Identifizierung von Realwelt-Objekten in multiplen Datenbanken.

Von der Fakultät Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
(Dr. rer. nat.)

genehmigte Dissertation

vorgelegt von

Diplom–Mathematiker
Mattis Neiling, geb. Meyer

geboren am 7. Juni 1969 in Eisenach

Gutachter: Prof. Dr. Hans-Joachim Lenz

Gutachter: Prof. Dr. Bernhard Thalheim

Gutachter: Prof. Dr. Klaus-Dieter Schewe

Tag der mündlichen Prüfung: 2. Februar 2004

Mattis Neiling

Identifizierung von Realwelt-Objekten in multiplen Datenbanken.

Zusammenfassung

Die Daten von Realwelt-Objekten können auf mehrere Datenbanken verteilt sein. Wie läßt sich herausfinden, welche der Daten sich auf dieselben Realwelt-Objekte beziehen, wenn kein globaler und konsistenter Identifizierer existiert?

Es wird ein allgemeines Modell für die Objektidentifizierung dargestellt, das aus den Schritten Konversion, Vergleich und Klassifikation besteht. Es umfaßt zudem: (1) Identifizierungskonzepte, (2) Softwarearchitektur, (3) Charakteristika der Datenqualität, (4) eine Vorauswahlmethode, die die Effizienz für große Datenbanken sicherstellt (unter Verwendung von Indexstrukturen) und (5) eine Spezifikation für die Evaluation von Verfahren, einschließlich Stichprobenziehung und Qualitätskriterien.

Wir bewerteten mehrere Verfahren mit Wohnungs-, Adreß- und Bibliotheksdaten. Wesentliche Ergebnisse sind, daß die Skalierbarkeit durch die verwandte Vorauswahlmethode und deren Umsetzung bestimmt ist sowie daß das Entscheidungsbaumverfahren eine höhere Korrektheit erreichte und robuster war als Record Linkage.

Mattis Neiling

Identification of Real-World Objects in Multiple Databases

Summary

Object Identification is essential where real-world objects data are distributed over multiple databases without any global and consistent identifier.

We present a generic object identification framework, consisting of three successive steps: Conversion, Comparison, and Classification. In addition, the framework covers: (1) concepts for identification, (2) its software architecture, (3) data quality characteristics, (4) a preselection technique that ensures efficiency for large databases (incorporating suitable index structures), and (5) a prescription for evaluation, including sampling and quality criteria. Based on the framework, methods can be specified, implemented and evaluated w.r.t. to the requirements of an application.

We evaluated several methods on real data. One main result is that scalability is determined by the applied preselection technique and its implementation. As another result we can state that Decision Tree Induction achieves better correctness and is more robust than Record Linkage.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Notation

An dieser Stelle findet man die in dieser Arbeit verwandte Schreibweise der mathematischen Zeichen und anderer abkürzender Bezeichnungen. Es wurde versucht, eine intuitive und einheitliche Notation zu finden. Dieses umzusetzen war nicht immer ganz einfach — in den berührten Teilgebieten der Informatik und Mathematik werden seit Jahren Bezeichnungskonventionen gepflegt, die gezwungenermaßen dasselbe Alphabet verwenden. Auch in dieser Hinsicht ist diese Arbeit als ein Versuch einer *Integration* zu werten — einer Vereinheitlichung der verschiedenen Schreibweisen. Es wurde Wert darauf gelegt, eingebürgerte Bezeichnungen auch in dieser Arbeit zu verwenden.

Zeichen	**Bedeutung**	**Seite**
$\mathbb{N}, \mathbb{Z}, \mathbb{R}$	die natürlichen, ganzen und reellen Zahlen	
$\mathbb{N}^k, \mathbb{R}^k$	das k-fache Kreuzprodukt von $\mathbb{N}$ bzw. $\mathbb{R}$	
p, p_i	reellwertige Skalare (z.B. diskrete Wahrscheinlichkeiten)	
$i, j, k, l, \imath$	Indizes, natürliche Zahlen	
d, m, n	Dimension eines Raumes, Kardinalität einer Menge	
I	Indexmenge	
N, N_i	Stichprobengröße	
A, B	Datenbanken (als Mengen von Elementen aufgefaßt), im II. Teil universale Relationen	23 103
A_i, A'	Teilmengen	
a, b, a_i, b_i	Elemente einer Menge, Instanzen (Tupel) einer Datenbank	22
X_i, Y_i	Einzelne Attribute eines Typs	
X, X^C, X', Y	Attributmengen, z.B $X = (X_1, \ldots, X_k)$	21
x, y, z	Vektoren, z.B. Attributwerte $x = X(a)$	
x_i	i-te Komponente des Vektors x	
$\alpha, \beta, \gamma, \eta, \kappa, \theta$	reellwertige, nicht-negative Skalare	
$\nu, \tau, \ell, \nu_i, \tau_i$	Typen eines Typsystems	20
$\lambda, \hat{\lambda}$	Likelihood-Quotient beim Record Linkage	76
δ, δ_i	Entscheidungsfunktionen auf einer Menge M	50
σ, σ_i	Selektoren	124
f, f_i	binäre Funktionen, speziell Vergleichsfunktionen	50
g, h	unäre Funktionen, speziell Transformationsfunktionen	49
R, r	Vergleichsraum und Vergleichsvektor	50
R_i, r_i	Komponente des Vergleichsraum und Vergleichswert	50
$\mathcal{C}, \mathcal{C}_i$	Constraints	38, 108
O, O_i	Universum von Realwelt-Objekten	
o, o_j	Realwelt-Objekte	
$\mathcal{I}$	Meta- und Kontext-Information für die Ableitbarkeit	49
$\mathcal{K}$	Klassifikation einer Menge	50
$\mathcal{K}_i$	Die Klassen einer Klassifikation, $\mathcal{K}_i \in \mathcal{K}$	50
$\mathcal{M}$	log-lineares Modell	80
S, S^A	Schema einer Datenbank	22
C, C_i	Klassen eines Schemas	21
$\mathcal{G}, \mathcal{N}, \mathcal{E}$	ein Graph und seine Knoten- und Kantenmenge	30

Operatoren und reservierte Bezeichner

Zeichen	**Bedeutung**	**Seite**
$\ell : \tau$	Referenz-Typ	21
$a :: C$	Instanziierung einer Klasse	22
$a =_\tau b$	Gleichheit der Wert-Typen von a, b	25
$X^C(a) \rightarrowtail b$	Referenz zwischen Datenbank-Objekten	36
$node \lhd node'$	Nachfolger-Relation auf einem Baum	33
$\perp$	NULL-Wert eines Attributes X_i	27
$\mathrm{dom}(X)$	Wertebereich (Domäne) einer Attributmenge	21
$X \vdash Y$	funktionale Abhängigkeit, Ableitung von Attributen	27,49
$h \succ h', f \succ f'$	Verfeinerung von Transformations- und Vergleichsfunktionen	53, 64
$\mathbf{P}\{\, X(a) = x \,\}$	Wahrscheinlichkeit des Ereignisses $X(a) = x$	75ff
$\hat{\mathbf{P}}\{\, X(a) = x \,\}$	Schätzung der Wahrscheinlichkeit $\mathbf{P}\{\, X(a) = x \,\}$	117
$a \hookrightarrow o$	Bezug eines Daten-Elementes auf ein Realwelt-Objekt	114
$a \equiv b$	Bezug zweier Daten-Elemente auf äquivalente Realwelt-Objekte	105
$Same$	binäre Relation, die Tupel $(a, b) \in A \times A$ enthält, für die $a \equiv b$ gilt	105
$STC(Z), TC(Z)$	(symmetrischer) transitiver Abschluß einer binären Relation Z	104
$dist(\cdot, \cdot)$	Distanz-Funktion (Metrik)	61
$\mathbb{1}_x$	Identifikatorfunktion, $\mathbb{1}_x(y) = \begin{cases} 1 & x = y, \\ 0 & x \neq y. \end{cases}$	

Die Verwendung der Schriftarten

Roman	Standardschrift für Text
Italic	Hervorhebungen im Text, Variablen in Formeln
`Typewriter`	Programmcode

Einleitung

Looking for duplicated records
within a large real-world database
is like fishing in troubled waters
User experience

Objektidentifizierung ist notwendig, wenn Daten über Realwelt-Objekte auf mehrere Datenbanken verteilt sind und je Realwelt-Objekt zusammengefaßt werden sollen. Sofern von den Datenbanken ein gemeinsamer und konsistenter Schlüssel (globaler Identifizierer) verfügbar ist, ist eine Datenbankübergreifende Identifizierung damit möglich. Andernfalls ist diese erschwert, jedoch ist mit den verfügbaren Daten über Realwelt-Objekte eine näherungsweise Identifizierung möglich. Insbesondere muß aber auf die vollständig korrekte Identifizierung verzichtet werden, da fälschlicherweise Daten zusammengefaßt werden können, obwohl sie sich auf verschiedene Realwelt-Objekte beziehen und umgekehrt kann es vorkommen, daß Daten von Realwelt-Objekten nicht als identisch erkannt werden. In der vorliegenden Arbeit werden Ansätze und Methoden für die Identifizierung von Realwelt-Objekten in Datenbanken, die keine globalen Identifizierer verwenden, untersucht und verglichen.

Die Datenintegration umfaßt neben der Objektidentifizierung auch die Datenbereinigung und —als erste Stufe— die Integration heterogener Datenbank-Schemata. In der vorliegenden Arbeit liegt der Fokus auf der Objektidentifizierung, sowohl die Datenbereinigung (*engl.: data cleansing*) als auch die Integration heterogener Datenbank-Schemata wird in dieser Arbeit unter dem Aspekt der Objektidentifizierung behandelt.[1] Für die Objektidentifizierung gehen wir davon aus, das eine Schemaintegration bezüglich der in den Datenbanken verfügbaren identifizierenden Informationen über Realwelt-Objekte erfolgt ist — wir setzen voraus, das diese Informationen in ein globales Schema eingebettet werden können.

Typische Anwendungen der Datenintegration sind integrierte Bibliothekskataloge, die Zusammenführung von Informationen aus Internet-Datenbanken, beispielsweise je Produkt für e-Shops (Preisvergleiche) oder für Kleinanzeigen (Erkennen von wiederholten Anzeigen). Eine weitere Anwendung ist das Verknüpfen von Informationen aus verschiedenen Datenbanken: Ein Beispiel ist die registerbasierte Volkszählung, bei der je Person Daten aus

[1] Zur Datenbereinigung sei beispielsweise auf die Arbeiten von H. Galhardas et al. [GFS+01a, GFS+01b, Gal01] verwiesen; Die Integration heterogener Datenbank-Schemata, in der Literatur auch als *Integration auf Schemaebene* oder *Schemaintegration* bezeichnet wird, wird in der Arbeit von Larson, Navathe und Elmasri [LNE89] besprochen.

Melde- und anderen Registern zusammengeführt werden sollen. Ein anderes Beispiel ist das Realwelt-Objekt-bezogene Integrieren von Informationen aus verschiedenen Quellen in einem Data-Warehouse. Das Entdecken von Duplikaten in *einer* Datenbank ist ein Spezialfall der Objektidentifizierung, beispielsweise angewandt für personenbezogene Kundendaten. Im Folgenden bezeichnen wir Daten aus unterschiedlichen Datenbanken, die sich auf ein Realwelt-Objekt beziehen, ebenfalls als Duplikate.

Sofern ein *eindeutiger, globaler und korrekter Schlüssel* in den Datenbanken verfügbar ist, lassen sich Realwelt-Objekte eindeutig identifizieren und damit können die Informationen je Realwelt-Objekt integriert werden (ein Beispiel für einen globalen Identifizierer ist die ISBN für Bücher). Für viele Anwendungen ist jedoch kein globaler Schlüssel verfügbar, so daß eine Identifizierung nur mittels der in den Datenbanken über die Realwelt-Objekte enthaltenen Informationen möglich ist. In dieser Arbeit wird davon ausgegangen, daß kein *globaler und korrekter Schlüssel* (Identifizierer) für die Realwelt-Objekte vorhanden ist. Aufgrund unvollständiger, fehlerhafter und veralteter Daten läßt sich im Allgemeinen auch keine *eindeutige Identifizierung* erreichen. Vielmehr lassen sich oft nur potentielle Duplikate feststellen, wobei diese mit einer Bewertung versehen werden können (z.B. Bildung einer Rangfolge). Die *Objektidentifizierung* soll letztlich mit Algorithmen (Objektidentifizierungs-Verfahren) umgesetzt werden, die weitgehend automatisch potentielle Duplikate finden, bewerten und klassifizieren. Für die Klassifikation kommen zumindest die zwei Klassen *Duplikate* oder *Nicht-Duplikate* in Frage, es können aber —je nach Anwendungsfall— auch weitere Klassen benützt werden, beispielsweise *unentscheidbar* oder *potentielle Duplikate*. Alternativ kann auch ein Zugehörigkeitswert oder eine Wahrscheinlichkeit für die beiden Klassen *Duplikate* und *Nicht-Duplikate* angegeben werden.

Mit der vorliegenden Arbeit wird ein Rahmen für den Vergleich verschiedener Verfahren der Objektidentifizierung in Datenbanken geschaffen. Insbesondere wird die Objektidentifizierung als ein spezielles *Klassifikationsproblem* modelliert, so daß Verfahren des Maschinellen Lernens für die Objektidentifizierung eingesetzt werden können. Durch die Verwendung von Methoden wie beispielsweise dem Entscheidungsbaum-Verfahren lassen sich eine Vielzahl von Objektidentifizierungs-Verfahren realisieren, die in dieser Arbeit mit den derzeit in der Praxis angewandten Objektidentifizierungs-Verfahren verglichen werden. Bei diesen Methoden handelt es sich im Wesentlichen um drei Ansätze: *Record Linkage*, die *Sorted Neighborhood-method* sowie die *gewichtete Aggregation von Distanzen*. Die Methode des *Record Linkage* basiert auf einem Likelihood-Quotienten-Test, die Schwierigkeit besteht hier hauptsächlich in der Schätzung von Wahrscheinlichkeitsverteilungen. Bei der *Sorted Neighborhood-method* werden ein feste Anzahl von aufeinanderfolgenden Datensätzen miteinander verglichen. Die Datensätze sind bezüglich eines (zu definierenden) Sortierschlüssels angeordnet und der Vergleich basiert auf einer *Equality-Theory*, die letztlich eine Klassifikation der Paare in Duplikate und Nicht-Duplikate darstellt. Bei der *gewichteten Aggregation von Distanzen* handelt es sich um eine Anwendung der Ähnlichkeitssuche in Datenbanken, wobei die verwandte lineare Aggregation von Distanzmaßen einzelner Merkmale nicht unproblematisch ist, da die

zugrundeliegende Unabhängigkeitsannahme in der Regel nicht aufrechtzuerhalten ist.

Für die Evaluation von ausgewählten Objektidentifizierungs-Verfahren verwenden wir drei Realwelt-Datenbanken, die die Vielzahl der Anwendungsgebiete verdeutlichen. Auf die Besonderheiten jeder dieser Datenbanken wird detailliert eingegangen, insbesondere werden die Schwierigkeiten bei der Objektidentifizierung für jede Test-Datenbank beschrieben. Wir untersuchen *Charakteristika* der Datenbanken, wie z.B. deren Größe, die Selektivität und Korrektheit einzelner Attribute oder die erwartete Anzahl von Duplikaten. Verschiedene Qualitätskriterien werden untersucht, unter anderem die *Korrektheit* eines Verfahrens. Für den Vergleich der Verfahren definieren wir eine *Benchmark* mit einer Spezifikation des Testablaufes, so daß die vorgelegten Resultate reproduzierbar sind.

Die Identifizierung in Objekt-Orientierten Datenbanken bildet zwar die Grundlage des hier verfolgten Herangehens an die Objektidentifizierung — und wird deshalb im ersten Kapitel besprochen, aber es wird in dieser Arbeit keine allgemeine Theorie der Identifizierung für Objekt-Orientierte Datenbanken dargelegt oder entwickelt.

Die hier verfolgte Identifikation von Realwelt-Objekten in Datenbanken unterscheidet sich von der Identifikation von Personen und anderen Realwelt-Objekten aus Medien wie Bildern oder Videosequenzen, da dort der Anordnung *identifizierender visueller Merkmale* eine große Bedeutung zukommt, und insbesondere die räumliche Anordnung einzelner Merkmale im Medium für die Identifikation wichtig sein kann, wie z.B. der Augenabstand bei Menschen. In der Regel werden für diese Art der Identifikation die Daten speziell gewonnen (z.B. Paßfotos und Fingerabdrücke) und sehr spezielle Verfahren für den Vergleich angewandt. Sofern sich für die verwandten *identifizierenden Merkmale* eine Erfassung in einer Datenbank nebst adäquater Vergleichsfunktionen realisieren läßt, läßt es sich als einen Spezialfall der in dieser Arbeit untersuchten Objektidentifizierung auffassen.

Ebenfalls verschieden zum hier verfolgten Ansatz verhält es sich mit der Fusion von Daten aus Quellen unterschiedlicher Art (*engl.: Data Fusion*), bei dem beispielsweise geographische Karten mit Satellitenbildern und Radaraufnahmen überlagert werden können. Für die Verschmelzung der Daten werden in der Regel nichtlineare Transformationen angewandt. Der dort verfolgte Ansatz ist grundsätzlich verschieden zum hier beschriebenen, da es nicht darum geht, Realwelt-Objekte zu identifizierenden, sondern die Zielstellung darin besteht, die gegebenen Daten gut aufeinander anzupassen (z.B. Überlagerung von geographischen Informationen, indem Bilder dergestalt transformiert werden, daß Lage und Ausdehnung von Flüssen, Siedlungen etc. nahezu deckungsgleich werden).

Ein weiteres Gebiet, das verschieden zum hier verfolgten Ansatz ist, ist die sogenannte Datenfusion (*engl.: statistical matching*), bei dem unterschiedlichen Erhebungen (z.B. Umfragen) anhand gemeinsamer Merkmale miteinander kombiniert werden können. Obwohl die dort verfolgte Idee grundsätzlich dem Identifizieren von Realwelt-Objekten entspricht, fehlt dort der Bezug zu identischen Realwelt-Objekten: Es werden zwei Erhebungen miteinander kombiniert, ohne daß es Realwelt-Objekte geben muß, auf

die sich beide Erhebungen beziehen. Eine Identifizierung findet dort folglich gar nicht statt, zwei Datensätze werden miteinander verbunden, wenn die (vorher auszuwählenden gemeinsamen) Merkmalswerte übereinstimmen.

Das Objektidentifizierungs-Problem

Informell fassen wir eine Datenbank als Menge von Elementen mit Daten über Realwelt-Objekte auf (eine genaue Definition wird auf Seite 23 gegeben). Als *Objektidentifizierungs-Problem* bezeichnen wir dann die folgende Aufgabenstellung:

> *Gegeben zwei Datenbanken, die Elemente mit Daten über ein Universum von Realwelt-Objekten enthalten, sind genau die Paare von Elementen* $(a, a') \in A_1 \times A_1$, $(b, b') \in A_2 \times A_2$ *sowie* $(a, b) \in A_1 \times A_2$ *zu finden, die sich jeweils auf ein und dasselbe Realwelt-Objekt beziehen.*

Die Paare $(a, a') \in A_1 \times A_1$ und $(b, b') \in A_2 \times A_2$ werden als *Duplikate* in A_1 bzw. A_2 bezeichnet, während die Menge der Paare $(a, b) \in A_1 \times A_2$ als *extensionale Überlappung* bezeichnet wird. $a \in A_1$ bezeichnen wir ebenfalls als Duplikat von $b \in A_2$ (und umgekehrt), wenn a und b sich auf ein und dasselbe Realwelt-Objekt beziehen.

Aufgrund der unterschiedlichen Repräsentation von Realwelt-Objekten in verschiedenen Datenbanken, fehlender globaler und konsistenter Identifizierer, fehlerhafter und veralteter Daten (über die nicht einmal bekannt sein muß, wie fehlerhaft oder veraltet sie sind), muß im Allgemeinen auf die vollständig korrekte Identifikation verzichtet werden. Die Forderung nach einer Berechenbarkeit auf beschränkten Ressourcen trägt ebenfalls dazu bei, daß in der Regel auf das vollständige Verarbeiten aller verfügbaren identifizierenden Informationen je Datenbank-Element verzichtet werden muß. Es muß zudem mit der Unsicherheit umgegangen werden, daß zwei Elemente sich auf dasselbe Realwelt-Objekt beziehen können, obwohl sie unterschiedliche Werte (oder Referenzen) haben. Andererseits können sich auch Elemente auf unterschiedliche Realwelt-Objekt beziehen, obwohl alle ihre Werte (und Referenzen) übereinstimmen.

