : LERS - A Data Mining System. In: Maimon, O. ; Rokach, L. : *Data Mining and Knowledge Discovery Handbook.* Springer Verlag, 2005, S. 1347–1351

[GBGB05] Grzymala-Busse, J. ; Grzymala-Busse, W. : Handling missing Attribute Values. In: Maimon, O. ; Rokach, L. : *Data Mining and Knowledge Discovery Handbook.* Springer Verlag, 2005, S. 37–57

[GM98] Grimmer, U. ; Mucha, H.-J. : Datensegmentierung mittels Clusteranalyse. In: Nakhaeizadeh, G. : *Data Mining - Theoretische Aspekte und Anwendungen.* Physica Verlag, 1998, S. 109–141

[GRS98] Guha, S. ; Rastogi, R. ; Shim, K. : CURE: An Efficient Clustering Algorithm for Large Databases. In: *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1998, S. 73–84

[GS94] Gawrys, M. ; Sienkiewicz, J. : RSL - The Rough Set Library - Version 2.0 / Institute of Computer Science - Warsaw University of Technology. 1994. – Research Report

[Hac05] Hachenberer, D. : *Mathematik für Informatiker.* Pearson Studium, 2005

[HCHZ02] Hu, X. ; Cercone, N. ; Han, J. ; Ziarko, W. : A Generalized Rough Sets Model. In: Lin, T. ; Yao, Y. Y. ; Zadeh, L. A.: *Data Mining, Rough Sets and Granular Computing*, Physica Verlag, 2002, S. 447–460

[HLH03] Hu, X. ; Lin, T. Y. ; Han, J. : A new Rough Set Model based on Database Systems. In: Wang, G. ; Qing, L. ; Yiyu, Y. ; Skowron,

A. : *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing* Bd. 2639, Springer Verlag, 2003, S. 114–121

[HQB08] Haijun, X. ; Qi, Z. ; Baoyi, W. : Rough Set Page Recommendation Algorithm Based on Information Entropy. In: *International Conference on Computer Science and Software Engineering*, 2008, S. 735–738

[HSS07] Heuer, A. ; Saake, G. ; Sattler, K.-U. : *Datenbanken - Konzepte und Sprachen.* 3. Auflage. Mitp-Verlag, 2007

[HTOH01] Hirano, S. ; Tsumoto, S. ; Okuzaki, T. ; Hata, Y. : A Clustering Method for Nominal and Numerical Data Based on Rough Set Theory. In: *Proceedings of the Joint JSAI 2001 Workshop on New Frontiers in Artificial Intelligence*, Springer Verlag, 2001, S. 400–405

[Hua98] Huang, Z. : Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. In: *Data Mining and Knowledge Discovery* Bd. 2. Kluwer Academic Publishers, 1998, S. 283–304

[JCJÖ92] Jacobson, I. ; Christerson, M. ; Jonsson, P. ; Övergaard, G. : *Object-Oriented Software Engineering.* Addison-Wesley, 1992

[Kan02] Kantardzic, M. : *Data Mining - Concepts, Models, Methods, and Algorithms.* Wiley & Sons Verlag, 2002

[KE04] Kemper, A. ; Eickler, A. : *Datenbanksysteme.* 5. Auflage. Oldenbourg Verlag, 2004

[KK04] Kumar, S. ; Krishna, P. R.: Clustering Web Transactions using Rough Approximation. In: *Fuzzy Sets and Systems* 148 (2004), Nr. 1, S. 131–138

[KPPS99] Komorowski, J. ; Pawlak, Z. ; Polkowski, L. ; Skowron, A. : Rough Sets: A Tutorial. In: Pal, S. K.; Skowron, A. : *Rough Fuzzy Hybridization: A New Trend in Decision Making*, Springer Verlag, 1999, S. 3–98

[KR87] Kaufmann, L. ; Rousseeuw, P. : Clustering by means of Medoids. In: Dodge, Y. : *Statistical Data Analysis based on the L1 Norm.* Elsevier Verlag, 1987, S. 405–416

[L3S07] L3S Research Center: *User Activity Logger documentation.* `http://pas.kbs.uni-hannover.de/Documentation/UAL/Index.html`. Version: 2007, Abruf: 28.08.2009

[LC96] Lin, T. Y.; Chen, R. : Supporting Rough Set Theory in Very Large Database Using ORACLE RDBMS. In: *Proceedings of the 1996 Asian. Soft Computing in Intelligent Systems and Information Processing*, 1996, S. 332–337

[Lin96] Lin, T. Y.: Rough Set Theory in Very Large Databases. In: *Symposium on Modeling, Analysis and Simulation* Bd. 2, 1996, S. 936–941

[Lin97] Lin, T. Y.: An Overview of Rough Set Theory from the Point of View of Relational Databases. In: *Bulletin of International Rough Set Society* Bd. 1, 1997, S. 30–34

[LKM+85] Lorie, R. ; Kim, W. ; McNabb, D. ; Plouffe, W. ; Meier, A. : Supporting Complex Objects in a Relational System for Engineering Databases. In: *Query Processing in Database Systems.* 1985, S. 145–155