Es wird nach Objektidentifizierungs-Verfahren gesucht, die eine näherungsweise Lösung für das oben dargestellte Objektidentifizierungs-Problem angeben, die also für zwei Datenbanken D_1', D_2' (derselben Art wie D_1, D_2, aber möglicherweise mit anderen Elementen) jeweils eine Menge von Paaren liefern, die als sich auf dieselben Realwelt-Objekte beziehend aufgefaßt werden sollen. Jedem Paar kann zusätzlich eine Bewertung hinzugefügt werden. Gegebene Qualitätskriterien, wie beispielsweise *Korrektheit* oder *Performanz*, lassen sich dann für verschiedene Objektidentifizierungs-Verfahren miteinander vergleichen.

Publikationen zum Thema der Arbeit

Die Veröffentlichungen zum Thema lassen sich weitestgehend auf zwei Forschungs-Schwerpunkte aufteilen:

- *Record Linkage* und
- *Duplikat-Erkennung in Datenbanken*,

so daß wir diese in zwei separaten Abschnitten besprechen.

Record Linkage

In den 50ern und 60ern des vorigen Jahrhunderts entwickelten H. Newcombe u. a. [NKAJ59] sowie I. Fellegi und A. Sunter [FS69] einen Ansatz zur Zusammenführung großer Datenbestände, der unter dem Namen *Record Linkage* seitdem vielfach angewandt und weiter erforscht wurde. Hauptanwendungsgebiet des *Record Linkage* ist die Verknüpfung von Personendaten in statistischen, medizinischen und Adreß-Datenbanken. Umfangreiches Material zum *Record Linkage* findet man in den Proceedings der beiden Konferenzen in Arlington, Virginia ([KA85], [AJ97]).

Record Linkage basiert auf einem statistischen Test, dem *Likelihood-Quotienten-Test*, bei dem der Quotient aus zwei Verteilungen für Vektoren von Vergleichswerten von Datensatz-Paaren als Teststatistik gebildet wird. Als Verteilungsannahme wird die Multinomialverteilung für Vektoren nominaler Vergleichswerte von Datensätzen zugrundegelegt. Die Multinomialverteilung wird jeweils für zwei Stichproben von Datensatz-Paaren geschätzt, für eine Stichprobe mit Datensatz-Paaren von Duplikaten (dort sogenannte *matches*), sowie für eine Datensatz-Paar-Stichprobe von Nicht-Duplikaten (sogenannte *nonmatches*). In Abhängigkeit vom Wert des dadurch geschätzten *Likelihood-Quotienten* werden die Hypothesen *Duplikat* und *Nicht-Duplikat* für ein Datensatz-Paar angenommen oder verworfen.[2] Um große Datenmengen effizient verarbeiten zu können, werden diejenigen Datensätze zu Blöcken zusammengefaßt und paarweise miteinander verglichen, die bezüglich bestimmter festzulegender Bedingungen übereinstimmen oder nicht zu stark abweichen, etwa die Postleitzahl oder das Geburtsdatum. Dieses Vorgehen ist in der Literatur zum Record Linkage als *Blocking* bekannt und wurde erstmals von Newcombe [New67] beschrieben.[3]

In den Arbeiten von Winkler [Win93, Win95], Meng und Rubin [MR93] sowie Liu und Rubin [LR94] wird beschrieben, wie eine verbesserte Schätzung der Multinomialverteilung durch Verwendung einer Modifikation des *EM-Algorithmus*[4] erreicht werden kann. Bei der verbesserten Schätzung werden Interaktionen von bis zu maximal drei Faktoren berücksichtigt (wir verwenden in dieser Arbeit ebenfalls Modelle mit Interaktionen zwischen zwei und drei Faktoren). Weitere Arbeiten setzen sich damit auseinander, ein nicht-überwachtes Vorgehen bei der Schätzung der Multinomialverteilung zu verwenden (siehe [Win89, LR01, Win02, Yan02]). Motiviert wird dieses Vorgehen durch die Schwierigkeit, Lernstichproben zu gewinnen, in denen

[2] Record Linkage ist ausführlich im Abschnitt 3.3.2 auf Seite 76 dieser Arbeit beschrieben.

[3] Im Kapitel 5 dieser Arbeit wird eine Vorauswahl von Paaren bestimmt, die u.a. solche Bedingungen als relationale Selektoren enthalten kann

[4] Der *Expectation Maximization-Algorithmus* wurde 1977 von Dempster, Laird und Rubin [DLR77] als Verfahren zur Schätzung unbekannter, fehlender Werte entwickelt.

die Duplikate bekannt sind, die sich für überwachtes Lernen nutzen lassen, während sich beliebig große Lernstichproben von Paaren generieren lassen, wenn nicht angegeben werden muß, welches die Duplikate sind. Basierend auf Bayes'schen Netzwerken werden unter Verwendung des EM-Algorithmus log-lineare Modelle mit latenten Klassen geschätzt. Die verwandten Lernstichproben enthalten hierbei einen Teil von Paaren bei denen bekannt ist, ob es Duplikate oder keine Duplikate sind und einen großen Anteil ohne diese Angabe. Dieses Vorgehen zeigt zufriedenstellende Ergebnisse für Personendaten, wenn eine hohe Daten-Qualität vorliegt, sehr wenige Vergleichswerte je Attribut verwandt werden und eine große Zahl von Duplikaten (z.B. mehr als 5%) in der Datenbasis enthalten ist. Die Anforderung an eine hohe Daten-Qualität läßt sich abschwächen, indem man eine gute Separierbarkeit von Duplikaten und Nicht-Duplikaten durch ihre Attributwert-Vergleiche fordert. Für Realwelt-Daten sind diese Voraussetzungen im allgemeinen nicht erfüllt, wir gehen aufgrunddessen darauf bei der Diskussion des Record Linkage im Kapitel 3 dieser Arbeit nicht näher ein.

Duplikat-Erkennung in Datenbanken

Die Duplikat-Erkennung und damit verbundene Datenbereinigung in Datenbanken wird in mehreren Arbeiten behandelt. Während beim Record Linkage ein statistisches (Lern-)Verfahren im Vordergrund steht, wurde in der Datenbank-Forschung zu diesem Thema anfänglich ein manuell erstelltes Regelwerk zur Duplikat-Erkennung verwandt. Schwerpunkt der Arbeiten bildet die Effizienz eines Systems für (sehr) große Datenbestände. Eine Berücksichtigung des bewährten Lösungsansatzes des Record Linkage oder geeigneter Lernverfahren zur Gewinnung eines Regelwerks findet erst in neueren Arbeiten statt (z.B. Elfeky et al. [EVE02], Bilenko und Mooney [BM03]).

Ausgangspunkt der meisten Arbeiten zur Duplikat-Erkennung in Datenbanken ist die Arbeit von Bitton und DeWitt [BD83], in der ein effizientes Verfahren zu Elimination von duplizierten Tupeln in einer Tabelle, d.h. von Tupeln, deren Attributwerte vollständig übereinstimmen, vorgestellt wird.

Wang und Madnick diskutieren in [WM89] das Datenbank-übergreifende Identifikations-Problem. Zur Lösung des Problems wird dort eine Menge von Regeln, aus denen die Gleichheit von Datenbank-Elementen abgeleitet werden kann, verwandt, die allerdings manuell erstellt werden muß.

In den Arbeiten von Lim, Srivastava et al. [LSPR93, LSS96] wird das Identifikationsproblem von Datenbank-Entitäten dargestellt und einige Eigenschaften des Problems diskutiert. Der dort verfolgte Ansatz, Äquivalenzen (in Form verallgemeinerter funktionaler Abhängigkeiten) zwischen (gegebenen oder ableitbaren) gemeinsamen Schlüssel-Attributen zur Identifizierung zu verwenden, ist aufgrund hoher Fehlerraten in den Daten für viele praktische Anwendungen unbrauchbar oder zu stark fehlerbehaftet. Die Verwendung von Heuristiken und probabilistischen Methoden (die Wahrscheinlichkeit von Fehlern in Attributen betreffend) wird dort kurz erwähnt, jedoch nicht weiter ausgeführt.

Der Ansatz von Wang und Madnick [WM89] wurde von Hernandez und Stolfo [HS95, Her96] erweitert, indem eine Gleichheits-Theorie (*engl.:*

Equational Theory) zugrundegelegt wird, bei der Abweichungen von Werten berücksichtigt werden können (die Gleichheits-Theorie muß jedoch ebenfalls durch einen Domain-Experten manuell erstellt werden). Für das effiziente Verarbeiten großer Datenbestände wurde von Hernandez und Stolfo ein Verfahren entwickelt, das unter dem Namen *Sorted Neighborhood Method* vielfach zur Duplikat-Erkennung verwandt wird. Den Kern dieses Verfahrens bildet das Sortieren der Datensätze bzgl. eines geeigneten abgeleiteten Sortierschlüssels und das Laden einer festen Zahl von Datensätzen in den Arbeitsspeicher. Es nutzt die Datenstruktur einer Schlange (*engl.: Queue*) zur Verarbeitung, d.h. es wird immer ein neuer Datensatz geladen und seine Werte überschreiben den Speicherplatz des bisher am Längsten im Speicher gehaltenen Datensatzes. Jeder neue Datensatz wird dann mit allen anderen im Arbeitsspeicher befindlichen verglichen und bezüglich eines Regelsystems (der *Equational Theory*) entschieden, ob es sich um Duplikate handelt.[5] Schwerpunkt der Forschung zur Duplikat-Erkennung bildete die Skalierbarkeit, d.h. der Berechnungsaufwand in Abhängigkeit von der Anzahl n der Datensätze. Durch die Beschränkung des Vergleiches je Datensatz auf eine feste Zahl k von Datensätzen skaliert diese Methode mit $O(k \cdot n)$ linear (ohne Berücksichtigung des vorherigen Sortierens).

Ausgehend von dieser Methode gab es mehrere Forschungsarbeiten. Das mehrfache Sortieren und Anwenden der *Sorted Neighborhood Method* wurde ebenfalls von Hernandez und Stolfo [Her96, HS98] sowie von Feekin und Chen [FC00] untersucht, da sich durch mehrere Durchläufe die Korrektheit erhöhen läßt. Alternativ zur Sortierung der Datensätze wurden in den Arbeiten von H. Galhardas et al. [GFS$^+$01b] Cluster-Verfahren verwandt, um ähnliche Datensätze in Gruppen zusammenzufassen, die dann jeweils paarweise miteinander verglichen werden.

Mehrere Autoren machen Vorschläge zu einem allgemeinen Modell (*engl.: Framework*) für die Objektidentifizierung. Zu erwähnen sind (neben den eigenen Arbeiten) folgende Ansätze:

- *Data Cleaning Framework* von Galhardas et al. [GFSS00, GFS$^+$01b, Gal01], dort wird die Datenbereinigung mittels verschiedener Operatoren in einer SQL-ähnlichen deklarativen Sprache umgesetzt. Lernverfahren werden dort nicht verwandt, für die Datensatz-Zuordnung wird davon ausgegangen, daß ein Matching-Operator spezifizierbar ist.

- Der Ansatz von Verykios, Elmagarmid, Elfeky et al. [VEE$^+$00, VEH00, CDEV] verwendet zwei Lernverfahren (Entscheidungsbaum-Verfahren und Instance Based Learning [AKA91]). Der Ansatz des Record Linkage wird allerdings dort nicht in den Vergleich mit einbezogen. Hervorzuheben bei diesem Ansatz ist, daß der gesamte Prozeß in die zwei Phasen *Searching* und *Matching* aufgeteilt wird, analog zu der in dieser Arbeit verwandten Unterteilung in die Phase der Vorauswahl von Paaren (Kapitel 5) und eine Klassifikations-Phase (Kapitel 2 und 3).

[5] Die *Sorted Neighborhood Method* wird detailliert im Abschnitt 3.3.4.1 auf Seite 93 als Spezialfall der *k-Nächsten Nachbarn Klassifikation* (Abschnitt 3.3.4 auf Seite 91) besprochen.

- Das Modell von Elfeky, Verykios und Elmagarmid [EVE02] führt die Ideen aus den im vorigen Punkt besprochenen Arbeiten fort. Im Unterschied zu den Vorläuferarbeiten wird der Ansatz des Record Linkage in der Arbeit [EVE02] aufgegriffen. In der dort vorgestellten Software (*Record Linkage Toolbox*) werden neben Record Linkage ebenso überwachte und nicht-überwachte Lernverfahren[6] berücksichtigt. Für Wert-Vergleiche kann aus einer Vielzahl von Vergleichsfunktionen gewählt werden. Dem effizienten Verarbeiten großer Datenmengen wird Bedeutung geschenkt, es werden *Blocking* und die *Sorted Neighborhood Method* eingesetzt. Es sei bemerkt, daß es von den diskutierten Modellen am ehesten dem in dieser Arbeit vorgestellten Ansatz entspricht.

- In der Arbeit von Bilenko und Mooney [BM03] wird die Duplikat-Erkennung reduziert auf ein Lernproblem für Zeichenketten-Vergleichsfunktionen je Attribut, deren (mehrdimensionale) Vergleichswerte darauf in einer binäre Entscheidungsregel für die Klassen *Duplikat/Nicht-Duplikat* kombiniert werden. Für beide Teilaufgaben (dem adaptiven Lernen von Zeichenketten-Vergleichsfunktionen und dem Gewinnen der Entscheidungsregel) wird das Klassifikations-Verfahren *Support Vector Machines*[7] eingesetzt. Hervorzuheben ist, daß das Lernen Domänen-abhängiger Gewichte für die unterschiedlichen Edit-Operationen (Einfügen/Löschen/Substituieren einzelner Zeichen) die Trennung zwischen Duplikaten und Nicht-Duplikaten verbessern kann. 4 Ebenso positiv wirkt sich die vorgestellte Korrektur der Gewichte der als *TF-IDF-Similarity* bekannten Bestimmung von Ähnlichkeiten zweier Zeichenketten aus.[8] Zu hinterfragen bleibt, inwieweit die alleinige Berücksichtigung von Edit-Operationen die Ähnlichkeit zwischen Zeichenketten fassen kann, da beispielsweise Abkürzungen damit nicht erkannt werden können.

Wir diskutieren kurz weitere Forschungsarbeiten. Im Rahmen der Informations-Integration von Daten aus Internet-Datenbanken wurde die Notwendigkeit der Objektidentifizierung ebenfalls erkannt. In dem Informations-Integrations-System *WHIRL* von Cohen [Coh98] wird basierend auf der TF-IDF-Similarity (siehe Fußnote 8 auf Seite 8) eine näherungsweise Objektidentifizierung umgesetzt. In der Arbeit von Tejada, Knoblock und Minton [TKM01] werden *object identification rules* mit einem Entscheidungsbaum-Verfahren aus einer Beispielstichprobe gelernt. Diese Regeln werden

[6] Das dort vorgestellte nicht-überwachte Lernverfahren *Clustering Record Linkage Model* basiert auf dem k-means-Algorithmus. Es sei bemerkt, daß bei diesem Algorithmus die Euklidische Distanz mehrdimensionaler Punkte für die Bildung der Cluster verwandt wird, die im Falle nicht binärer Vergleichsfunktionen (wie z.B. der Minimum-Edit-Distanz) keine sinnvollen Werte liefern kann.

[7] Als *Support Vector Machines (SVM)* wird ein Klassifikations-Verfahren bezeichnet, das ein (optimales) Trennungsfunktional für positive und negative Beispiele in einem mehrdimensionalen Vektorraum bestimmt, das aus einer bestimmten Klasse stammt (lineare, quadratische oder andere Funktionale, die Klasse wird durch die Wahl einer sogenannten Kern-Funktion bestimmt), siehe Vapnik [Vap98].

[8] Bei der Berechnung der *TermFrequency-InverseDocumentFrequency* zweier Zeichenketten werden Häufigkeiten von Termen berücksichtigt, Übereinstimmung seltener Terme führt zu hohen Werten der TF-IDF-Similarity; siehe [Sal89].

zur Verknüpfung der Daten in einem Informations-Integrations-System angewandt.

In einigen Publikationen wird der Vergleich von Zeichenketten intensiv untersucht, da typographische Fehler eine der Hauptschwierigkeiten bei der Duplikat-Erkennung sind. Ausgehend von Zeichenketten-Vergleichsfunktionen wie dem Vergleich von N-Grams, der Minimum-Edit-Distanz [Gus97] oder dem Smith-Waterman-Algorithmus [SW81] haben mehrere Autoren neue und modifizierte Vergleichsfunktionen für Zeichenketten entwickelt.

Beispielsweise wird die Duplikat-Erkennung in den Arbeiten von Monge und Elkan [ME96] sowie Monge [Mon97] durch spezielle Zeichenketten-Vergleiche umgesetzt. Nur die Datensätze, deren Werte nicht zu stark voneinander abweichen (z.B. eine geringe Minimum-Edit-Distanz haben, oder falls ein Wert eine Abkürzung des anderen Attributwertes ist) werden miteinander verglichen. Die Datensätze mit Vergleichswerten, die eine vorher zu definierende Schranke übertreffen, werden dann als Duplikate ausgegeben.

Zu nennen sind weiter die Arbeiten von Jaro [Jar89] und Winkler [Win90], in denen neue Vergleichsfunktionen vorgestellt und diskutiert werden.

In einer Reihe neuerer Arbeiten findet man Ansätze, die adaptiv mittels überwachter Lern-Verfahren Domänen-spezifische Zeichenketten-Vergleichsfunktionen auf Beispiel-Stichproben lernen, siehe z.B. die Arbeiten von Cohen et al. [CRF03a, CRF03b], Yancey [Yan03] sowie die weiter oben besprochene Arbeit von Bilenko und Mooney [BM03]. Obwohl das Finden von Vergleichsfunktionen, die eine gute Trennung der Duplikate von Nicht-Duplikaten erreichen, wichtig ist für die Duplikat-Erkennung, ist dieses nur ein Baustein im gesamten Prozeß.

Wir geben noch einige Arbeiten an, die sich auf Teilaspekte des in dieser Arbeit beschriebenen Ansatzes beziehen.

Einen Überblick über Methoden zur effizienten Auswahl von Paaren und den Vergleich von vier Methoden (Blocking, SNM, Bigram-Indexing und Canopy-Clustering) findet man beispielsweise bei Baxter, Christen und Churches [RBC03].

Die Arbeiten von Christen, Churches et al. [CCZ02, CCLZ02] setzen sich mit der Standardisierung von Namens- und Adreß-Feldern auseinander, gewinnen also für die Identifizierung geeignete abgeleitete Attribute.

Erwähnenswert ist außerdem die Literaturstudie von Gu, Baxter, Vickers und Rainsford [GBVR03], die einen Überblick über den gegenwärtigen Stand der Forschung zum Record Linkage gibt und außerdem eine Liste der verfügbaren Software enthält. Weitere Artikel, in denen Informationen über Software gefunden werden kann, sind die Arbeiten von Winkler [Win01b] sowie Bell und Sethi [BS01, S. 86].

Forschungsbeitrag der Dissertation

Mit der vorliegenden Arbeit wird erstmals ein mathematisches Modell („Framework“) für die Identifizierung von Realwelt-Objekten in multiplen Datenbanken angegeben. Dieses Modell bildet nicht nur eine methodische Grundlage für die Objektidentifizierung, sondern liefert auch einen Rahmen für die

praktische Anwendbarkeit, insbesondere auch für große Datenbanken.[9]

Dieses Modell bietet eine vereinheitlichte Darstellung für die verschiedenen Ansätze zur Lösung des Objektidentifizierungs-Problems, die im vorangegangenen Abschnitt dargestellt wurden. Durch diese Vereinheitlichung wird eine Evaluation der Ansätze möglich. Wir gehen auf die Besonderheiten der Arbeit näher ein:

- Die Identifizierung von Realwelt-Objekten in mehreren Datenbanken wird als *abstraktes Klassifikations-Problem* modelliert, so daß Verfahren aus der Statistik und dem Data Mining zum Lernen von Klassifikationen eingesetzt werden können. Um diese Verfahren anwenden zu können, wird eine Transformation der für die Identifizierung verwendbaren Daten in ein gemeinsames Schema und ein Vergleich für diese abgeleiteten Werte durchgeführt. Mit den Vergleichswerten einer Beispiel-Stichprobe kann dann eine Klassifikation gelernt werden (siehe dazu die Kapitel 2 und 3).

- Die Fundierung der Objektidentifizierung mit Objekt-Orientierten Identifikationsansätzen im 1. Kapitel ist ebenfalls neu. Zur Identifizierung ziehen wir rationale Bäume heran, die neben den Attribut-Werten eines Datenbank-Objektes auch Referenzen auf andere Datenbank-Objekte aufweisen können (und damit die Werte dieser Datenbank-Objekte und Werte der Datenbank-Objekte, die von ihnen referenziert werden etc.). In der im vorangegangenen Abschnitt besprochenen Literatur wird über den Vergleich von Attributen einer Relation nicht hinausgegangen (d.h. es wird die Wert-basierte Identifikation angewandt).

- Die Einbettung der Daten aus unterschiedlichen Quellen in ein Schema sowie die Ableitung identifizierender Attribute wurde in der Literatur zur Objektidentifizierung bisher nicht untersucht, sondern es werden semantisch äquivalente Attribute vorausgesetzt.[10] Ebenfalls unberücksichtigt blieb bisher die Behandlung fehlender Werte im Prozeß der Objektidentifizierung, deren Berücksichtigung die Korrektheit der Objektidentifizierung erhöhen kann. Diese Aspekte werden im 2. Kapitel behandelt.

- Ein weiterer Aspekt, der in der Forschung bisher unzureichend untersucht wurde, ist die Bestimmung von Qualitätsmerkmalen für Objektidentifizierungs-Probleme. In der vorliegenden Arbeit werden einige für die Identifizierung relevanten Merkmale in der Form *semantischer Constraints* vorgestellt (Kapitel 4).

- Die Interpretation der Auswahl von (alternativen) Methoden zur effizienten Auswahl von Paaren (wie z.B. Blocking-Variablen) als Optimierungs-Problem, dem ein Kosten-Modell zugrundeliegt, ist bislang noch

[9] Z.B. Sicherstellung der Skalierbarkeit bzgl. Datenbankgröße unter Kontrolle der Fehlerraten durch Steuerung der Vorauswahl, vgl. Kapitel 5

[10] Im Falle der Duplikat-Erkennung in *einer* Tabelle handelt es sich um dieselben Attribute.

nicht in der Literatur besprochen worden. Die zugrundeliegende Idee hierbei ist, daß immer eine große Bandbreite für die Auswahl von Selektoren besteht (z.B. Auswahl der Blocking-Variablen, akzeptiertes Toleranz-Intervall für den Band-Join), aus denen ein Vielzahl zusammengesetzter Selektoren konstruiert werden kann. Im 5. Kapitel wird ein Kosten-Modell vorgestellt, das einen generischen Zugang für die Auswahl eröffnet: Man muß sich nicht von vornherein auf die Auswahl-Methode festlegen, sondern kann in einer Kosten-Funktion die Kosten für die Paarbildung (d.h. Anzahl der zu verarbeitenden Paare) und zusätzliche Kosten (z.B. Sortieren, Index-Erstellung) berücksichtigen. Das eigentliche Optimierungs-Problem besteht dann darin, eine Kombination der Selektoren zu finden, die die Kostenfunktion minimiert, wobei eine Nebenbedingung erfüllt werden soll, die die Fehlerrate für das Nicht-Selektieren dubletter Paare beschränkt. Die Lösung des Optimierungs-Problems läßt sich direkt bestimmen. Für die relationalen Selektoren (d.h. Blocking-Variablen, mit und ohne Toleranz-Intervall) läßt sich zudem die Selektion effizient durch die Verwendung von den in Datenbank-Management-Systemen verfügbaren Indexstrukturen erreichen.[11]

- Durch die im formalen Modell erfolgte Vereinheitlichung wird eine Evaluation der Ansätze bezüglich Qualitätskriterien wie Korrektheit und Performanz möglich. Im Kapitel 6 wird ein Ansatz des Benchmarkings dreier Methoden vorgestellt und für drei Fallstudien (Datenbanken mit Wohnungsanzeigen, Kunden-Adressen sowie Bibliotheksdaten) exemplarisch durchgeführt. Auf den gesamten Testablauf, insbesondere auf die Besonderheiten der Stichprobenziehung wird dabei detailliert eingegangen.

Struktur und Inhalt der Arbeit

Die Arbeit gliedert sich in zwei Teile, einen theoretischen und einen praktischen Teil. Sie hat folgende Struktur:

Teil I: Theoretische Grundlagen der Objektidentifizierung in Datenbanken

- **Kapitel 1:** Datenbanken
- **Kapitel 2:** Das formale Modell
- **Kapitel 3:** Klassifikation

Teil II: Evaluation von Objektidentifizierungs-Verfahren

- **Kapitel 4:** Datenqualität
- **Kapitel 5:** Vorauswahl von Datensatz-Paaren
- **Kapitel 6:** Benchmarking

[11] Es sei bemerkt, daß die Verwendung von effizienten Indexstrukturen für den Prozeß der Objektidentifizierung bislang noch nicht untersucht wurde, im 5. Kapitel wird darauf eingegangen.

Teil I: Theoretische Grundlagen der Objektidentifizierung

Innerhalb des ersten Teils wird zunächst auf Identifikationsansätze in Datenbanken eingegangen und es werden einige der verwandten Methoden näher untersucht. Im darauffolgenden Kapitel wird das formale Modell eingeführt, das die Grundlage für die Evaluation der Verfahren bildet. Der Kern dieses formalen Modells ist die Aufteilung der Objektidentifizierung in drei aufeinanderfolgende Schritte:

1. **Konversion:** Transformation der Daten in ein einheitliches Schema, das aus Elementen besteht, die die Informationen zu einzelnen Realwelt-Objekten abgegrenzt von anderen Elementen enthält (z.B. Tupel einer Tabelle, Teilbäume eines Graphen).

2. **Vergleich:** Vergleich von Paaren von Elementen mittels geeigneter, frei wählbarer Vergleichsfunktionen.

3. **Klassifikation:** Klassifikation von Paaren von Elementen anhand ihrer Vergleichswerte, beispielsweise als *Duplikat* oder *Nicht-Duplikat*.

Für die Umsetzung dieses Modells wird als Software-Architektur die von Wiederhold [Wie92] eingeführte *Mediator-Wrapper-Architektur* verwandt, wobei die Objektidentifizierung in einem speziellen Mediator, dem *Identificator* realisiert ist. Im dritten Kapitel werden die Besonderheiten einzelner Klassifikationsverfahren dargestellt und ihre Anwendung zum Klassifizieren von Paaren von Elementen beschrieben.

Teil II: Evaluation von Objektidentifizierungs-Verfahren

Der zweite Teil befaßt sich mit dem Vergleich von Objektidentifizierungs-Verfahren. Das vierte Kapitel setzt sich mit Fragen der Qualität von Daten auseinander. Das fünfte Kapitel geht auf Möglichkeiten der Optimierung durch geeignete Vorauswahl von Paaren unter Verwendung von Indexstrukturen ein. Im sechsten Kapitel wird der Vergleich von Objektidentifizierungs-Verfahren durchgeführt — den wir als *Benchmarking* bezeichnen. Zunächst geben wir Definitionen für Qualitätskriterien für Objektidentifizierungs-Verfahren, wie z.B. die Korrektheit. Im Anschluß daran wird die Spezifikation der Tests beschrieben. Der Vergleich wird für drei Realwelt-Datenbanken umgesetzt, die drei unterschiedlichen Anwendungsgebieten entsprechen:

- Berliner Wohnungsanzeigen aus den Online-Editionen zweier Tageszeitungen,
- Bibliographische Daten aus dem *Kooperativen Bibliotheksverbund Berlin-Brandenburg*, sowie
- Kundendaten eines deutschen Unternehmens.

Die für die verwandten Daten erforderliche Wahl der Vergleichsfunktionen sowie die notwendigen Anpassungen und Parametrisierungen der einzelnen Methoden werden beschrieben. Der Vergleich wird für folgende drei Verfahren durchgeführt:

- Klassifikation mit Assoziationsregeln (AssoClass),
- Record Linkage, sowie
- Entscheidungsbaum-Verfahren.

Die Ergebnisse der einzelnen Testläufe und eine Auswertung mit kritischer Diskussion der Resultate schließen das Kapitel ab.

Teil I

Theoretische Grundlagen der Objektidentifizierung in Datenbanken

Kapitel 1

Datenbanken

Knuper, knuper, kneischen,
wer knüpft an meinem Häuschen?

Der Wind, der Wind,
das himmlische Kind.

Aus „Hänsel und Gretel“[1]

Inhalt dieses Kapitels. Wir setzen uns mit der Identität und Repräsentation von Realwelt-Objekten in Datenbanken auseinander. Ausgehend von einem Typsystem führen wir die Begrifflichkeiten Klasse, Schema, Datenbank und Datenbank-Objekt ein. Es werden Identifizierungs-Ansätze für relationale und Objekt-Orientierte Datenbanken untersucht. Besondere Aufmerksamkeit wird Referenz-Bäumen von Datenbank-Objekten in Objekt-Orientierten Datenbanken gewidmet, da diese für die Identifizierung herangezogen werden können. Es werden identifizierende Attributmengen und identifizierende Referenz-Strukturen eingeführt. Die Übertragung der Identifizierungs-Ansätze auf multiple Datenbanken wird diskutiert.

1.1 Objekte und Objektidentität

1.1.1 Realwelt-Objekte und ihre Identität

Als *Realwelt-Objekte* bezeichnen wir Elemente aus einem Universum von Objekten in der Wirklichkeit, die sich klar voneinander abgrenzen lassen. Die *Objektidentität von Realwelt-Objekten* ist durch die in der Realwelt verwandten Identifikationsmechanismen determiniert, beispielsweise mit dem *Leibnizschen Ununterscheidbarkeitsprinzip* [Lei60]: Dinge, die nicht durch einstellige Prädikate P (einer gegebenen Sprache über diese Dinge) unterschieden werden können, also bezüglich der gegebenen Sprache ununterscheidbar sind, sind identisch; d.h. es gilt für Realwelt-Objekte o, v

$$o \equiv v \iff \forall P : \big(P(o) \wedge P(v)\big) \vee \big(\neg P(o) \wedge \neg P(v)\big). \tag{1.1}$$

[1] Die Zitate aus sechs bekannten Märchen der Gebrüder Grimm, die die Kapitel einleiten, sind der illustrierten Ausgabe *Die Kinder- und Hausmärchen der Brüder Grimm* des Kinderbuchverlags Berlin von 1970 (9. Auflage) entnommen.

Als *Datenbank-Objekte* bezeichnen wir hingegen die *elektronische Repräsentation* einzelner Realwelt-Objekte. Die *Objektidentität von Datenbank-Objekten* ist eng verbunden mit der verwandten Repräsentation der Realwelt-Objekte in einer Datenbank. In der Repräsentation kann einerseits ein *eindeutiger Identifizierer* vorhanden sein, der verwandt werden kann (z.B. Primärschlüssel für relationale Datenbanken), andererseits läßt sich eine Repräsentation der Realwelt-Objekte auch ohne *eindeutigen Identifizierer* umsetzen, so daß die Identifizierung anhand der in der Datenbank repräsentierten Eigenschaften und Beziehungen von Realwelt-Objekten erfolgen muß. Wir gehen in den folgenden Abschnitten auf beide Alternativen näher ein.

Nach Beeri und Thalheim [BT99] zeichnen sich Realwelt-Objekte (*engl.: real-world objects* or *entities*) durch folgende Wesensmerkmale aus:

- Ein Realwelt-Objekt ist eindeutig identifizierbar anhand seiner Vergangenheit, seiner Eigenschaften und seiner Beziehungen (zu anderen Realwelt-Objekten),
- Die Menge der Eigenschaften und Beziehungen ist beliebig und veränderlich,
- Ein Realwelt-Objekt hat einen Lebenszyklus, d.h. es beginnt zu einem Zeitpunkt zu existieren und die Existenz endet an einem späteren Zeitpunkt,
- Es existiert unabhängig von anderen Realwelt-Objekten.[2]

Wenngleich Realwelt-Objekte eine hohe Variabilität bezüglich der ihnen zuzuordnenden Eigenschaften und Beziehungen haben, läßt sich diese Flexibilität nur begrenzt bei der elektronischen Speicherung von Daten in Datenbanken berücksichtigen. Für die Speicherung der Daten findet eine Modellbildung statt, der eine endliche (und zunächst feste) Auswahl von Eigenschaften und Beziehungen der Realwelt-Objekte zugrundeliegt. Ausgehend von den Anforderungen einer Anwendung (einer Benutzer-Sicht) wird ein *Modell der Wirklichkeit* für ein Universum von Realwelt-Objekten entworfen, es wird eine logische Repräsentation erstellt. Dieses *Modell der Wirklichkeit* wird dann in ein Datenmodell überführt, das ein Modell für eine elektronische Repräsentation darstellt. Basierend auf dieser elektronischen Repräsentation können dann Daten über Realwelt-Objekte (des Universums) in einer Datenbank elektronisch gespeichert werden (vgl. Abbildung 1.1). Eine spezifische Software (beispielsweise ein Datenbank-Management-System) ermöglicht dann Interaktionen eines Benutzers oder von Anwendungen mit einer Datenbank. Diese Software stellt Schnittstellen für die Datenbank zur Verfügung, die Zugriffsmöglichkeiten mittels eines Befehlssatzes, einer Programmiersprache oder mittels einer Abfragesprache wie SQL[3] erlaubt.

[2] Der Beginn der Existenz eines Realwelt-Objektes hängt in der Regel jedoch von anderen Realwelt-Objekten ab.

[3] SQL ist die Abkürzung für *Structured Query Language*, eine in relationalen Datenbanken verwandte Abfragesprache, siehe auch Beispiel 1.11 auf Seite 24.

Da ein Modell nur für eine beschränkte Zahl von Anwendungen verwendbar ist, erhält man in der Praxis verschiedene Modelle der Wirklichkeit, die zu unterschiedlichen elektronischen Repräsentationen von *identischen* Realwelt-Objekten führen. Insbesondere für die Identifikation von Realwelt-Objekten ist diese Variabilität problematisch, da nicht alle für die Objektidentifizierung verwendbaren Informationen in einem Modell enthalten sein müssen (sofern sie für eine Anwendung nicht relevant sind). Weiterhin können die in zwei Modellen gemeinsam enthaltenen Informationen unterschiedlich repräsentiert sein, so daß eine (möglicherweise nur partielle) Überführung/Einbettung in ein *globales, einheitliches Modell* erforderlich wird. Ein weiteres Problem stellt die Granularität/Modellsicht der elektronischen Repräsentation dar: In zwei Modellen für *dasselbe* Universum von Realwelt-Objekten können unterschiedliche Auffassungen über die Definition eines Realwelt-Objektes vorliegen. Beispielsweise kann in einer Datenbank, die Informationen über Bücher enthält, das Realwelt-Objekt *Buch* als *Auflage eines Buches* modelliert sein, so daß sämtliche Bücher einer Auflage als *ein* Realwelt-Objekt aufgefaßt werden, während in einer Datenbank, die für die Verwaltung des Bestandes einer Bibliothek entworfen wird, sehr wohl zwischen den einzelnen Büchern einer Auflage *identischen Inhaltes* unterschieden werden muß. Wir haben es hier mit zwei unterschiedlichen Konzepten der Objektidentität zu tun, wobei eine Überführung der Identität eines *Buches b* in die Identität einer *Auflage eines Buches a* offensichtlich durch die Zusammenfassung aller Bücher einer Auflage zu einem *abstrakten* Realwelt-Objekt erfolgen kann, sofern die Information über die *Auflage* von *b* aus seiner Repräsentation ermittelt werden kann.

Um eine Identifizierung von Realwelt-Objekten in elektronischen Repräsentationen unterschiedlicher Modelle zu ermöglichen, sind für eine vereinheitliche Sicht eine Benutzers auf die Realwelt-Objekte nach Kent et al. [KAA+92] die unterschiedlichen Konzepte der Objektidentität wie folgt in Einklang zu bringen:[4]

(a) Die in jeder einzelnen elektronischen Repräsentation umgesetzte Objektidentität der Datenbank-Objekte ist geeignet zu kontrollieren,

(b) Für die vereinheitlichte Sicht auf die Realwelt-Objekte ist eine Objektidentität festzulegen, sowie

(c) Es sind Abbildungen zwischen den Objektidentitäten der Datenbank-Objekte aus (a) und der Objektidentität der Realwelt-Objekte in (b) zu realisieren, beispielsweise durch die Einbettung der verschiedenen elektronischen Repräsentationen in eine globale elektronische Repräsentation, die eine vereinheitlichte Sicht darstellt.

1.1.2 Identität von Objekten in Datenbanken

Die Repräsentation von Realwelt-Objekten in Informationssystemen wird in den Arbeiten von Abrial [Abr74], Kent [Ken78], Ullmann [Ull88], Matsus-

[4] „*It is necessary to (a) manage identity for the underlying objects reflecting the data in the external sources, (b) manage identity for the objects in the end user's unified view, and (c) maintain mappings between the two.*“ [KAA+92, S.2]

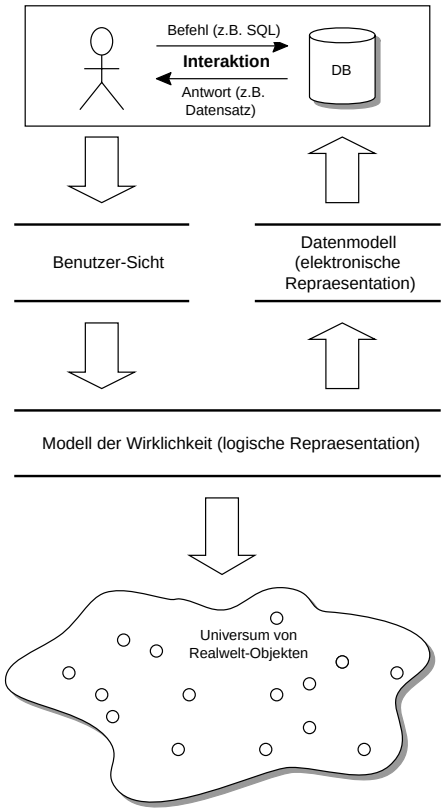


Abbildung 1.1: Die Modellierung von Datenbanken

hima und Wiederhold [MW90] diskutiert.

Der *Objekt-Begriff* wurde in der Informatik durch die *Objekt-Orientierung* geprägt. Als *Objekt* wird die Repräsentation von Realwelt-Objekten oder abstrakten Konzepten in einem elektronischen System bezeichnet. Jedes Objekt verfügt über eine *Objektidentität* und über Wesensmerkmale, die ähnlich den Wesensmerkmalen von Realwelt-Objekten sind. Nach Khoshafian und Copeland [KC86] sowie Beeri und Thalheim [BT99] haben *Objekte* die folgenden Charakteristika:

- Ein Objekt hat eine (interne) Struktur und einen Zustand (Status) entsprechend dieser Struktur,
- Ein Objekt hat einen Lebenszyklus: es wird erzeugt, kann verändert und gelöscht werden,
- Seine Identität ist unveränderlich während seiner Existenz. Die Identität ist das Merkmal, mit dem man jedes Objekt von allen anderen Objekten unterscheiden kann (*object identity is immutable*),
- Ein Objekt existiert unabhängig von anderen Objekten.

1.2 Typsysteme und Datenbank-Schemata

Eine Datenbank genügt einer Struktur, die als *Schema* bezeichnet wird. Ausgehend von Typsystemen können unterschiedliche Schemata gebildet

werden. Ein Typsystem besteht aus Basis-Datentypen, Konstruktoren für Tupel und komplexe Datentypen sowie Label für Referenzen auf andere Elemente. Wir benutzen die von Schewe verwandte vereinfachte Notation für Typsysteme [Sch97, Sch01].

Definition 1.1 (Typsystem). Es sei $n \in N_+$ und ν sei ein Basis-Datentyp, d.h. $\nu \in \{BOOLEAN, REAL, INTEGER, NATURAL, STRING, \ldots\}$. ℓ bezeichne den Label-Datentyp und $\ell : \tau'$ den Datentyp für Referenzen auf Label.[5] Zur Konstruktion von Tupeln verwenden wir den Tupel-Konstruktor $\times$. Weiter seien $\{\cdot\}$ und $[\cdot]$ die Konstruktoren für endliche Mengen und endliche Listen. Dann können wir ein allgemeines System von endlichen Typen τ rekursiv definieren,[6]

$$\tau = \ell \mid \ell : \tau' \mid \nu \mid \tau_1 \times \cdots \times \tau_n \mid \{\tau_1\} \mid [\tau_1], \tag{1.2}$$

wobei die Komponenten $\tau_1, \ldots, \tau_n$ Typen seien, die rekursiv gemäß dem Typsystem (1.2) gebildet werden, jedoch ohne daß sie den Typ τ selbst als Komponente enthalten.

Einen Typ τ bezeichnen wir als *Objekt-Typ*, wenn er (mindestens) einen Label-Datentyp ℓ enthält, und als *Wert-Typ* sonst.

Analog zur Definition *relationaler Datenbank-Schemata* und *Datenbanken* nach Paredaens, De Bra, Gyssens und Van Gucht [PBGG89] führen wir *Objekt-Orientierte Datenbank-Schemata* und *Datenbanken* ein. Eine Datenbank wird als Instanz eines Datenbank-Schemas aufgefaßt, wobei sie zusätzlichen Integritätsbedingungen, den *Constraints*, genügen muß. Der wesentliche Unterschied besteht hier darin, daß wir (1) statt Attributen von Basis-Datentypen strukturierte, komplex-wertige Attribute zulassen, denen ein Typ obigen Typsystems zugrundeliegt und (2) in Attributen *Label* zur eindeutigen Identifizierung von (strukturierten, komplex-wertigen) Tupeln verwandt und diese zur Referenzierung zwischen Tupeln ausgenutzt werden können.