[Mac67] MacQueen, J. : Some Methods for Classification and Analysis of Multivariate Observations. In: LeCam, L. ; Neyman, J. : *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability* Bd. 1, 1967, S. 281–297

[Mit97] Mitchell, T. : *Machine Learning.* Mcgraw-Hill Higher Education, 1997

[Mod93] Modrzejewski, M. : Feature Selection using Rough Sets Theory. In: *Proceedings of the European Conference on Machine Learning* Bd. 667, Springer Verlag, 1993, S. 213–226

[Mom01] Momjian, B. : *PostgreSQL - Einführung und Konzepte.* Addison-Wesley, 2001

[MP84] Marek, W. ; Pawlak, Z. : Rough Sets and Information Systems. In: *Fundamenta Informaticae* Bd. 17, 1984, S. 105–115

[MR05] Maimon, O. ; Rokach, L. : Clustering Methods. In: Maimon, O. ; Rokach, L. : *Data Mining and Knowledge Discovery Handbook.* Springer Verlag, 2005, S. 321–352

[MU03] Matthiessen, G. ; Unterstein, M. : *Relationale Datenbanken und SQL*. 3. Auflage. Addison-Wesley, 2003

[MW03] Meier, A. ; Wüst, T. : *Objektorientierte und objektrelationale Datenbanken - Ein Kompass für die Praxis*. 3. Auflage. Dpunkt Verlag, 2003

[NWD06] Najjar, J. ; Wolpers, M. ; Duval, E. : Attention Metadata: Collection and Management. In: *WWW2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data*, 2006

[ØK97] Øhrn, A. ; Komorowski, J. : ROSETTA - A Rough Set Toolkit for Analysis of Data. In: *Proceedings of the Third International Joint Conference on Information Sciences*, 1997, S. 403–407

[Paw] Pawlak, Z. : *Rough Set Elements (1)*. `http://chc60.fgcu.edu/images/articles/RoughSetElements1.pdf`, Abruf: 04.05.2009

[Paw81] Pawlak, Z. : Information Systems Theoretical Foundations. In: *Information System* 6 (1981), Nr. 3, S. 205–218

[Paw82] Pawlak, Z. : Rough Sets. In: *International Journal of Computer and Information Science* Bd. 11, 1982, S. 341–356

[Paw91] Pawlak, Z. : *Rough Sets - Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, 1991

[Pet05] Petersohn, H. : *Data Mining: Verfahren, Prozesse, Anwendungsarchitektur*. Oldenbourg Verlag, 2005

[PGBSZ95] Pawlak, Z. ; Grzymala-Busse, J. ; Slowinski, R. ; Ziarko, W. : Rough Sets. In: *Communications of the ACM* Bd. 38, 1995, S. 88–95

[Pos09] PostgreSQL Global Development Group: *PostgreSQL 8.3.8 Documentation: PL/pgSQL - SQL Procedural Language*. `http://www.postgresql.org/docs/8.3/`. Version: 2009, Abruf: 22.10.2009

[PS94] Pawlak, Z. ; Skowron, A. : The Rough Membership Function. In: *Advances in the Dempster Shafer Theory of Evidence* (1994), S. 251–271

[PSS+98] Predki, B. ; Slowinski, R. ; Stefanowski, J. ; Susmaga, R. ; Wilk, S. : ROSE - Software Implementation of the Rough Set Theory.

In: Polkowski, L. ; Skowron, A. : *Rough Sets and Current Trends in Computing.* Springer Verlag, 1998, S. 605–608

[PTL00] Polkowki, L. ; Tsumoto, S. ; Lin, T. Y.: A Rough Set Perspective on Knowledge Discovery in Information Systems. In: Polkowki, L. ; Tsumoto, S. ; Lin, T. Y.: *Rough Set Methods and Applications. New Developments in Knowledge Discovery in Information Systems*, Physica Verlag, 2000, S. 9–45

[QW09] Qu, Z. ; Wang, X. : Application of Clustering Algorithm and Rough Set in Distance Education. In: *International Workshop on Education Technology and Computer Science* 1 (2009), S. 489–493

[RG02] Ramakrishnan, R. ; Gehrke, J. : *Database Management Systems.* 3. Auflage. Mcgraw-Hill Higher Education, 2002

[SC01] Shen, Q. ; Chouchoulas, A. : Rough Set-based Dimensionality Reduction for Supervised and Unsupervised learning. In: *International Journal of Applied Mathematics and Computer Science* 11 (2001), Nr. 3, S. 583–601

[Sch09] Scherbaum, A. : *PostgreSQL - Datenbankpraxis für Anwender, Administratoren und Entwickler.* Open Source Press, 2009

[Seg08] Segaran, T. : *Kollektive Intelligenz: analysieren, programmieren und nutzen.* O'Reilly Verlag, 2008

[SHS05] Saake, G. ; Heuer, A. ; Sattler, K.-U. : *Datenbanken: Implementierungstechniken.* 2. Auflage. Mitp-Verlag, 2005

[Sip05] Sipser, M. : *Introduction to the Theory of Computing.* 2. Auflage. Thomson Course Technology, 2005