Basierend auf einem *Typ* des Typsystem führen wir eine *Klasse* mit *Attributen* und korrespondierenden *Wertebereichen* ein. Für einen Typ $\tau = \nu_1 \times \cdots \times \nu_n$ entspricht die Klassen-Definition dem *primitiven Relationen-Schema* nach Paredaens et al. [PBGG89], das aus Attributen von Basis-Datentypen besteht (den Spalten-Namen), wobei der Wertebereich eines Attributes durch die Menge atomarer Werte seines Datentyps gegeben ist.

Definition 1.2 (Klasse). Als *Klasse* C bezeichnen wir ein Tripel

$$C = (\tau, X, \mathrm{dom}(X)), \tag{1.3}$$

wobei

- τ ein Typ des Typsystems sei, jedoch kein Basis-Datentyp,

[5] Zur Verwendung von Labeln und Referenzen auf Label machen wir im Folgenden die Annahmen 1.8–1.10.

[6] Auf die Verwendung von Konstruktoren für Multi-Mengen $[[\tau]]$ und für Funktionen $\tau_1 \to \tau_2$ wird in dieser Arbeit nicht eingegangen.

- $X = (X_1, \ldots, X_n)$ Bezeichner für die Komponenten des Typs τ sind; X bezeichnen wir auch als (strukturierte) Attributmenge, X_i als Attribute. Die Struktur der Attributmenge X setzt sich gemäß der Konstruktion von τ zusammen.

- $\mathrm{dom}(X)$ den Wertebereich der Attributmenge bezeichnet, d.h. der zulässigen Werte; $\mathrm{dom}(X)$ ist durch den Typ τ eindeutig bestimmt (siehe die anschließende Bemerkung 1.3), wir schreiben aufgrund dessen für den Wertebereich verkürzt $\mathrm{dom}(X) = \tau$.

Bemerkung 1.3. $C = (\tau, X, \mathrm{dom}(X))$ sei eine Klasse. Der Wertebereich $\mathrm{dom}(X)$ ergibt sich aus den zulässigen Werten der in τ enthaltenen Basis-Datentypen, Label- und Referenz-Typen,[7] die mittels der Verwendung der drei Konstruktoren $\times$, $\{\,\}$, $[\,]$ den Typ τ bilden.

Z.B. ergibt sich für $C = (\tau, X, \mathrm{dom}(X))$ aus

$$\tau = \ell \times [REAL] \times NATURAL \times \{STRING\}$$

der Wertebereich $\mathrm{dom}(X)$ mit

$$\begin{aligned}\mathrm{dom}(X) \;=\; &\{\, label \in LABEL \,\} \\ &\times \{\, [x_1, \ldots, x_j] \mid j \in \mathbb{N} \wedge (x_i \text{ ist vom Typ } REAL,\, i = 1, \ldots, j) \,\} \\ &\times \{1, \ldots, \max(NATURAL)\} \\ &\times \{\, \{s \mid s \text{ ist vom Typ } STRING\} \,\}\end{aligned}$$

Umgekehrt läßt sich aus dem Wertebereich $\mathrm{dom}(X)$ auch der zugrundeliegende Typ ersehen, so daß es genügt, eine Klasse durch ihre Attributmenge und deren Wertebereich anzugeben, $C = (X, \mathrm{dom}(X))$.

Definition 1.4 (Datenbank-Schema). Ein *Datenbank-Schema* S (oder kurz ein *Schema* S) ist ein Paar

$$S = (\{C_i\}_{i=1,\ldots,n}, \mathcal{C}) \tag{1.4}$$

mit

- $\{C_i\}_{i=1,\ldots,n} = \left\{(\tau_i, X^{(i)}, \mathrm{dom}(X^{(i)})\right\}_{i=1,\ldots,n}$ einer Menge von Klassen und

- einer endlichen Menge von Integritätsbedingungen (Constraints) $\mathcal{C}$ über $\{C_i\}_{i=1,\ldots,n}$ sei (Constraints werden in der Definition 1.6 eingeführt).

Definition 1.5 (Tupel und mögliche Instanz). $S = (\{C_i\}_{i=1,\ldots,n}, \mathcal{C})$ sei ein Datenbank-Schema.

Ein *(strukturierter, komplex-wertiger) Tupel* über einer Klasse $C_i = (\tau_i, X^{C_i}, \mathrm{dom}(X^{C_i}))$ ist eine Funktion $a : X^{C_i} \to \mathrm{dom}(X^{C_i})$. Die Werte der Attributmenge X^{C_i} für einen Tupel a schreiben wir als $X^{C_i}(a)$.[8] Einen Tupel a bezeichnen wir auch als *Instanz von* C_i, Schreibweise $a :: C_i$.

Eine Menge von Tupeln a über (jeweils einer) der Klassen $\{C_i\}$, $i = 1, \ldots, n$, bezeichnen wir als *Tupelmenge A über S*. Eine Tupelmenge A wird auch als *mögliche Instanz von S* bezeichnet.

[7] Für die Label werden Werte einer Menge $LABEL$ verwandt, die auf Basis-Datentypen basieren, beispielsweise auf $LABEL = NATURAL$; Referenz-Typen haben denselben Wertebereich, da sie auf Label verweisen.

[8] In der Literatur wird alternativ zu $X^{C_i}(a)$ auch die Notation $a[X^{C_i}]$ oder $a.X^{C_i}$ verwandt.

Definition 1.6 (Constraint). S sei ein Datenbank-Schema. Ein *Constraint* $\mathcal{C}_\imath$ über S ist eine aussagenlogische Formel, die jeder möglichen Instanz A über S genau einen Wahrheitswert (entweder $TRUE$ oder $FALSE$) zuordnet.

Definition 1.7 (Datenbank-Instanz). $S = (\{C_i\}_{i=1,\ldots,n}, \mathcal{C})$ sei ein Datenbank-Schema und A eine mögliche Instanz von S.

A ist eine (echte) *Instanz* des Schemas S, wenn sie alle Constraints $\mathcal{C}_\imath \in \mathcal{C}$ erfüllt. A bezeichnen wir auch als *Datenbank*.

Tupel $a \in A$ bezeichnen wir als *Datenbank-Objekte*, wenn sie über ein Label verfügen, d.h. wenn der ihnen zugrundeliegende Typ ein Objekt-Typ ist.

Wir machen drei Annahmen für die Verwendung von Labeln in Datenbank-Objekten. Diese Annahmen sind durch entsprechende aussagenlogische Formeln auszudrücken und in die Menge von Constraints $\mathcal{C}$ eines Datenbank-Schemas aufzunehmen.

Erstens: Label sollen an einer ausgezeichneten Stelle eines Tupels enthalten sein.

Annahme 1.8 (Normalform für Label). A sei eine Datenbank über einem Schema $S = \{C_1, \ldots, C_k\}$. Jedes Datenbank-Objekt $a :: C_i$, $1 \leq i \leq k$, kann mehrere (alternative) Label des Typs ℓ enthalten. Die Label des Typs ℓ sind als *erstes* Element im Tupel-Konstruktor enthalten,[9] d.h. für die Attributmenge X^C von C gilt $\mathrm{dom}(X^C) = \tau = \ell \times \tau'$ oder $\mathrm{dom}(X^C) = \tau = \{\ell\} \times \tau'$ mit einem Wert-Typ τ'.

Zweitens: Wir erlauben Referenzen nur zwischen Datenbank-Objekten und nur innerhalb einer Datenbank.

Annahme 1.9 (wohldefinierte Objekt-Referenzierung). A sei eine Datenbank über einem Schema S. Wir nehmen an, daß jede Instanz $a \in A, a :: C$ einer Klasse $C \in S$, die eine Referenz auf ein anderes Datenbank-Objekt enthalte, selbst über ein Label verfügt.

Drittens: Je Klasse eines Schemas soll in einer Datenbank der Wert eines Labels (von einer Instanz der Klasse) eindeutig sein.

Annahme 1.10 (Eindeutigkeit von Labeln). A sei eine Datenbank über einem Schema $S = \{C_1, \ldots, C_k\}$. $X^{C_i} = \{X_1^{C_i}, \ldots, X_{n_i}^{C_i}\}$ bezeichne die Attributmenge von C_i mit $\mathrm{dom}(X_j^{C_i}) = \ell$ oder $\mathrm{dom}(X_j^{C_i}) = \{\ell\}$. Wir nehmen an, daß jeder Wert eines Labels Datenbank-Objekte einer Klasse eindeutig identifiziert,[10] d.h. für alle Datenbank-Objekte $a \in A, b \in A \backslash \{a\}$

[9] Es handelt sich nicht um eine Beschränkung der Allgemeinheit, sondern um eine Vereinfachung, die die Verarbeitung erleichtert: Label eines Objekt-Typs dürfen nur an einer (für alle Objekt-Typen festen) Position eines Tupels auftreten und nicht an beliebigen —und möglicherweise in einem Tupel an verschiedenen— Stellen sein (wie etwa in $\nu_1 \times [\{(nu_2 \times \{\ell\})\} \times \ell] \times [\ell \times \tau'])$.

[10] Falls der *Identifizierer* durch die Übereinstimmung aller Label eines Objekt-Typs gegeben sein soll, bilden wir aus allen Labeln $\{\ell_i\}_{i=1,\ldots,k}$ eines Tupels ein zusammengesetztes Super-Label $\ell^* = \ell_{j_1} + \ldots + \ell_{j_k}$, wobei wir eine deterministische Ordnung für die Reihenfolge (etwa die lexikographische Ordnung) und ein (nicht in den Labeln vorkommendes) Trennzeichen '+' verwenden.

mit $a, b :: C_i$ für ein $i \in \{1, \ldots, k\}$ gilt

$$\mathit{Label}(a) \cap \mathit{Label}(b) = \emptyset, \tag{1.5}$$

wobei die Menge aller Label eines Datenbank-Objektes definiert ist durch

$$\mathit{Label}(a) := \begin{cases} \{X_j^{C_i}(a)\} & \mathrm{dom}(X_j^{C_i}) = \ell, \\ X_j^{C_i}(a) & \mathrm{dom}(X_j^{C_i}) = \{\ell\}\,. \end{cases} \tag{1.6}$$

Um in Datenbanken Elemente gezielt selektieren, auswerten und bearbeiten zu können, benötigen wir Abfragesprachen. Eine Abfragesprache Q ermöglicht, Teilmengen von Elementen (und eine Auswahl ihrer Attribute X) aus einer Datenbank A über S auswählen und bearbeiten zu können. Q enthält Konstrukte zur Definition von Strukturen, zur Manipulation (Einfügen, Ändern, Löschen) sowie zur Selektion von Daten. Wir betrachten im Folgenden für die Identifizierung in existierenden Datenbanken nur Abfragen zur Selektion von Daten. Eine *deklarative Abfragesprache* erlaubt die Formulierung von Abfragen in einer beschreibenden Form; die Datenselektion erfolgt durch die Angabe von *Kriterien für Attribute* $X_i^C \subset X^C$.

Die *Mächtigkeit* einer Abfragesprache besteht einerseits darin, inwieweit sie es ermöglicht, Abfragen für (komplexe) Typstrukturen τ zu formulieren, bei denen Kriterien an Konstruktor-Elemente von Basis-Datentypen verwandt werden können (beispielsweise für geschachtelte Typstrukturen oder Mengen- und Listen-Typen). Andererseits ist die *Mächtigkeit* einer Abfragesprache auch durch ihre Eignung bestimmt, transitive Referenzen auf andere Elemente in der Datenbank verfolgen zu können.

Beispiel 1.11 *(SQL).* Mit der in relationalen Datenbanken verwandten deklarativen Abfragesprache SQL ist es beispielsweise möglich, für Typen, die aus Basis-Datentypen bestehen, Bedingungen zu formulieren, es ist jedoch nicht möglich, Kriterien an zusammengesetzte Typen zu formulieren. Die rekursive Abarbeitung von Referenzen auf andere Tupel werden von SQL nicht unterstützt.[11]

Eine Abfrage läßt sich als Abbildung zwischen einem Quellschema und einem Zielschema auffassen.

Definition 1.12 (Datenselektions-Abfrage). S und S' seien Schemata mit den Klassen $C_1, \ldots, C_d$ sowie $C'_1, \ldots, C'_{d'}$, $d, d' \in \mathbb{N}_+$, die jeweils einem Typ τ_{C_i} sowie $\tau_{C'_i}$ des Typsystems genügen. Eine Datenselektions-Abfrage q_S auf S ist eine Abbildung von Instanzen der Klassen $C_1, \ldots, C_d$ jeder Datenbank A über S auf Instanzen von Klassen eines Schemas S'. Eine Menge von Abfragen auf S fassen wir in einer Abfragesprache Q_S zusammen.

1.3 Identifikation in Datenbanken

Im diesem Abschnitt gehen wir auf die Identifikation von Instanzen (Tupeln) in relationalen und von Datenbank-Objekten in Objekt-Orientierten Datenbanken ein.

[11] Da in relationalen Datenbanken keine Label verwandt werden, wird die rekursive Abarbeitung von Schlüssel-Fremdschlüssel-Referenzen betrachtet.

Vorher geben wir eine Definition für die Gleichheit der Werte einer Attributmenge zweier Instanzen $a, b \in A$ an. Wir definieren die Gleichheit der Werte durch das vollständige Übereinstimmen aller Werte (d.h. aller Wert-Typen) der Attributmenge.

Definition 1.13 (Wert-Gleichheit einer Attributmenge). Für zwei Instanzen $a, b \in A$, $a, b :: C$ mit der Attributmenge X^C definieren wir die Gleichheit der Werte einer Attribut-Teilmenge $X \subset X^C$, geschrieben als $X(a) =_\tau X(b)$, für atomare Attribute und rekursiv für zusammengesetzte Attributmengen:

- *Wert-Gleichheit atomarer Attribute*: X_j sei ein Attribut der Attributmenge X mit atomaren Wert-Typen, d.h. $\mathrm{dom}(X) = \nu$, $= \ell$ oder $= \ell : \tau$. Die Gleichheit ist dann durch die Gleichheit des zugrundeliegenden Basis-Datentyps gegeben, bzw. für Attribute mit Labeln und Referenzen auf Label werden die Werte ignoriert (d.h. sie werden immer als gleich betrachtet, der Vergleichswert wird bei zusammengesetzten Attributen auf TRUE gesetzt).

  $X_j(a) =_\tau X_j(b)$ ist definiert durch:

$$X_j(a) = X_j(b) \qquad \text{für } \mathrm{dom}(X_j) = \nu \tag{1.7a}$$
$$\text{TRUE} \qquad \text{für } \mathrm{dom}(X_j) = \ell \tag{1.7b}$$
$$\text{TRUE} \qquad \text{für } \mathrm{dom}(X_j) = \ell : \tau \tag{1.7c}$$

- *Wert-Gleichheit zusammengesetzter Attributmengen*: X sei aus den Attributen (oder Attributmengen) $X_1, \ldots, X_n$ durch Tupel-, Listen- oder Mengen-Konstruktion zusammengesetzt. d.h. $\mathrm{dom}(X) = (\tau_1 \times \cdots \times \tau_n)$, $= []$ oder $= \{\}$.

  Dann ist die Wert-Gleichheit $X(a) =_\tau X(b)$ wie folgt rekursiv über ihre Attribut-Teilmengen definiert:

$$\bigwedge_{j=1}^{n} X_j(a) =_\tau X_j(b) \;\text{ für }\; X = X_1 \times \cdots \times X_n, n \in \mathbb{N}, \tag{1.8a}$$
$$\bigwedge_{j=1}^{n} X_j(a) =_\tau X_j(b) \;\text{ für }\; X = [X_1, \ldots, X_n], n \in \mathbb{N}, \tag{1.8b}$$
$$\Big(\exists \phi : \{1, \ldots, n\} \to \{1, \ldots, n\},\ \phi^{-1} \text{ existiert}^{12}$$
$$\bigwedge_{j=1}^{n} X_{\phi(j)}(a) =_\tau X_j(b)\Big) \;\text{ für }\; X = \{X_1, \ldots, X_n\}, n \in \mathbb{N}. \tag{1.8c}$$

Die Formeln (1.8) bedeuten, daß für eine zusammengesetzte Attribut-Menge genau dann Wert-Gleichheit gilt, wenn sie für alle Komponenten gilt.

[12] ϕ ist eine Permutation auf $\{1, \ldots, n\}$ und ist in (1.8c) erforderlich, da ein Mengen-Typ keine feste Ordnung hat. Die in beiden Mengen enthaltenen Werte können unterschiedlich angeordnet sein. Sofern eine Permutation der Werte existiert, die eine Menge ebenso anordnet wie die andere, sind diese Wert-gleich.

Es sei bemerkt, daß die Struktur-Gleichheit der Attributmengen Voraussetzung für die Wert-Gleichheit zweier Instanzen ist, implizit also in der Definition 1.13 enthalten ist. Wir geben ein Beispiel:

Beispiel 1.14. Wir betrachten eine Klasse $C = (\tau, X, \mathrm{dom}(X))$ mit dem Typ

$$\tau = \ell \times STRING \times NATURAL \times \{(\ell : \tau \times Name)\},$$

und der Attributmenge

$$X = (ID, Name, Geburtsjahr, Kinder),$$

mit $Kinder = \{Kind\}$ und entsprechenden Wertebereichen. Es seien folgende Instanzen von C in einer Datenbank A gegeben:

$$\begin{aligned}
a_1 &= (1, \text{'Anna'}, 1967, \{(3, \text{'Carl'}), (4, \text{'Dana'})\}) \\
a_2 &= (2, \text{'Bert'}, 1966, \{(3, \text{'Carl'}), (4, \text{'Dana'})\}) \\
a_3 &= (3, \text{'Carl'}, 1990) \\
a_4 &= (4, \text{'Dana'}, 1987) \\
a_5 &= (5, \text{'Bert'}, 1966, \{(3, \text{'Carl'})\}) \\
a_6 &= (6, \text{'Bert'}, 1966, \{(8, \text{'Dana'}), (7, \text{'Carl'})\}) \\
a_7 &= (7, \text{'Carl'}, 1990) \\
a_8 &= (8, \text{'Dana'}, 1985)
\end{aligned}$$

In A sind für *Bert*, *Carl* und *Dana* mehrere Instanzen enthalten, wobei die Wert-Gleichheit $a_3 =_\tau a_7$ gilt, da Name und Geburtsjahr übereinstimmen und das Label ID ignoriert wird. Jedoch ist $a_4 \neq_\tau a_8$, da das Geburtsjahr nicht übereinstimmt. Weiter ist $a_2 =_\tau a_6$, da Name und Geburtsjahr übereinstimmen und die Namen der Kinder nur vertauscht sind (Die Bijektion ϕ in (1.8c) ist durch $\phi(1) = 2, \phi(2) = 1$ gegeben). Für alle anderen Paare (i, j) gilt aber $a_i \neq_\tau a_j$, insbesondere gilt $a_2 \neq_\tau a_5$ und $a_5 \neq_\tau a_6$, da in a_5 nur das Kind *Carl* angegeben ist.

1.3.1 Identifikation in relationalen Datenbanken

Das Schema einer relationalen Datenbank besteht aus Klassen C_i von Typen $\tau_i = \nu_1 \times \cdots \times \nu_{n_i}$ (ν_i sind Basis-Datentypen) deren Instanzen in einer Datenbank wir je Klasse zusammenfassen und als *Tabelle* bezeichnen. Die Klasse C_i bezeichnen wir auch als *Relation*. Da die Typen τ_i keine Label enthalten, handelt es sich nicht um Datenbank-Objekte sondern um Tupel aus Basis-Datentypen. Die Identifikation von Tupeln kann also nur über ihre Werte erfolgen.

In relationalen Datenbanken werden zur Identifikation von Datensätzen innerhalb von Tabellen geeignete Attribute zu einem Schlüssel zusammengefaßt. Es wird festgeschrieben, daß ein Schlüssel immer eindeutig zu sein hat, d.h. eine Wert-Kombination der Schlüsselattribute darf nur einmal in der betreffenden Tabelle der Datenbank vorkommen. Für jede Tabelle

der Datenbank kann ein Schlüssel für die interne Unterscheidung von Datensätzen festgelegt werden, dieser wird als Primärschlüssel (*engl.: primary key*) bezeichnet. Als Schlüssel-Eigenschaft wird verlangt, daß aus dem Wert des Primärschlüssels alle anderen Attribut-Werte bestimmt werden können, was als *funktionale Abhängigkeit* bezeichnet wird.

Definition 1.15 (funktionale Abhängigkeit). $C = C(X_1, \ldots, X_l)$ sei eine Relation mit der Attributmenge $X := (X_1, \ldots, X_l)$.

(i) Eine Attributmenge $Y' \subset X$ bezeichnen wir als *funktional abhängig* von einer Attributmenge $Y \subset X$, wenn für eine Instanz $a :: C$ der Relation (einen Tupel) die Attributwerte $Y'(a)$ immer durch die Attributwerte von $Y(a)$ eindeutig bestimmt sind. Wir schreiben dafür $Y \vdash Y'$.[13]

(ii) Eine Attributmenge $Y' \subset X$ ist *voll funktional abhängig* von einer Attributmenge $Y \subset X$ genau dann, wenn $Y \vdash Y'$ gilt und Y zusätzlich folgende Minimalitätsbedingung erfüllt:

$$\forall Y'' \subset Y : \quad Y'' \nvdash Y'. \tag{1.9}$$

Definition 1.16 (Schlüssel). $C = C(X_1, \ldots, X_l)$ sei eine Relation mit der Attributmenge $X := (X_1, \ldots, X_l)$.

Eine Attribut-Teilmenge $Y \subset X$ ist ein *Schlüssel* (*engl.: key*) für die Relation C genau dann, wenn $Y \vdash X$ gilt.

Wir bezeichnen Y als *echten Schlüssel*, falls X voll funktional abhängig von Y ist. Falls $Y \vdash X$ gilt, jedoch die Minimalitätsbedingung (1.9) nicht erfüllt ist, bezeichnen wir Y als *unechten Schlüssel*.

Echte Schlüssel werden oft auch als *Kandidatenschlüssel* bezeichnet. Einer dieser Schlüssel kann je Relation als *Primärschlüssel* (*engl.: primary key*) ausgezeichnet werden, er wird dann für die Identifikation in der Relation verwandt. In der Regel wird ein echter Schlüssel ausgewählt. Primärschlüssel werden für Verknüpfungen von Tabellen einer Datenbank verwandt.

Um das Fehlen von Werten in Schlüssel-Attributen erlauben zu können, erweitern wir den Vergleich zweier Attribute auf NULL-Werte.

Definition 1.17 (NULL-Wert). Ein Attribut X_i hat einen NULL-Wert für eine Instanz $a \in A$, wenn kein Wert für X_i angegeben ist. Wir bezeichnen einen NULL-Wert mit dem Zeichen $\perp$, d.h. wir schreiben dann $X_i(a) = \perp$. Ein NULL-Wert ist nicht aus dem Wertebereich von X_i —$\perp \notin \mathrm{dom}(X)_i$; $\perp$ ist ein besonderes Zeichen, das das Fehlen jeglichen Wertes zum Ausdruck bringt.

Definition 1.18 (Übereinstimmung bis auf NULL-Werte). Es sei $Y \subset X$ eine Attributmenge einer Relation $C = C(X) = C(X_1, \ldots, X_l)$,

[13] Wir weichen hier von der üblichen Notation ab: In der Regel wird die funktionale Abhängigkeit mit $Y \to X_i$ bezeichnet, den Pfeil „→“ verwenden wir aber ausschließlich für Funktionen. Die Notation $Y \vdash X_i$ wird im formalen Modell (Abschnitt 2.2 auf Seite 48) auch für abgeleitete —und mithin funktional abhängige— Attribute verwandt.

$Y = (X_{i_1}, \ldots, X_{i_d})$ mit $1 \leq d \leq l$. Das Fehlen des Wertes eines Attributes aus dem Wertebereich bezeichnen wir als NULL-Wert, geschrieben $\perp$. Wir erweitern den Wertebereich $\mathrm{dom}(Y)$ um NULL-Werte, und fassen dann $\perp$-Werte beim Vergleich als Werte des erweiterten Wertebereiches auf (d.h. $y_j = \perp \wedge y'_j = \perp$ betrachten wir als gleich).

Wir setzen den erweiterten Wertebereich $\perp$-$\mathrm{dom}(Y)$ von Y mit

$$\perp\text{-}\mathrm{dom}(Y) := ((\mathrm{dom}(X_{i_1}) \cup \{\perp\}) \times \cdots \times ((\mathrm{dom}(X_{i_d}) \cup \{\perp\}). \tag{1.10}$$

Die um NULL-Werte ($X_i(a) = \perp$) erweiterte Gleichheit ist für $y = (y_1, \ldots, y_d)$, $y' = (y'_1, \ldots, y'_d) \in \perp\text{-}\mathrm{dom}(Y)$ definiert durch

$$y =_\perp y' :\Longleftrightarrow \Big(\forall j = 1, \ldots, d : \big((y_j = \perp \wedge y'_j = \perp) \vee \big((y_j, y'_j \neq \perp \wedge y_j = y'_j)\big)\Big). \tag{1.11}$$

Der Begriff der funktionalen Abhängigkeit läßt sich auf NULL-Werte erweitern, indem beispielsweise in der Definition 1.15 der Wertebereich $\mathrm{dom}(\cdot)$ jeweils durch $\perp$-$\mathrm{dom}(\cdot)$ ersetzt wird.[14] Damit können wir in einem Schlüssel auch Attribute mit NULL-Werten verwenden. Wenn eine Attributmenge Y als Primärschlüssel für eine Relation festgelegt ist, wird die Eindeutigkeit ihrer Werte durch eine Integritätsbedingung (*engl.: integrity constraint*) erzwungen: Diese Integritätsbedingung wird bei Einfüge- und Änderungstransaktionen geprüft — falls die Integritätsbedingung verletzt wird, wird eine Transaktion durch das Datenbank-Management-System zurückgewiesen.

Definition 1.19 (Integritätsbedingung für Primärschlüssel). Es sei $Y \subset X$ der Primärschlüssel einer Relation $C = C(X) = C(X_1, \ldots, X_l)$. A sei eine Datenbank über einem Schema S, das die Klasse C enthalte. Dann ist die *Integritätsbedingung für Primärschlüssel* (*engl.: primary key constraint*)

$$\big[\forall a \in A\, \forall b \in A \backslash \{a\},\, a, b :: C :\ \big(\neg(Y(a) =_\perp Y(b)) \wedge \wedge\, (\exists Y_i \in Y : (Y_i(a) \neq \perp \vee Y_i(b) \neq \perp))\big)\big] \tag{1.12}$$

zu erfüllen. Das heißt, die Schlüsseleigenschaft von Y besteht in der Übereinstimmung der Attributwerte Y_j zweier Tupel bzw. in dem Fehlen der Attributwerte von Y_j beider Tupel, wobei jedoch zumindest für ein Attribut Y_i Werte vorliegen.[15]

Es gibt drei Möglichkeiten, Schlüssel für eine Relation zu bilden:

- **Natürliche Schlüssel:** Eine geeignete Auswahl von Attributen wird zu einem Schlüssel deklariert. Für diese Attributauswahl muß durch Modellannahmen sichergestellt sein, daß der Schlüssel immer eindeutig ist. Diese Attribute sind üblicherweise Merkmale, die auch in der Wirklichkeit zur Identifikation verwandt werden, z.B. Ausweisnummer oder Namen und Geburtsdatum für Personen.

[14] Ein allgemeines Konzept funktionaler Abhängigkeiten für Relationen mit NULL-Werten wird in den Arbeiten von Levene und Loizou [LL95, LL99] untersucht.

[15] Falls für einen Primärschlüssel Y zusätzlich die Integritätsbedingung NOT NULL, $[\forall a \in A, a :: C\, \forall Y_i \in Y : Y_i(a) \neq \perp]$ gefordert ist, hat die Erweiterung der Gleichheit auf NULL-Werte keine Bedeutung für Y, (1.12) vereinfacht sich dann zu $[\forall a \in A\, \forall b \in A \backslash \{a\},\, a, b :: C :\ (Y(a) \neq Y(b))]$.

- **Surrogat-Schlüssel:** Die Tabelle wird um ein Attribut ergänzt und sichergestellt, daß kein Wert doppelt vorkommt (Bei jeder Einfügetransaktion wird ein neuer Wert für dieses Attribut erzeugt, d.h. ein von allen vorhandenen Werten abweichender Wert, z.B. durch Inkrementierung bei einem INTEGER-Attribut).

- **Klassifikatorische Schlüssel:** Ein klassifikatorischer Schlüssel besteht aus einem natürlichen und einem Surrogat-Schlüssel, d.h. er besteht aus einem *klassifizierenden* und einem *identifizierenden* Teil. Der klassifizierende Teil wird für einen Datensatz anhand einer Taxonomie bestimmt (z.B. Produktgruppenschlüssel für Artikel, Studiengang für Studenten etc.).

Ein Vorteil von natürlichen Schlüsseln besteht darin, daß sie für den Anwender Bedeutung haben: Mit den Werten eines natürlichen Schlüssels lassen sich die Realwelt-Objekte auch in der Realität identifizieren, während Surrogat-Schlüssel außerhalb einer Datenbank in der Regel keine Bedeutung haben.

Es kann sich bei natürlichen Schlüsseln nachteilig auswirken, daß Duplikate der Attributwert-Kombinationen verboten sind. Wenn man sich etwa auf Name, Vorname und Geburtsdatum festgelegt hat, darf es *Heinz Müller, geb. am 2.3.1947* nur einmal in der Tabelle geben, im Falle einer Volkszählungsdatei eine nicht zu vertretende Restriktion. Für Surrogat-Schlüssel spielen solche Probleme keine Rolle, sie können bei ihnen nicht entstehen, denn unabhängig vom restlichen Datensatz (und damit weitgehend unabhängig von den Modellannahmen) wird ein inhaltlich bedeutungsloser neuer Schlüsselwert erzeugt.

Da natürliche Schlüssel oft aus mehreren Attributen zusammengesetzt sind, ist der Umgang mit ihnen aufwendiger, insbesondere müssen bei der Verknüpfung von Tabellen alle Schlüsselattribute mitgeführt werden. Bei Änderungen der Werte dieser Attribute müssen die Änderungen in allen betroffenen Tabellen vorgenommen werden. Die Verwendung natürlicher Schlüssel führt damit zu einem erhöhten Speicherbedarf und größerer Fehleranfälligkeit durch die redundante Speicherung. Außerdem muß man in der Regel Performanzverluste in Kauf nehmen (z.B. bei der Bearbeitung von Abfragen über mehrere Tabellen), ein weiterer Grund, das aus dieser eher technischen Sicht Surrogat-Schlüssel oft vorgezogen werden.

Klassifikatorische Schlüssel lassen durch den klassifizierenden Teil die Einordnung in die Taxonomie erkennen und vermeiden Duplikate durch denselben Mechanismus, der für Surrogat-Schlüssel verwandt wird. Problematisch ist es jedoch, wenn die Taxonomie im Laufe der Zeit Veränderungen unterworfen ist, so daß in vorhandenen Datensätzen der klassifizierende Teil aktualisiert werden muß: Unterschiedliche Taxonomien einer Grundgesamtheit lassen sich nicht immer ineinander überführen! Als direkte Folge einer nicht überführbaren Taxonomie muß die Identifikation mittels eines anderen (neuen) Schlüssels realisiert werden.

1.3.2 Identifikation in Objekt-Orientierten Datenbanken

Grundlage der Identifikation von Datenbank-Objekten bildet das Prüfen auf Übereinstimmung von Werten und Referenzen. Anders als in relationalen Datenbanken kann aber die Struktur der Datenbank-Objekte selbst sowie die Struktur ihrer Referenz-Bäume (d.h. die bei der Traversierung der Referenzen auf andere Datenbank-Objekte entstehenden Bäume) für die Identifikation ausgenutzt werden.

1.3.2.1 Der Objekt-Graph einer Datenbank und Referenz-Bäume von Datenbank-Objekten

Wir führen die graphische Darstellung einer Datenbank in der vereinfachten Graph-Notation nach Beeri und Thalheim [BT99] ein.

Eine Datenbank A über einem Schema S kann als Objekt-Graph dargestellt werden. Ein Objekt-Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ ist definiert auf einer Menge $\mathcal{N} = \mathcal{N}_O \cup \mathcal{N}_V$, $\mathcal{N}_O \cap \mathcal{N}_V = \emptyset$ von Objekt- und Wert-Knoten (O ist die Menge der Datenbank-Objekte in A und V eine Menge atomarer Werte von Basis-Datentypen) und einer Menge von gerichteten Kanten $\mathcal{E}$ mit Kanten-Bezeichnern *Names*.

Je Instanz einer Klasse in A gibt es einen Objekt-Knoten. Wert-Knoten gibt es je einmal je vorkommendem Wert von Basisdaten-Typen und Labeln, je Klasse M_i in S und Typ-Konstruktor-Elemente (wie $\times$, $[\,], \{\,\}$) so oft wie in Instanzen verwandt. Zulässige Kanten-Bezeichner $name \in Names$ sind die Attributnamen der Attribute in X^{C_i}, der Bezeichner *element*, das für das Enthaltensein in mehrwertigen Attributen sowie für die Instanzierung einer Klasse verwandt wird, d.h. der Referenz eines Datenbank-Objektes auf seine Klasse. Ein weiterer Bezeichner ist *state*, mit dem Kanten bezeichnet werden, die Objekt-Knoten mit dem Knoten des zugehörigen Tupel-Konstruktors verbinden. Eine Kante in $\mathcal{E}$ ist bestimmt durch die Typ-Konstruktion. Eine Ausnahme bilden lediglich Referenzen auf Label $\ell : \tau$ von Datenbank-Objekten, hier verweist eine gerichtete Kante direkt auf den referenzierten Objekt-Knoten anstatt auf sein Label ℓ und wird mit dem Bezeichner des referenzierenden Typs (dessen Attributnamen) versehen.

In einem Objekt-Graphen können wir ausgehend von einem beliebigen Knoten einen Referenz-Baum durch Traversierung der gerichteten Kanten konstruieren.

Die eingeführten Begriffe erläutern wir an einem Beispiel.

Beispiel 1.20 *(Drei Freunde).* Wir betrachten eine Klasse C, $C = FRIENDS$ mit der Attributmenge $X^C = (ID, Vorname, Freunde)$ und dem Wertebereich $\mathrm{dom}(X^C) = \tau_C$,

$$\tau_C = \ell : (STRING \times \{\ell : \tau_C\}).$$

Mit C ist ein Schema S für Datenbanken A gegeben, in der befreundete Menschen mit ihren Vornamen erfaßt werden können. Es sei folgende Datenbank-Instanz A mit den drei Freunden Hans, Kurt und Egon gegeben:

$$A = \big((h, \text{„Hans“}, \{e, k\}), (k, \text{„Kurt“}, \{e, h\}), (e, \text{„Egon“}, \{h, k\})\big).$$

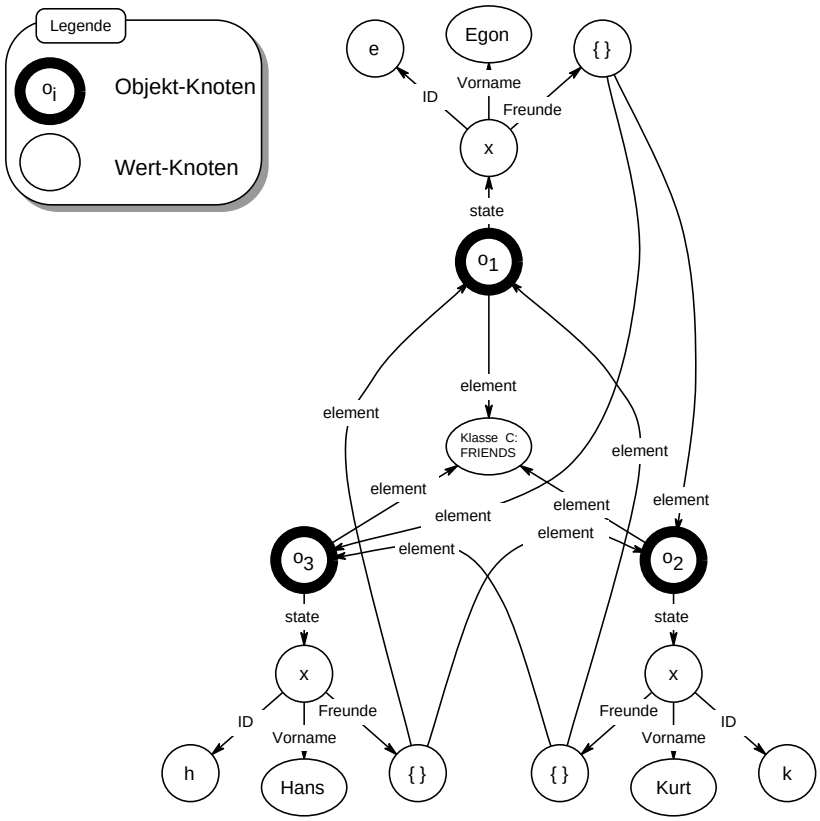


Abbildung 1.2: Der Referenz-Graph für die drei Freunde aus Beispiel 1.20

Wir erhalten den in Abbildung 1.2 gezeigten Referenz-Graph für die drei Freunde, da jeder Freund der beiden anderen ist. Für jeden der drei Freunde können wir einen abzählbar unendlichen Referenz-Baum bilden, der ausgehend von seinem Objekt-Knoten jeweils zu dessen beiden Freunden verzweigt, und dann wieder zu dessen Freunden, etc. (vgl. Abbildung 1.3). Jeder dieser drei Bäume ist rational, da es nur endlich viele verschiedene Teilbäume gibt.[16]

Abschließend sei bemerkt, daß im Falle zweier gleichnamiger Freunde, die zudem denselben Freundeskreis haben, eine Unterscheidung ihrer Instanzen in einer Datenbank dieses Schemas (zumindest anhand ihrer Vornamen und Referenzen) nicht möglich ist; beispielsweise falls drei Freunde *Werner, Wolfgang* und *Wolfgang* heißen würden.

Lemma 1.21. *Sei A eine nicht-leere Datenbank über einem Schema S. Dann ist der Objekt-Graph $\mathcal{G}$ von A endlich, d.h. er enthält endlich viele Knoten und Kanten.*

Beweis. A ist eine endliche Instanz von S, enthält also endlich viele Elemente. Jedes Element $a \in A$ ist Instanz einer Klasse C aus S und verfügt

[16] Es gibt genaugenommen sieben verschiedene endliche Teilbäume, die jeweils nur aus einem Wurzelknoten bestehen, jeweils mit den drei Werten der Label (ID's), den drei Vornamen und der Klasse. Jeder Teilbaum, der mit einem der anderen der neun Knoten beginnt (d.h. mit einem Objekt-Knoten oder dem Knoten eines der Tupel- und Mengen-Konstruktoren) ist unendlich, aber durch den Wurzelknoten eindeutig bestimmt.

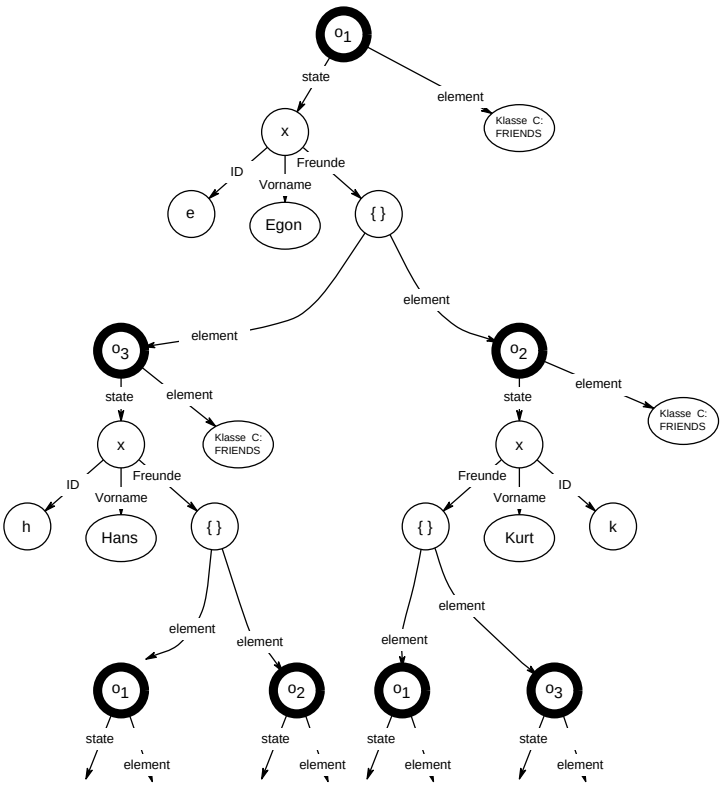


Abbildung 1.3: Der Referenz-Baum für einen Tupel der Datenbank aus Beispiel 1.20

über Werte endlich vieler Komponenten-Typen des C definierenden Typs. Jeder dieser Komponenten-Typen ist einelementig (für Label und Basis-Datentypen) oder hat endlich viele Unter-Komponenten-Typen (für alle Konstruktoren). Also gibt es nur endlich viele Knoten und Kanten zwischen ihnen. □

Lemma 1.22. *Der Referenz-Baum eines jeden Knotens im Objekt-Graphen $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ einer Datenbank A ist eindeutig bestimmt.*

Beweis. Wir führen den Beweis mittels vollständiger Induktion über die Tiefe des Baumes. $node \in \mathcal{N}$ sei ein beliebiger, aber fester Knoten.

Induktionsanfang ($i = 1$):

Der Wurzel-Knoten eines Referenz-Baumes ist eindeutig bestimmt durch den Knoten $node$.