[SM99] Stonebraker, M. ; Moore, D. : *Objektrelationale Datenbanken - Die nächste große Welle.* Hanser Verlag, 1999

[SN04] S.Russell; Norvig, P. : *Künstliche Intelligenz: Ein moderner Ansatz.* 2. Auflage. Pearson Studium, 2004

[SR92] Skowron, A. ; Rauszer, C. : The Discernibility Matrices and Functions in Information Systems. In: Slowinski, R. : *Decision Support by Experience - Application of the Rough Sets Theory*, Kluwer Academic Publishers, 1992, S. 331–362

[SS96] Skowron, A. ; Stepaniuk, J. : Tolerance Approximation Spaces. In: *Fundamenta Informaticae* Bd. 27, 1996, S. 245–253

[SST97] Saake, G. ; Schmitt, I. ; Türker, C. : *Objektdatenbanken: Konzepte, Sprachen, Architekturen.* International Thomson Publishing, 1997

[Swi01] Swiniarski, R. W.: Rough Sets Methods in Featue Reduction and Classification. In: *International Journal of Applied Mathematics and Computer Science* 11 (2001), Nr. 3, S. 565–582

[TB07] Tripathy, H. ; B.K.Tripathy: A Rough Set Approach for Clustering the Data using Knowledge Discovery in World Wide Web for E-Business. In: *Proceedings of the IEEE International Conference on e-Business Engineering*, IEEE Computer Society, 2007, S. 717–722

[TCX07] Tan, S. ; Cheng, X. ; Xu, H. : An efficient global Optimization Approach for Rough Set based Dimensionality Reduction. In: *International Journal of Innovative Computing, Information and Control* Bd. 3, 2007, S. 725–736

[Tre04] Tretyakov, K. : *Machine Learning Techniques in Spam Filtering.* `http://math.ut.ee/~kt/spam/spam.pdf`. Version: Mai 2004, Abruf: 07.06.2009

[TS05] Türker, C. ; Saake, G. : *Objektrelationale Datenbanken.* Dpunkt Verlag, 2005

[TSWH08] Tseng, V. ; Su, J.-H. ; Wang, B.-W. ; Hsiao, C.-Y. : A Novel Recommendation Method Based on Rough Set and Integrated Feature Mining. In: *Proceedings of theThird International Conference on Innovative Computing Information and Control*, IEEE Computer Society, 2008, S. 330–334

[Tür03] Türker, C. : *SQL 1999 und SQL 2003: Objektrelationales SQL, SQLJ und SQL/XML.* Dpunkt Verlag, 2003

[Tur04] Turau, V. : *Algorithmische Graphentheorie.* 2. Auflage. Oldenbourg Verlag, 2004

[UAJ06] Upadhyaya, S. ; Arora, A. ; Jain, R. : Rough Set Theory: Approach for Similarity Measure in Cluster Analysis. In: *Proceedings of the 2006 International Conference on Data Mining*, 2006, S. 353–356

[UHMH05] Urman, S. ; Hardman, R. ; McLaughlin, M. ; Heidenberger, D. : *Oracle Database 10g - PL/SQL Programmierung*. Hanser Verlag, 2005

[WF01] Witten, I. ; Frank, E. : *Data Mining - Praktische Werzeuge und Techniken für das maschinelle Lernen*. Hanser Verlag, 2001

[WNVD07] Wolpers, M. ; Najjar, J. ; Verbert, K. ; Duval, E. : Tracking Actual Usage: the Attention Metadata Approach. In: *Educational Technology and Society* 10 (2007), Nr. 3, S. 106–121

[Wro95] Wroblewski, J. : Finding minimal Reducts using Genetic Algorithms. In: Wang, P. : *Soft Computing at Second Annual Joint Conference on Information Science*, 1995

[Wro98] Wroblewski, J. : Genetic Algorithms in Decomposition and Classification Problems. In: Polkowski, L. ; Skowron, A. : *Rough Sets in Knowledge Discovery: Methodology and Application*, Physica Verlag, 1998, S. 472 – 492

[WV06] Witt, K.-U. ; Vossen, G. : *Grundkurs Theoretische Informatik*. 4. Auflage. Vieweg Verlag, 2006

[YLS07] Yang, N. ; Li, T. ; Song, J. : Construction of Decision Trees based Entropy and Rough Sets under Tolerance Relation. In: Li, T. ; Xu, Y. ; Ruan, D. : *Proceedings of the International Conference on Intelligent Systems and Knowledge Engineering*, 2007

[YY06] Yang, L. ; Yang, L. : Study of Cluster Algorithm based on Rough Set Theory. In: *Proceedings of the Sixth International Conference on Intelligent Systems Design and Application*, IEEE Computer Society, 2006, S. 492–496

[Zia93] Ziarko, W. : Variable Precision Rough Set Model. In: *Journal of Computer and System Science* 46 (1993), S. 39–59

[ZY04] Zhang, M. ; Yao, J. : A Rough Sets based Approach to Feature Selection. In: *Annual Meeting of the North American Fuzzy Information Processing Society* Bd. 1, IEEE, 2004, S. 434–439