Induktionsschritt ($i \Longrightarrow (i+1)$):

Der Referenzbaum $\mathcal{T}$ sei bis zur Tiefe i eindeutig bestimmt. Wir zeigen, daß er dann auch bis zur Tiefe $i+1$ eindeutig bestimmt ist. Sei also $node' \in \mathcal{N}$ ein Knoten der Tiefe i in $\mathcal{T}$. Dann hat $node'$ genau dann im Referenzbaum $\mathcal{T}$ eine abgehende Kante zu einem Knoten $node'' \in \mathcal{N}$ mit dem Bezeichner $name$, wenn im Objekt-Graph eine gerichtete Kante $(node', name, node'') \in \mathcal{E}$ existiert. Da die Kanten des Objekt-Graphes fest und eindeutig sind, ist der Induktionsschritt vollzogen. □

Der folgende Satz besagt, daß die Referenz-Bäume zwar unendlich sein können, aber eine *reguläre Struktur* aufweisen, die als rationaler Baum bezeichnet wird (siehe dazu Courcelle [Cou83], Harel [Har86] sowie Thomas [Tho90]).

Satz 1.23. *Sei A eine Datenbank über einem Schema S und $\mathcal{G}$ der zugehörige Objekt-Graph.*

Dann ist der Referenz-Baum jedes Elementes $a \in A$ rational, d.h. er ist endlich verzweigend und besitzt nur endlich viele voneinander unterscheidbare Teilbäume.

Beweis. Wir haben zu zeigen, daß (1) jeder Referenz-Baum endlich verzweigend ist, sowie daß (2) jeder Referenz-Baum nur endlich viele verschiedene Teilbäume hat.

Zu (1): $\mathcal{G}$ ist gemäß Lemma 1.21 endlich. $k^* \in \mathbb{N}$, $k^* < \infty$ bezeichne das Maximum aller von einem Knoten abgehenden gerichteten Kanten. Also ist jeder Referenz-Baum über $\mathcal{G}$ endlich verzweigend.

Zu (2): $max \in \mathbb{N}$ bezeichne die Anzahl der Knoten in $\mathcal{G}$. Der Referenz-Baum ausgehend von einem beliebigen Knoten in $\mathcal{G}$ ist nach Lemma 1.22 eindeutig bestimmt. Es gibt also höchstens max unterschiedliche Referenzbäume in $\mathcal{G}$. Sei nun $\mathcal{T}$ der Referenz-Baum eines Elementes $a \in A$. Jeder Teilbaum $\mathcal{T}'$ von $\mathcal{T}$ hat einen Knoten aus $\mathcal{G}$ als Wurzel-Knoten. Also ist die Anzahl der unterschiedlichen Teilbäume von $\mathcal{T}$ durch max beschränkt. □

Wir geben eine endliche Repräsentation für Referenz-Bäume an. Dazu definieren wir eine Nachfolger-Relation auf Bäumen.

Definition 1.24 (Nachfolger-Relation auf einem Baum). Sei $\mathcal{T}$ der Referenz-Baum eines Datenbank-Objektes $a \in A$. Sei $node \in \mathcal{T}$ ein Knoten, der kein Blatt-Knoten ist.[17] Sei $\mathcal{T}'$ der Teilbaum von $\mathcal{T}$, der $node$ als Wurzel-Knoten hat.

Einen Knoten $node' \in \mathcal{T}$ bezeichnen wir dann als *Nachfolger* von $node$ in $\mathcal{T}$, wenn der Knoten $node'$ in $\mathcal{T}'$ als Nicht-Wurzel-Knoten enthalten ist. Schreibweise $node \lhd node'$.

Die endliche Repräsentation eines Referenz-Baumes $\mathcal{T}$ konstruieren wir wie folgt. Falls ein Objekt-Knoten in einem unendlichen Teilbaum von $\mathcal{T}$ wiederholt enthalten ist (d.h. zyklisch referenziert wird), können wir den Teilbaum, der mit dem zweiten Auftreten dieses Objekt-Knotens beginnt, abschneiden, da seine abgehenden Kanten in dem verbleibenden Referenz-Baum schon enthalten sind. Aufgrund der Endlichkeit des Objekt-Graphen muß jeder von der Wurzel ausgehende Pfad in einem unendlichen Teilbaum eine Wiederholung von (mindestens) einem Objekt-Knoten enthalten. Wir erhalten daher eine endliche Repräsentation für einen Referenz-Baum, wenn wir in allen unendlichen Teilbäumen diejenigen Verzweigungen abschneiden, die bei der ersten Wiederholung eines Objekt-Knotens beginnen.

[17] Ein Knoten $node \in \mathcal{T}$ ist genau dann ein Blatt-Knoten von $\mathcal{T}$, wenn der Teilbaum in $node$ aus dem Wurzel-Knoten $node$ besteht.

Definition 1.25 (Beginn und Teil eines Referenz-Baumes). $\mathcal{T}$ sei der Referenz-Baum eines Datenbank-Objektes $a \in A$.

Einen endlichen Baum $\mathcal{T}'$ bezeichnen wir als *Beginn von* $\mathcal{T}$, wenn er aus $\mathcal{T}$ hervorgeht, indem unendliche Teilbäume in $\mathcal{T}$ direkt hinter Objekt-Knoten abgeschnitten werden.

Einen endlichen Baum $\mathcal{T}'$ bezeichnen wir als *(endlichen) Teil von* $\mathcal{T}$, wenn er aus $\mathcal{T}$ hervorgeht, indem (endliche und/oder unendliche) Teilbäume von $\mathcal{T}$ abgeschnitten werden.

Satz 1.26 (Endliche Repräsentation eines Referenz-Baumes). *$\mathcal{T}$ sei der Referenz-Baum eines Datenbank-Objektes $a \in A$.*

Dann existiert ein Beginn $\mathcal{T}'$ von $\mathcal{T}$, der für jeden Blatt-Knoten $node \in \mathcal{T}'$ die Formel

$$node \in \mathcal{N}_V \ \vee \ \big(node \in \mathcal{N}_O \ \wedge \ node \lhd node\big). \tag{1.13}$$

erfüllt. Durch $\mathcal{T}'$ ist $\mathcal{T}$ eindeutig bestimmt, $\mathcal{T}'$ bezeichnen wir als endliche Repräsentation von $\mathcal{T}$. *Zudem existiert immer eine endliche Repräsentation von $\mathcal{T}$, die minimal bezüglich der Größe des entstehenden Baumes ist.*

Beweis. Es sind Existenz, Eindeutigkeit (des repräsentierten Referenz-Baumes) sowie Minimalität zu zeigen.

Existenz: Falls $\mathcal{T}$ endlich ist, ist die $\mathcal{T}'$ gleich $\mathcal{T}$. Andernfalls existieren nach Satz 1.23 endlich viele Teilbäume von $\mathcal{T}$ — $\mathcal{T}$ ist ein rationaler Baum. Aufgrund der Endlichkeit der Knoten-Menge im Objekt-Graphen (Lemma 1.21) müssen in jedem unendlichen Pfad eines unendlichen Teilbaumes Objekt-Knoten *node* wiederholt enthalten sein, die also die Nachfolger-Relation $node \lhd node$ erfüllen. Die unendlichen Pfade schneiden wir dann hinter diesen Objekt-Knoten ab, d.h. sie werden zu einem Blatt-Knoten in $\mathcal{T}'$, indem wir alle in $\mathcal{T}$ von ihnen abgehenden Kanten weglassen. Es bleibt zu zeigen, daß es genügt, den Referenz-Baum an endlich vielen Objekt-Knoten abzuschneiden, obwohl die Kardinalität der Menge aller Referenz-Pfade überabzählbar sein kann.[18]

Sei max die Anzahl von Objekt-Knoten im Objekt-Graph (es gilt $max < \infty$) und $d \in \mathbb{N}$ die maximale Anzahl von Wert-Knoten, die bei der Referenzierung zwischen zwei Objekt-Knoten im Objekt-Graphen auftreten können, ohne daß zwischen ihnen ein weiterer Objekt-Knoten liegt, die also einander direkt referenzieren. $\mathcal{T}$ ist aufgrund seiner Rationalität endlich verzweigend, so daß ein Beginn von $\mathcal{T}$ der Tiefe $(d \cdot (max + 1))$ endlich viele Pfade hat (d.h. wir schneiden den Baum nach allen $(d \cdot (max + 1))$-ten Knoten ab und verkürzen weiter bis zum letzten enthaltenen Objekt-Knoten). Jeder der Pfade von der Wurzel des Referenz-Baumes zu Blatt-Knoten, die

[18] Die Kardinalität der Menge aller Referenz-Pfade kann überabzählbar sein, da sich Objekt-Graphen konstruieren lassen, deren Referenz-Pfade durch die Menge $\mathbb{R}$ der reellen Zahlen darstellbar sind. Der aus dem Objekt-Graph $\mathcal{G}$ des Beispiels 1.20 auf Seite 30 resultierende Referenz-Baum eines Datenbank-Objektes (vgl. Abbildung 1.3) läßt sich als Repräsentation des Intervalls [0,1] interpretieren, wenn eine b-adische Zahlen-Darstellung mit der Basis 3 und den Ziffern 0,1,2 zugrundegelegt wird. Eine reelle Zahl $x \in [0, 1]$ ist dann eine unendliche Sequenz der Ziffern 0,1,2. Wenn jedem der Objekt-Knoten der drei Freunde eine der Ziffern 0,1,2 zugeordnet wird, existiert zu jeder reellen Zahl des Intervalls [0,1] ein Pfad in $\mathcal{G}$.

Objekt-Knoten sind, enthält deshalb $(max + 1)$ Objekt-Knoten. Folglich muß mindestens ein Objekt-Knoten aus dem Objekt-Graphen wiederholt vorkommen.

Eindeutigkeit: Sei $\mathcal{T}'$ eine endliche Repräsentation eines Referenz-Baumes, $\mathcal{T}'$ sei also endlich und (1.13) sei erfüllt. Wir geben an, wie der vollständige Referenz-Baum aus einer seiner endlichen Repräsentationen rekonstruiert werden kann. Sei *node* ein Objekt-Knoten, der ein Blatt-Knoten in $\mathcal{T}'$ sei. Wegen (1.13) existiert ein Teilbaum $\mathcal{T}''$ von $\mathcal{T}'$, der *node* als Wurzel-Knoten hat aber nicht nur aus *node* besteht. Wir ersetzen in $\mathcal{T}'$ den Blatt-Knoten *node* durch den gesamten Teilbaum $\mathcal{T}''$. Unendlich oft wiederholtes Anwenden dieser Ersetzungsregel für alle Blatt-Knoten liefert genau einen Referenz-Baum.

Minimalität: Wenn alle unendlichen Referenz-Pfade immer beim *zweiten* Auftreten des ersten wiederholten Objekt-Knotens *node* abgeschnitten werden (d.h. falls $\forall node', node \lhd node' \lhd node : \neg(node' \lhd node')$ gilt), erhalten wir die minimale endliche Repräsentation. □

Bemerkung 1.27. Mit Satz 1.23 können wir für jede Datenbank die endliche Menge der rationalen Referenz-Bäume ermitteln und auswerten. Jedoch läßt sich dieses in praktischen Anwendungen kaum realisieren. Dazu gibt es mehrere Gründe:

- **Größe eines Referenz-Baumes**. Obgleich die Referenz-Bäume aufgrund ihrer Rationalität sämtlichst endlich repräsentierbar sind, können in der endlichen Repräsentation eines Referenz-Baumes jedoch Elemente aus A mehrfach enthalten sein, die endliche Repräsentation eines Referenz-Baumes kann im Extremfall größer werden als die zugrundeliegende Datenbank selbst.[19]

- **Veränderungen der Referenz-Bäume**. Die Referenz-Bäume von Datenbank-Objekten ändern sich durch schreibende Operationen (Einfügen, Löschen oder Ändern von Elementen). Also gibt es durchaus verschiedene endliche Mengen von Referenz-Bäumen für Datenbanken eines Schemas. Im Allgemeinen läßt sich auch nicht sicherstellen, daß die Vereinigung dieser endlichen Mengen wiederum endlich ist, da die Anzahl der möglichen Datenbanken, der Instanzen eines Schemas unbeschränkt ist.

Aufgrund der Ressourcen-Beschränktheit läßt sich daher ein vollständiges Berechnen und Auswerten aller Referenz-Bäume einer Datenbank für praktische Anwendungen kaum ausnutzen. Wir müssen daher eine Beschränkung der zu untersuchenden Referenz-Strukturen vornehmen. Dazu gibt es mehrere Alternativen, beispielsweise indem wir nur die Werte vergleichen und Referenzen nicht untersuchen (*Wert-basierte Identifikation*) oder indem die Tiefe der untersuchten Referenzen in den Referenz-Bäumen begrenzt wird (*Identifikation auf n-beschränkten Referenz-Bäumen*); beides wird in der

[19] Aus der Konstruktion der endlichen Repräsentation im Beweis des Satzes 1.26 kann man ersehen, daß in einem Referenz-Pfad der endlichen Repräsentation im Extremfall alle Objekt-Knoten vorkommen können, und zwar wenn es eine Kette von Referenzen der Art o_i referenziert o_{i+1} für alle $i < max$ und o_{max} referenziert o_1.

Folgerung 1.30 auf Seite 38 besprochen. Da in der Regel nicht die gesamte (Referenz-) Struktur zur Identifikation benötigt wird, sondern bereits Teile davon Tupel eindeutig identifizieren können, führen wir *identifizierende Attributmengen* und *identifizierende Referenz-Strukturen* ein, vgl. Definitionen 1.31 und 1.33 auf Seite 38. Diese beiden Identifizierungsansätze sind im Übrigen die Grundlage der Identifizierung von Realwelt-Objekten, die im formalen Modell des folgenden Kapitels beschrieben wird.

1.3.2.2 Identifizierungsansätze

Analog zur Integritätsbedingung für Primärschlüssel in relationalen Datenbanken (vgl. Definition 1.19 auf Seite 28) ist es erforderlich, Integritätsbedingungen in einer Objekt-Orientierten Datenbank zu formulieren, deren Erfüllung durch das Objekt-Orientierte Datenbank-Management-System sichergestellt werden muß. Diese Integritätsbedingungen basieren auf Objektfunktionalen Abhängigkeiten, die wir im Folgenden darstellen. Das Konzept Objekt-funktionaler Abhängigkeiten stellt eine Verallgemeinerung des Konzeptes funktionaler Abhängigkeiten in einzelnen Relationen relationaler Datenbanken dar und wurde in der Dissertation von J. Rasch [Ras98] als Grundlage der wert-basierten Identifikation verwandt. Objekt-funktionale Abhängigkeiten werden auf dem Schema S einer Datenbank A formuliert, neben den Abhängigkeiten zwischen Werten von Subtypen einer Klasse in S können Zusammenhänge zwischen Instanzen mehrerer Klassen unter Berücksichtigung der Referenzierung über Label betrachtet werden.

Definition 1.28 (Constraints für Objekt-funktionale Abhängigkeiten). Ein *Constraint* $\mathcal{C}_i$ ist eine aussagenlogische Formel für Klassen C_i eines Schemas S, er enthält die Klassen C_i als freie Variablen und wird geschrieben als $\mathcal{C}_i(C_{j_1}, \ldots, C_{j_l})$, für ein $l \in \mathbb{N}_+$, $1 \leq l \leq d$.

Ein Constraint kann folgende atomaren Terme aufweisen:

- *Wert-Vergleiche für Tupel* $a =_\tau b$, die Instanzen einer Klasse sind, Werte von Wert-Typen in τ verglichen werden, vgl. Definition 1.13 auf Seite 25.
- *Wert-Vergleiche für Basis-Datentypen* $x \diamond y$ für Instanzen (Werte) x, y eines Basis-Datentyps ν für $\diamond \in \{<, \leq, =, \geq, >, \neq\}$.
- *Referenzen*, $X_j^C(a) \rightarrowtail b$ bezeichnet die Referenz eines Datenbank-Objektes a auf ein Datenbank-Objekt b mittels der Angabe eines Labels des Typs ℓ von b als Wert des Attributes X_j^C, das den Wertebereich $\mathrm{dom}(X_j^C) = \ell : \tau$ habe.

Als Konzepte für die Gleichheit von Datenbank-Objekten stehen mehrere Ansätze zur Verfügung, die sich insbesondere darin unterscheiden, auf welche Weise Referenzen zu anderen Datenbank-Objekten berücksichtigt werden. Mit jedem Gleichheits-Konzept läßt sich ein Ansatz zur Identifikation von Datenbank-Objekten realisieren. Wir gehen im Folgenden auf die *Wert-Gleichheit* (*engl.: value equality*), die *flache Gleichheit* (*engl.: shallow equality*) und die *tiefe Gleichheit* (*engl.: deep equality*) ein. Zur Begründung

der Identifikation auf n-beschränkten Referenz-Bäumen geben wir noch die *n-beschränkte tiefe Gleichheit* (*engl.: n-bounded deep equality*) an.

Definition 1.29 (Übereinstimmung von Datenbank-Objekten). Sei D eine Datenbank über einem Schema $S = \{C_1, \ldots, C_k\}$. $a, b \in A$ seien zwei Datenbank-Objekte einer Klasse, $a, b :: C_i$, $i \in \{1, \ldots, k\}$ mit der Attributmenge X^{C_i}.

Wir definieren zwei Konzepte für die *die lokale Gleichheit* zweier Datenbank-Objekte:

- *Wert-Gleichheit*: a und b bezeichnen wir als *Wert-übereinstimmend* (*engl.: value equal*), wenn Struktur und Werte der Attributmenge X^{C_i} übereinstimmen, d.h. falls $X^{C_i}(a) =_\tau X^{C_i}(b)$ gilt. Schreibweise $a \overset{\mathsf{v}}{=} b$.

- *Flache Gleichheit*: a und b bezeichnen wir als *flach übereinstimmend* (*engl.: shallow equal*), wenn sie Wert-übereinstimmend sind und dieselben Datenbank-Objekte referenziert werden, d.h. falls gilt

$$(a \overset{\mathsf{v}}{=} b) \wedge \big(\forall c \in A, \forall j \text{ mit } \mathrm{dom}(X_j^{C_i}) = (\ell : \tau) : (X_j^{C_i}(a) \rightarrowtail c \iff X_j^{C_i}(b) \rightarrowtail c)\big). \tag{1.14}$$

  Schreibweise $a \overset{\mathsf{s}}{=} b$.

Außerdem geben wir Konzepte für die *globale Gleichheit* zweier Datenbank-Objekte an, die auf der vollständigen bzw. partiellen Übereinstimmung aller Werte-Knoten der Referenz-Bäume basieren:

- *Tiefe Gleichheit*: a und b bezeichnen wir als *tief übereinstimmend* (*engl.: deep equal*), wenn eine Struktur-Gleichheit beider Referenz-Bäume vorliegt sowie alle Wert-Knoten der Referenz-Bäume übereinstimmende Werte aufweisen.[20] Schreibweise $a \overset{\mathsf{d}}{=} b$.

- *n-beschränkte tiefe Gleichheit*: Sei $n \in \mathbb{N}$. a und b bezeichnen wir als *n-beschränkt tief übereinstimmend* (*engl.: n-bounded deep equal*), wenn eine Struktur-Gleichheit und Gleichheit aller Wert-Knoten beider Referenz-Bäume bis zur Referenzierungs-Tiefe n vorliegen (d.h. für alle Objekt-Knoten, die transitiv über bis zu $(n-1)$ Objekt-Knoten referenziert werden — die Referenz-Bäume werden vor dem $(n+1)$-ten referenzierten Objekt-Knoten abgeschnitten). Schreibweise $a \overset{\mathsf{d,n}}{=} b$.

Aus den vier Gleichheits-Konzepten leiten wir jeweils einen Identifikationsansatz ab und geben Constraints an, die für eine Datenbank zu erfüllen sind, wenn diese angewandt werden sollen.

[20] Um die tiefe Gleichheit formal angeben zu können ist es erforderlich, Graph-Automorphismen einzuführen (ein Graph-Automorphismus ist eine Knoten- und stark Kanten-erhaltende Abbildung des Objekt-Graphen in sich selbst). Zwei Objekt-Knoten $node, node'$ sind dann ununterscheidbar (also die ihnen zuzuordnenden Datenbank-Objekte tief gleich), wenn ein Graph-Automorphismus existiert, der $node$ auf $node'$ abbildet (vgl. [BT99]).

Folgerung 1.30 (Identifikationsansätze). *S sei ein Schema mit Klassen C_j. Dann erhalten wir aus den Gleichheits-Konzepten aus Definition 1.29 die folgenden Identifikationsansätze.*

1. *Die folgende Objekt-funktionale Abhängigkeit sei für alle Zustände von A erfüllt:*

$$\mathcal{C}_{\textit{Value-Id}} := [\forall C_j \in S\, \forall a \in A\, \forall b \in A \backslash \{a\},\, a, b :: C_j : \neg(a \overset{\mathsf{v}}{=} b)] \quad (1.15)$$

   Dann können in A Datenbank-Objekte durch ihre Werte identifiziert werden(Wert-basierte Identifizierung*).*

2. *Falls die Objekt-funktionale Abhängigkeit*

$$\mathcal{C}_{\textit{Shallow-Id}} := [\forall C_j \in S\, \forall a \in A\, \forall b \in A \backslash \{a\},\, a, b :: C_j : \neg(a \overset{\mathsf{s}}{=} b)] \quad (1.16)$$

   für alle Zustände von A erfüllt ist, können Datenbank-Objekte in A durch ihre (unterschiedlichen) Werte oder Referenzen auf andere Datenbank-Objekte identifiziert werden.

3. *Falls die Objekt-funktionale Abhängigkeit*

$$\mathcal{C}_{\textit{Deep-Id}} := [\forall C_j \in S\, \forall a \in A\, \forall b \in A \backslash \{a\},\, a, b :: C_j : \neg(a \overset{\mathsf{d}}{=} b)] \quad (1.17)$$

   für alle Zustände von A erfüllt ist, können in A Datenbank-Objekte durch die Struktur und Werte ihrer Referenzbäume (respektive deren endliche Repräsentation) identifiziert werden.

4. *Sei $n \in \mathbb{N}$. Falls die Objekt-funktionale Abhängigkeit*

$$\mathcal{C}_{\textit{n-Deep-Id}} := [\forall C_j \in S\, \forall a \in A\, \forall b \in A \backslash \{a\},\, a, b :: C_j : \neg(a \overset{\mathsf{d},n}{=} b)] \quad (1.18)$$

   für alle Zustände von A erfüllt ist, können Datenbank-Objekte in A durch die Struktur und Werte ihrer Referenzbäume bis zur Referenzierungs-Tiefe n identifiziert werden.

Analog zu dem Schlüssel-Konzept in relationalen Datenbanken suchen wir in Objekt-Orientierten Datenbanken nach identifizierenden Attribut-Teilmengen der Attributmenge einer Klasse (genaugenommen handelt es sich um Teilstrukturen).

Definition 1.31 (Identifizierende Attributmenge). *A* sei eine Datenbank über S mit einer Klasse C, die die Attributmenge $X^C = \{X_1^C, \ldots, X_k^C\}$ habe. Sofern $X \subset X^C$ den Constraint

$$\mathcal{C}_{\text{Attr-Id}}(C_j) := [\forall a \in A\, \forall b \in A \backslash \{a\},\, a, b :: C_j : \neg(X(a) =_\tau X(b))] \quad (1.19)$$

für alle Zustände von A erfüllt, bezeichnen wir X als *identifizierende Attributmenge* von C.

Beispiel 1.32 *(Identifikation mit OID's).* A sei eine Datenbank über S mit einer Klasse C, die die Attributmenge $X^C = \{X_1^C, \ldots, X_k^C\}$ habe. Die Klasse C habe für jede ihrer Instanz genau ein Label, d.h. es sei $\text{dom}(X_1^C) = \ell$. Wegen Annahme 1.10 auf Seite 23 sind Label eindeutig, also gilt (1.19) für die Label X_1^C. Dann sind die Label *Objekt-Identifizierer (OID's)* für Instanzen von C in A.

Definition 1.33 (Identifizierende Referenz-Struktur). A sei eine Datenbank über S mit einer Klasse C, die die Attributmenge $X^C = \{X_1^C, \ldots, X_k^C\}$ habe. $\mathcal{T}$ bezeichne den Referenz-Baum eines Datenbank-Objektes $a \in A$.

Einen endlichen Teil $\mathcal{T}'$ von $\mathcal{T}$ bezeichnen wir als *identifizierende Referenz-Struktur*, wenn für alle $a \in A$ und alle $b \in A\backslash\{a\}$, die Instanzen einer Klasse C_j sind, folgendes gilt: Die endlichen Teile $\mathcal{T}'_a$ und $\mathcal{T}'_b$ ihrer Referenz-Bäume sind Struktur-gleich und alle ihre Wert-Knoten stimmen überein.

Um für eine *identifizierende Referenz-Struktur* einen Constraint angeben zu können, bilden wir eine zusammengesetzte Klasse, deren Instanzen dem endlichen Teil $\mathcal{T}'$ des Referenz-Baumes $\mathcal{T}$ entspricht und alle Werte von $\mathcal{T}'$ aufnimmt (Diese Klasse entspricht einer *Sicht* (*engl.: view*) auf die Datenbank, die genau den endlichen Teil $\mathcal{T}'$ des Referenzbaumes selektiert).

Basis des zusammengesetzten Typ-Konstruktors $\tau_{C_j^*}$ ist τ_{C_j}. Zunächst werden alle nicht in $\mathcal{T}'$ enthaltenen Komponenten-Typen entfernt. Dann ersetzen wir in $\tau_{C_j^*}$ für alle enthaltenen Referenz-Typen $\ell : \tau_{C_i}$ den Typ $\ell : \tau_{C_i}$ jeweils durch den Typ τ_{C_i} der referenzierten Klasse C_i, wobei nur die in $\mathcal{T}'$ enthaltenen Komponenten-Typen übernommen werden. Referenz-Typen $\ell : \tau_{C_{i'}}$, die im Typ-Konstruktor jeder referenzierten Klasse C_i enthalten sind, werden ebenso durch den referenzierten Typ $\tau_{C_{i'}}$ ersetzt, insofern dessen Wert-Knoten ebenfalls in $\mathcal{T}'$ enthalten sind. Die Attributmenge $X^{C_j^*}$ der zusammengesetzten Klasse C_j^* und deren Werte erhält man entsprechend der Konstruktion aus den Attributmengen und -Werten der in $\mathcal{T}'$ enthaltenen Datenbank-Objekte.

Dank der Konstruktion der zusammengesetzten Klasse C_j^* können wir die Referenz-Struktur-basierte Identifikation auf die Identifikation mittels identifizierender Attributmenge zurückführen.

Sofern der folgende Constraint für alle Zustände von A erfüllt ist

$$\mathcal{C}_{\text{Ref-Id}}(C_j) := [\forall a \in A\, \forall b \in A\backslash\{a\},\, a, b :: C_j : \neg(X^{C_j^*}(a) =_\tau X^{C_j^*}(b))], \quad (1.20)$$

bezeichnen wir $X^{C_j^*}$ als *identifizierende Referenz-Struktur* von C.

Beispiel 1.34. Wir betrachten eine Datenbank, die Stammbaum-Daten über Personen in Instanzen einer Klasse C vorhält, und jeweils den Namen und das Geburtsdatum sowie die Angabe von Vater und Mutter über Referenzen auf deren Datenbank-Objekte erfaßt. Wir haben also die Attributmenge

$$X^C = (X_1, \ldots, X_5) = \{ID, Name, GebDat, Mutter, Vater\}$$

mit dem Wertebereich

$$\text{dom}(X^C) = \tau_C = \ell \times STRING \times DATE \times \ell : \tau \times \ell : \tau.$$

Falls eine Instanz von C in A über den eigenen Namen und Geburtsdatum sowie die Namen und Geburtsdaten von Mutter und Vater identifiziert werden soll, definieren wir den zusammengesetzten Typ τ_{C^*} entsprechend Definition 1.33. Der zusammengesetzte Typ τ_{C^*} ergibt sich dann mit

$$\tau_{C^*} = \ell \times STRING \times DATE \times \tau_C^v \times \tau_C^v$$

(mit $\tau_C^v = STRING \times DATE$), die zugehörige Attributmenge ist durch

$$X^{C^*} = \{ID, Name, GebDat, \\ Mutter\_Name, Mutter\_GebDat, \\ Vater\_Name, Vater\_GebDat\}$$

gegeben. Die identifizierende Referenz-Struktur wird dann durch die konstruierte identifizierende Attributmenge X^{C^*} dargestellt.

1.3.3 Identifizierung in multiplen Datenbanken

Wenn aus mehreren Datenbanken Daten über Realwelt-Objekte eines Universums verfügbar sind, möchte man diese je Realwelt-Objekt zusammenfassen. Wir benötigen hierfür einen Datenbank-übergreifenden Identifizierungsansatz. Wie eingangs in Abschnitt 1.1.1 auf Seite 19 angesprochen, lassen sich unterschiedlich umgesetzte Identifikationsansätze für die Identifizierung in multiplen Datenbanken verwenden, wenn eine Zuordnung der jeweils modellierten Objekt-Identitäten angegeben werden kann.

Im Falle verteilter oder föderierter Datenbank-Systeme lassen sich die genannten Identifizierungsansätze übergreifendend realisieren, sofern eine Softwarekomponente die gleichzeitige Erfüllung der entsprechenden Integritätsbedingungen in allen beteiligten Datenbanken überwacht. Die übergreifende Identifikation ist infolgedessen unproblematisch. Dieser Fall wird deshalb in dieser Arbeit nicht untersucht.

Für autonome Datenbanken, bei denen das Datenbank-übergreifende Kontrollieren von Integritätsbedingungen nicht gegeben ist, ist eine übergreifende Identifizierung jedoch erschwert. Falls die Datenbanken gemeinsame, identifizierende Attributmengen oder Referenz-Strukturen aufweisen (oder sich solche ableiten lassen), können diese für die Identifizierung genutzt werden. Sofern die verfügbaren Daten fehlerfrei sind, sind die im Abschnitt 1.3.2.2 eingeführten Identifizierungsansätze anwendbar. Die Annahme der Fehlerfreiheit ist jedoch für praktische Anwendungen oft nicht aufrecht zu erhalten.

Bemerkung 1.35. Für zwei Datenbank-Objekte aus unterschiedlichen Datenbanken ist einerseits aus der Nicht-Übereinstimmung identifizierender Attributmengen nicht zwingend zu schlußfolgern, daß sich die Datenbank-Objekte auf unterschiedliche Realwelt-Objekte beziehen. Andererseits kann es auch vorkommen, das die Werte identifizierender Attributmengen übereinstimmen, obwohl sich zwei Datenbank-Objekte auf verschiedene Realwelt-Objekte beziehen.

Für zwei Datenbanken mit unterschiedlichen Schemata, die Daten über Realwelt-Objekte eines Universums als Instanzen jeweils einer Klasse enthalten, ist eine Vorverarbeitung notwendig. Es muß zunächst eine gemeinsame identifizierende Attributmenge oder Referenz-Struktur aus den Schemata gewonnen werden. Dabei kann es durch die unterschiedliche Repräsentation der Realwelt-Objekte in den beiden Datenbanken dazu kommen, daß die gemeinsame identifizierende Attributmenge oder Referenz-Struktur je Realwelt-Objekt mehr oder weniger stark variierende Daten

aufweist, so daß eine übergreifende Identifizierung mit der Auswertung von Wert-Übereinstimmungen versagen wird.

Die im vorigen Abschnitt dargelegten Identifizierungsansätze basieren auf Gleichheit, nämlich einer Strukturgleichheit und der Übereinstimmung der strukturgleichen Attributmengen. Leider sind die in realen Datenbanken vorgehaltenen Daten oft fehlerhaft, unvollständig oder veraltet. In vielen Fällen ist auch unbekannt, wie die Fehler entstanden sind oder wie hoch die Fehlerrate eines Attributes ist. Dadurch ist die Anwendbarkeit dieser Ansätze auf multiple autonome Datenbanken nur begrenzt möglich und in der Regel mit hohem (zudem meist manuellem) Aufwand verbunden, da eine konsequente (und für alle beteiligten Datenbanken konsistente) Bereinigung von Fehlern erfolgen muß.

Als direkte Konsequenz ergibt sich, daß eine vollständig korrekte Zuordnung derjenigen Instanzen, die sich auf dieselben Realwelt-Objekte beziehen, nicht immer erreichbar ist. Folglich wird eine Identifizierung eine *Korrektheit* aufweisen, die sich daran messen läßt (1) wie hoch der Anteil (oder die Wahrscheinlichkeit) nicht aufgedeckter Paare von Instanzen ist, die sich auf dieselben Realwelt-Objekte beziehen und (2) wie hoch der Anteil von Paaren von Instanzen ist, die fälschlicherweise einander zugeordnet wurden, obwohl sie sich auf verschiedene Realwelt-Objekte beziehen (den ersten Fehler bezeichnen wir als α-Fehler, während der zweite β-Fehler genannt wird; Details zur Schätzung dieser Fehler und weitere Korrektheitsmaße sind im Benchmarking-Kapitel ab Seite 141 zu finden).

Für die Identifikation in multiplen autonomen Datenbanken sind Identifizierungsansätze erforderlich, die bei vertretbarem Berechnungs-Aufwand eine hinreichende Korrektkeit der Identifizierung erreichen. Geeignet sind dabei Methoden, die für vorgelegte Paare von Instanzen eine Bewertung vornehmen, die schließlich in einer Klassifikation mündet. Die Klassifikation fällt für vorgelegte Paare eine Entscheidung, ob sie als sich auf dieselben Realwelt-Objekte beziehend erachtet werden oder nicht.

Wiewohl ein Identifizierungsansatz durch manuelles Erstellen der Entscheidungsregel von Experten gewonnen werden kann, ist dieses Vorgehen sehr aufwendig und nicht notwendigerweise von Erfolg gekrönt. Problematisch ist dabei zudem, daß eine Übertragung einer festgelegten Klassifikation auf andere Anwendungsfälle selten möglich ist.

Der in dieser Arbeit verfolgte Ansatz geht davon aus, daß es erfolgversprechender ist, die Entscheidungsregel aus Beispiel-Stichproben von Paaren von Instanzen zu lernen, für die jeweils bekannt ist, ob sie sich auf dieselben Realwelt-Objekte beziehen. Dazu lassen sich Verfahren aus der Statistik und dem Maschinellen Lernen verwenden.

Ausgehend von der Identifikation mit *identifizierenden Attributmengen* und *identifizierenden Referenz-Strukturen* (vgl. Seite 38) wählen wir diejenigen endlichen Teil-Strukturen der Referenz-Bäume von Datenbank-Objekten aus, die sich für die Identifizierung eignen, d.h. die Eigenschaften und Beziehungen der Realwelt-Objekte beschreiben, die auch für deren Identifikation in der realen Welt geeignet sind und die aus der Repräsentation der Realwelt-Objekte in den verschiedenen Datenbanken zu rekonstruieren sind.

Diese Teil-Strukturen sind Kandidaten für identifizierende Referenz-Strukturen, für ihre Werte können benutzerdefinierte Vergleichsfunktionen angewandt werden (wie z.B. die Minimum-Edit-Distanz). Basierend auf den (mehrdimensionalen) Vergleichswerten der Paare einer Beispiel-Stichprobe kann eine Klassifikation gelernt werden. Die Fähigkeit von Klassifikationsverfahren, unterschiedliche Klassen zu separieren, führt dazu, daß für Paare mit verschiedenen Vergleichswerten der Teil-Strukturen unterschiedliche Klassifikatoren gebildet werden. Diese Klassifikatoren wählen aus den Kandidaten der verwandten Teil-Strukturen je Vergleichswert-Kombination die jeweils am Besten für die Identifizierung in der Beispiel-Stichprobe geeignete Teil-Struktur aus.

Kapitel 2

Das Modell für die Identifizierung von Realwelt-Objekten

Heute back ich, morgen brau ich,
übermorgen hol ich der Königin ihr Kind;

ach, wie gut ist, daß niemand weiß,
daß ich Rumpelstilzchen heiß!

Aus „Rumpelstilzchen“

Inhalt dieses Kapitels. In diesem Kapitel wird das Modell für die Objektidentifizierung vorgestellt. Zwei Datenbanken A und B enthalten Informationen über Teile eines Universums von gleichartigen Realwelt-Objekten O. Zunächst werden die lokalen Schemata von A und B in ein globales Schema eingebettet. Die in A und B vorhandenen Informationen sind dann in geeigneten Attributen des globalen Schemas abgelegt. Dann werden die sowohl aus A als auch aus B ableitbaren Attribute gewonnen, die für die Identifizierung verwendbar sind. Für diese abgeleiteten Attribute werden Vergleichsfunktionen für Paare von Elementen aus A, B definiert, die entweder Abstandsmaße oder auch kategorielle Fallunterscheidungen sein können. Der Behandlung von Sonderfällen und NULL-Werten kommt beim Vergleich zweier Elemente eine besondere Bedeutung zu. Die eigentliche Identifizierung wird dann mittels einer Klassifikation auf den Vergleichswerten umgesetzt. Je nach Art der gewählten Vergleichsfunktionen lassen sich unterschiedliche Klassifikationsverfahren für die Identifikation einsetzen. Abschließend wird die Software-Architektur vorgestellt, die auf der Mediator-Wrapper-Architektur für die Informations-Integration basiert.

Wir betrachten zwei Datenbanken A und B, die Daten über Universen $O_\imath$, $\imath \in \mathbb{N}_+$ von jeweils gleichartigen Realwelt-Objekten enthalten, wie z.B. Personen, Adressen und Kraftfahrzeuge.

Für die Identifikation der Realwelt-Objekte, über die in A und B Daten vorhanden sind, ist die Annahme eines gemeinsamen globalen Schemas S

vorteilhaft. Mit einer (zumeist nur partiellen) Einbettung der Schemata S^A und S^B in das globale Schema S gewinnen wir identifizierende Attributmengen und Referenz-Strukturen, die beide Datenbanken gemeinsam aufweisen. Diese können wir dann für die Objektidentifizierung heranziehen.

2.1 Das globale Schema

2.1.1 Motivation

Im Kapitel 1 haben wir uns mit der Identifikation von Datenbank-Objekten auseinandergesetzt. In diesem Kapitel wird dargestellt, wie eine Identifizierung von Realwelt-Objekten, die als unterschiedliche Datenbank-Objekte repräsentiert sein können, umgesetzt werden kann.

Seien $S^A = \{C_1^A, \ldots, C_{n_A}^A\}$, $S^B = \{C_1^B, \ldots, C_{n_B}^B\}$ die Schemata zweier Datenbanken A, B, die je Klasse $C_i^A \in S^A$ und $C_j^B \in S^B$ Daten zu Realwelt-Objekten eines Universums $O_\imath$, $\imath \in \{1, \ldots, m\}$ mit $m < |S^A| + |S^B|$ enthalten.

Beispiel 2.1. Für die Daten in einem Melderegister A könnten das zum Beispiel die Universen $O_1 = \{\text{Personen}\}$, $O_2 = \{\text{Wohnungen}\}$, $O_3 = \{\text{Adressen}\}$, $O_4 = \{\text{Personaldokumente}\}$ und $O_5 = \{\text{Kraftfahrzeuge}\}$ sein, während in einer Sozialversicherungsdatei B Daten über die Universen O_1, O_3 und weitere Universen wie $O_6 = \{\text{Arbeitgeber}\}$ und $O_7 = \{\text{Krankenkassen}\}$ enthalten sind. Gemeinsam sind beiden Datenbanken A und B die Universen O_1 und O_3; Daten von Instanzen, die sich auf Realwelt-Objekte der nicht gemeinsamen Universen O_2, $O_4, \ldots, O_7$ beziehen, lassen sich nur insofern zur Identifizierung von Realwelt-Objekten in beiden Datenbanken heranziehen, als sie Aufschluß über die gemeinsamen Universen O_1 und O_3 geben können.

Seien also $C_i^A \in S^A$ und $C_j^B \in S^B$ zwei Klassen mit Typen $\tau_{C_i^A}, \tau_{C_j^B}$, die die elektronische Repräsentation von Realwelt-Objekten eines Universums $O_\imath$ darstellen. $X^{C_i^A}$, $X^{C_j^B}$ bezeichne die Attributmengen von C_i^A und C_j^B mit den Wertebereichen $\text{dom}(X^{C_i^A}) = \tau_{C_i^A}$, $\text{dom}(X^{C_j^B}) = \tau_{C_j^B}$. Um zu ermitteln, ob sich zwei Instanzen $a :: C_i^A$ und $b :: C_j^B$ auf ein und dasselbe Realwelt-Objekt beziehen, sind die Attributwerte $X^{C_i^A}(a)$ und $X^{C_j^B}(b)$ auf Übereinstimmungen zu untersuchen. Wir betrachten zunächst nur die Wert-Typen von $\tau_{C_i^A}$ und $\tau_{C_j^B}$. Es sind zwei Fälle zu unterscheiden:

1. Die Klassen C_i^A und C_j^B sind äquivalent, d.h. sie sind Typ-gleich $\tau_\imath = \tau_{C_i^A} = \tau_{C_j^B}$ und sie sind semantisch äquivalent, d.h. es gilt die Wert-Gleichheit $X^{C_i^A}(a) =_{\tau_\imath} X^{C_j^B}(b)$ für gleichzeitig zu erzeugende Instanzen $a :: C_i^A$ und $b :: C_j^B$ mit den Daten eines Realwelt-Objektes $o \in O_\imath$,[1]

[1] $X^{C_i^A}(a) =_\tau X^{C_j^B}(b)$ bezeichnet die Wert-Gleichheit aller Wert-Typen von τ, Label und Referenzen auf andere Datenbank-Objekte können verschieden sein, vgl. Definition 1.13 auf Seite 25.

2. Die Klassen C_i^A und C_j^B sind nicht äquivalent, d.h. sie sind nicht semantisch äquivalent oder nicht Typ-gleich[2]

Im 1. Fall können wir die Attributmengen $X^{C_i^A}, X^{C_j^B}$ eineindeutig aufeinander abbilden und für jeden Komponenten-Typ des gemeinsamen Typs τ Vergleiche der Attributwerte durchführen. Basierend auf den auftretenden Vergleichswerten kann eine Entscheidungsregel für Paare von Instanzen aufgestellt werden, die Paare beispielsweise als Duplikate oder Nicht-Duplikate klassifiziert.

Im 2. Fall, wenn keine Typgleichheit vorliegt, ist zunächst kein direkter Vergleich der Komponenten-Typen von $\tau_{C_i^A}$ oder $\tau_{C_j^B}$ möglich. Es sind (partielle) Einbettungen

$$h_\imath^A : \tau_{C_i^A} \to \tau_\imath,\ h_\imath^B : \tau_{C_j^B} \to \tau_\imath$$

der Komponenten-Typen in die Komponenten-Typen $\tau_\imath$ eines gemeinsamen, globalen Typs τ notwendig, um Vergleiche durchführen zu können. Für den gemeinsamen Typ $\tau_\imath$ muß die semantische Äquivalenz der eingebetteten Daten eines Realwelt-Objektes erfüllt sein, d.h. wir fordern die Wert-Gleichheit

$$h_\imath^A(X^{C_i^A}(a)) =_\tau h_\imath^B(X^{C_j^B}(b)) \tag{2.1}$$

für gleichzeitig zu erzeugende Instanzen $a :: C_i^A$ und $b :: C_j^B$ mit den Daten eines Realwelt-Objektes $o \in O_\imath$. Dann können Vergleiche der Attributwerte wie im 1. Fall realisiert werden.

Außerdem ist die Referenzierung auf andere Datenbank-Objekte der Typen $\tau_{C_i^A}$, $\tau_{C_j^B}$ bzw. des gemeinsamen Typs $\tau_\imath$ auf Korrektheit zu prüfen, d.h. es ist zu prüfen, ob Werte von Komponenten-Typen der Form $\ell : \tau$ Referenzen zu Datenbank-Objekten haben, die sich jeweils auf dasselbe Realwelt-Objekt beziehen. Dazu müssen wir wiederum Datenbank-Objekte erzeugen, auf die von zwei Instanzen $a :: C_i^A$ und $b :: C_j^B$ mit den Daten eines Realwelt-Objektes $o \in O_\imath$ referenziert werden kann. Sofern Datenbank-Objekte referenziert werden, die Instanzen anderer Klassen $C_{i'}^A, C_{j'}^B$ sind (mit $i' \neq i, j' \neq j$), ist zunächst eine (partielle) Einbettung in einen gemeinsamen Typ $\tau_{\imath'}$ zu finden, der semantisch äquivalent ist (wie für C_i^A, C_j^B, siehe oben). Offensichtlich kann eine Einbettung zweier Typen nur dann existieren, wenn sich Instanzen beider Klassen $C_{i'}^A, C_{j'}^B$ auf Realwelt-Objekte *eines* Universums $O_{\imath'}$ beziehen.

Letztlich konstruiert man durch dieses Vorgehen sukzessiv ein *gemeinsames, globales Schema S* für diejenigen Universen, über die in Instanzen der Klassen sowohl der einen als auch der anderen Datenbank Daten vorhanden sind (die Klassen der Universen, für die nur in einer der Datenbanken Daten vorhanden sind, können zusätzlich mit in das globale Schema aufgenommen werden).

[2]Nicht Typ-gleich heißt, daß es Komponenten-Typen von $\tau_{C_i^A}$ oder $\tau_{C_j^B}$ gibt, die nicht als Komponenten-Typ im jeweilig anderen Typ enthalten sind.

2.1.2 Die Annahmen für das globale Schema

Um mehrere Datenbanken zusammenführen zu können, müssen wir eine grundlegende Annahme treffen. Diese Annahme ist die notwendige Voraussetzung, um eine Identifikation von Realwelt-Objekten zu ermöglichen. In der Annahme 2.2 wird Kompatibilität zu einem globalen Schema gefordert, die durch die in Abschnitt 2.1.1 skizzierten Einbettungen zu bewerkstelligen sind. Im Anschluß diskutieren wir die Besonderheiten solcher Einbettungen.

Annahme 2.2 (Globales Schema). Seien A und B Datenbanken, die Daten über Realwelt-Objekte zumindest eines gemeinsamen Universums enthalten.

Wir nehmen an, daß es *ein globales Schema* $S = \{C_1, \ldots, C_n\}$ gibt, so daß die in beiden Datenbanken enthaltenen Daten über Realwelt-Objekte je eines Universums jeweils als Instanzen einer Klasse C_i modelliert sind.

Um die Annahme 2.2 zu konkretisieren, führen wir die Begrifflichkeiten für Einbettungen ein. Wir betrachten zuerst die Einbettung einzelner Typen in einen gemeinsamen Typ.

Definition 2.3 (Einbettung eines Typs). Es seien S, S' Schemata und $C_i \in S$, $C_j \in S'$ zwei Klassen, die jeweils den Typen τ_i bzw. τ_j genügen. Instanzen der Klassen C_i und C_j enthalten Daten über Realwelt-Objekte eines Universums O. X^{C_i} und X^{C_j} bezeichne die zugehörigen Attributmengen mit den Wertebereichen $\mathrm{dom}(X^{C_i}) = \tau_i$, $\mathrm{dom}(X^{C_j}) = \tau_j$. τ sei der Typ einer Klasse C, deren Instanzen ebenfalls Daten über Realwelt-Objekte eines Universums O enthalten.
Eine injektive Abbildung $h : \tau_j \to \tau$ bezeichnen wir als *Einbettung* des Typs τ_j in den Typ τ.[3]
Eine Einbettung h ist *vollständig* bzw. *vollständig Typstruktur-erhaltend*, wenn jeder Komponenten-Typ $\tau_{j,k}$ von τ_j auf einen Komponenten-Typ $\tau_\imath$ von τ abgebildet wird.
Eine Einbettung h ist *partiell*, wenn ein Komponenten-Typ $\tau_{j,k}$ von τ_j existiert, der auf keinen Komponenten-Typ $\tau_\imath$ von τ abgebildet wird.

Definition 2.4 (Wert-erhaltende Einbettungen). Zwei Einbettungen $h_1 : \tau_i \to \tau$, $h_2 : \tau_j \to \tau$ sind *Wert-erhaltend*, wenn die Wert-Gleichheit

$$h_1(X^{C_i}(a)) =_\tau h_2(X^{C_j}(b))$$

für aus den Daten eines Realwelt-Objektes $o \in O$ zu einem Zeitpunkt t erzeugte Instanzen $a :: C_i^A$ und $b :: C_j^B$ gilt.

Aus der Einbettung von einzelnen Typen zweier Schemata in übereinstimmende Ziel-Typen können wir die Einbettung der Schemata gewinnen, indem wir alle übereinstimmenden Ziel-Typen τ in einem globalen Schema zusammenfassen. Dadurch wird es möglich, Einbettungen zu erweitern auf Referenzen zwischen Instanzen von Klassen eingebetteter Typen (Referenzen auf Instanzen nicht eingebetteter Typen können trivialerweise nicht in S modelliert werden).

[3] Injektivität einer Typ-Transformation h heißt, daß jeder Komponenten-Typ eines Ziel-Typs τ unter h höchstens einen Komponenten-Typ des Quell-Typs τ_j als Urbild haben kann.

Definition 2.5 (Einbettung zweier Schemata). Es seien S^A, S^B Schemata und $h^A := (h_1^A, \ldots, h_n^A)$ und $h^B := (h_1^B, \ldots, h_n^B)$ die jeweiligen Einbettungen von Typen aus S^A und S^B in die Typen τ_i eines Schemas $S = \{C_1, \ldots, C_n\}$ für Instanzen von Realwelt-Objekten der Universen $O_1, \ldots, O_n$. Dann bezeichnen wir die Schemata S^A, S^B als in das globale Schema S eingebettet.
Eine Einbettung ist *redundanzfrei*, wenn jeder Komponenten-Typ des Typs jeder Klasse in S unter h^A höchstens einen Komponenten-Typ des Typs einer Klasse aus S^A als Urbild hat.

Definition 2.6 (Intensionale Überlappung zweier Einbettungen). Die *Intensionale Überlappung* zweier Einbettungen h^A, h^B der Schemata S^A und S^B in S ist gegeben durch die Anzahl der atomaren Wert-Typen τ von Klassen in S, für die sowohl in S^A als auch in S^B Komponenten-Typen existieren, d.h. daß diese Komponenten-Typen Urbilder von τ unter h^A bzw. h^B sind.
Atomare Wert-Typen bezeichnen Komponenten-Typen der Form $[\nu]$, $\{\nu\}$ und ν, für beliebige Basis-Datentypen ν.

Die intensionale Überlappung entspricht der Anzahl von Attributen, die bei der Einbettung aus beiden Schemata gewonnen werden.

Da ein Schema aus endlichen Klassen besteht, die nur endlich viele atomare Wert-Typen enthalten, gibt es für zwei Schemata immer eine maximale intensionale Überlappung. Sie ist bestimmt durch:

Definition 2.7 (Intensionale Überlappung zweier Schemata). Die *Intensionale Überlappung* zweier Schemata S^A und S^B ist das Maximum der intensionalen Überlappung der Wert-erhaltenden redundanzfreien Einbettungen in beliebige Schemata S.

Definition 2.8 (Referenz-erhaltende Einbettung). Die Einbettung eines Schemas S^A in S bezeichnen wir als *Referenz-erhaltend*, wenn es zu jeder Referenz zwischen Instanzen zweier in S eingebetteten Klassen C_i^A und C_j^A aus S^A eine Referenz zwischen Instanzen der entsprechenden Klassen in S gibt.

Definition 2.9 (Referenz-Äquivalenz zweier Einbettungen). Die Schemata S^A, S^B seien Referenz-erhaltend eingebettet in ein globales Schema S mit den Einbettungen h^A, h^B.

Dann bezeichnen wir die Einbettungen beider Schemata als *Referenz-äquivalent*, wenn Referenzen zwischen Instanzen von Klassen aus S in Datenbanken über beiden Schemata S^A und S^B repräsentierbar sind.

Die Einbettungen beider Schemata sind *maximal Referenz-äquivalent*, wenn alle Referenzen, die zwischen Instanzen eingebetteter Klassen in S^A sowie in S^B existieren, ebenfalls in S enthalten sind.

Basierend auf den Definitionen können wir zwei weitere Annahmen formulieren:

Annahme 2.10 (Maximale Wert-erhaltende Einbettung). Wir nehmen an, daß die zwei Schemata S^A und S^B Wert-erhaltend in ein globales Schema S eingebettet sind. Weiter nehmen wir an, daß die *intensionale Überlappung* beider Einbettungen maximal ist.

Mit der Annahme 2.10 stellen wir sicher, daß bei der Einbettung keine *in beiden Datenbanken vorhandenen* Informationen verlorengehen, d.h. daß alle in Instanzen der in beiden Datenbanken gemeinsam enthaltenen Daten in das globale Schema eingebettet werden. Dadurch gilt die Wert-basierte-Identifizierbarkeit im globalen Schema, sofern sie für die in S eingebetteten Attribute in den Datenbanken über den Schemata S^A und S^B gilt (siehe Formel (1.15) auf Seite 38).

Werden Referenzen für die Identifikation in Datenbanken über den Schemata S^A und S^B verwandt (wie z.B. in (1.16) und (1.17) auf Seite 38 oder bei Verwendung einer identifizierenden Referenz-Struktur, (1.20) auf Seite 39), müssen wir zusätzlich zur Annahme 2.10 die Äquivalenz der Referenzen zwischen Datenbank-Objekten fordern.

Annahme 2.11 (Referenz-äquivalente Einbettung). Wir nehmen an, daß beide Schemata S^A und S^B maximal Referenz-äquivalent eingebettet sind in ein globales Schema S.

2.2 Das formale Modell: Dreistufige Identifizierung

Wir nehmen gemäß der Annahmen 2.2, 2.10 (und 2.11, falls Referenzen zwischen Datenbank-Objekten zur Identifizierung herangezogen werden sollen) an, daß sich die in den Datenbanken A und B verfügbaren identifizierenden Informationen über die betrachteten Realwelt-Objekte in einem globalen Schema S beschreiben lassen und daß diese Einbettungen maximal Wert- und Referenz-erhaltend sind. Die lokalen Schemata seien also in das globale Schema eingebettet, die in einer Datenbank enthaltenen identifizierenden Informationen sind in geeignete Attribute des globalen Schemas transformiert. Aus diesen Attributen des globalen Schemas nehmen wir dann die Ableitung identifizierender Attributmengen und Referenz-Strukturen vor.

Bemerkung 2.12 *(Datenbereinigung).* Die Ableitung identifizierender Attributmengen und Referenz-Strukturen umfaßt auch die Bereinigung der Daten, insbesondere die Standardisierung von Zahlen- und Datumsformaten und Vereinheitlichung von Schreibweisen betreffend. Wir halten aber in jedem Fall die originären Daten vor, da in diesen auch Daten enthalten sein können, die nicht standardisierbar sind, aber trotzdem für die Objektidentifizierung relevant sein können (z.B. Adreß- oder Namens-Zusätze).

Die in diesem Abschnitt eingeführte Aufteilung der Objektidentifizierung in die drei Schritte *Conversion*, *Comparison* und *Classification* wurde von Neiling und Lenz eingeführt und anhand von Beispielen diskutiert (vgl. [Nei99, NL00b, NL02, NL00a]).

Definition 2.13 (Meta-Daten und Kontext-Information). Als *Meta-Daten* $\mathcal{I}_M$ für eine Datenbank A bezeichnen wir jegliche Daten über die Datenbank, die Aufschluß geben über die Bedeutung von Attributen, Korrektheit von Attributen, Zusammenhänge und funktionale Abhängigkeiten zwischen Attributmengen, Schemainformationen wie Struktur und Typen

u.v.a.m.. Desweiteren schließen wir auch die nicht Daten-spezifischen Eigenschaften der Datenbanken ein, z.B. die Art der Abfrageschnittstelle, Performanz, oder Kosten.

Die *Kontext-Information* $\mathcal{I}_C$ umfaßt weitere Daten und deren Meta-Daten, die die in A enthaltenen Daten ergänzen oder korrigieren können und *unabhängig* von der Datenbank A verfügbar gemacht werden können. Typische Beispiele sind Thesauri, Wörterbücher und Taxonomien, aber auch Daten aus anderen Datenbanken, wie z.B. Straßen- oder Namens-Verzeichnisse. Wir unterscheiden im folgenden nicht explizit Meta-Daten und Kontext-Information, die *zusätzliche Information* $\mathcal{I}$ umfaßt neben der Kontext-Information $\mathcal{I}_C$ ebenfalls die Meta-Daten $\mathcal{I}_M$, $\mathcal{I} := \mathcal{I}_M \cup \mathcal{I}_C$. In der englischsprachigen Literatur wird die *zusätzliche, externe Information* als *auxilary information* bezeichnet.

Die Ableitbarkeit eines Attributes besagt, daß die Werte dieses Attributes aus den Werten anderer Attribute gewonnen werden können.

Definition 2.14 (Ableitbarkeit).

(i) Eine Attributmenge Y heißt ableitbar aus einer Attributmenge X (*engl.: derivable*), wenn eine **surjektive Transformationsabbildung** $h_X : \mathrm{dom}(X) \to \mathrm{dom}(Y)$ existiert (Schreibweise $X \vdash Y$).

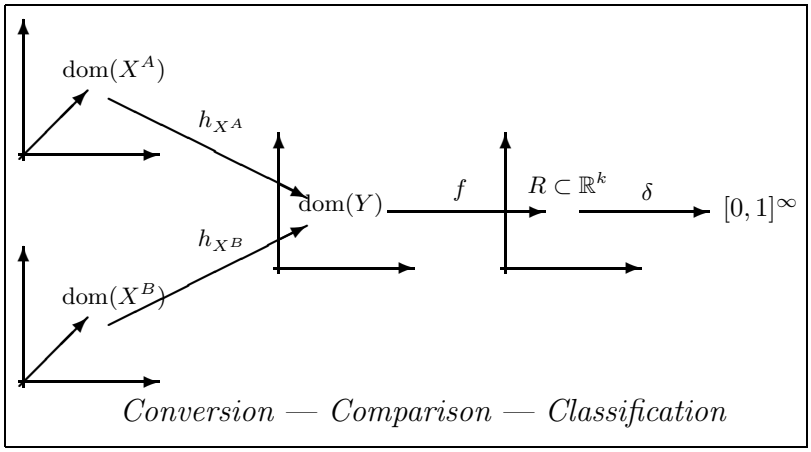
(ii) Eine Attributmenge Y heißt ableitbar aus einer Attributmenge X unter zusätzlicher (Kontext-) Information $\mathcal{I}$, wenn eine **surjektive Transformationsabbildung** $h_X(\cdot;\mathcal{I}) : \mathrm{dom}(X) \to \mathrm{dom}(Y)$ existiert (Schreibweise $(X;\mathcal{I}) \vdash Y$).

Die Forderung nach der Surjektivität der Transformationsfunktion stellt sicher, daß jede mögliche Wert-Kombination von Y in $\mathrm{dom}(Y)$ auch ein Urbild in $\mathrm{dom}(X)$ hat. Injektivität können wir von dieser Funktion jedoch nicht voraussetzen, da mehrere Werte von X auf einen Wert von Y abgebildet werden können, wie z.B. bei phonetischen Kodierungen von Namen.

Die ableitbaren Attribute sind letztlich der Schlüssel der Identifikation: Die Informationen, die in den vorhandenen Attributen erfaßt sind, lassen sich oft nicht direkt zur Identifikation einsetzen, sondern müssen standardisiert werden (z.B. geographische Bezeichnungen, Telefonnummern mit Vorwahlen, Namenszusätze etc.) oder mit Hilfe zusätzlicher Kontext-Information $\mathcal{I}$ (wie z.B. Straßen- und Namensverzeichnisse, Thesauri für Abkürzungen) hergeleitet werden. Letzteres soll an einem Beispiel verdeutlicht werden. Weitergehende Ausführungen zur Ableitbarkeit werden im Abschnitt 2.3 auf Seite 52 gegeben.

Beispiel 2.15. Für Personen ist das Alter *age* aus dem Geburtsdatum b ableitbar, wenn das aktuelle Datum (*today*) als zusätzliche Information herangezogen wird. Die Transformationsabbildung $h : DATE \to \{0, 1, 2, \ldots\}$, $age := h(b; today)$ ergibt sich dann wie folgt (b bezeichne das Geburtsdatum):

$$h(b; today) := \begin{cases} \texttt{YEAR}(today) - \texttt{YEAR}(b) : & \texttt{MONTH}(today) > \texttt{MONTH}(b) \\ \texttt{YEAR}(today) - \texttt{YEAR}(b) : & \texttt{MONTH}(today) = \texttt{MONTH}(b) \\ & \texttt{and DAY}(today) \geq \texttt{DAY}(b) \\ \texttt{YEAR}(today) - \texttt{YEAR}(b) - 1 : & \text{otherwise} \end{cases}$$



Legende:		
dom(X), dom(X^C).	Wertebereich einer Attributmenge	S.21
dom(Y)	Wertebereich der ableitbaren Attribute Y	S.49
$R \subset \mathbb{R}^k$	Wertebereich der Vergleichsfunktion f	S.50
$I \subset \mathbb{N}$	Index einer Klassifikation $\mathcal{K}$ auf R	S.50
h_{X^A}, h_{X^B}	Transformationsfunktionen für A, B	S.49
f	Vergleichsfunktion	S.50
δ	Endscheidungsregel einer Klassifikation	S.50

Abbildung 2.1: Dreistufige Identifizierung mit Klassifikation.

Die aus beiden Datenbanken A, B respektive ihrer Attributmengen X^A, X^B ableitbaren Attribute können wir paarweise vergleichen. Dazu führen wir Vergleichsfunktionen ein:

Definition 2.16 (Vergleichsfunktion). Y sei eine Attributmenge mit $|Y| = d$. R sei eine k-dimensionale Menge, $0 < k \leq d$. Dann bezeichnen wir eine Funktion $f : (\text{dom}(Y))^2 \rightarrow R$ als *Vergleichsfunktion* (*engl.: comparison function*).
R bezeichnen wir als Vergleichsraum (*engl.: comparison space*).

Für die Darstellung eines k-dimensionalen Vergleichswertes $r \in R$ können wir einen k-dimensionalen Vektor $r = (r_1, \ldots, r_k)^T$ mit $r_i \in \mathbb{N}$ oder $r_i \in \mathbb{R}$ verwenden. Dieser Vektor ist im Allgemeinen nicht als Punkt des $\mathbb{R}^k$ auffaßbar, da in der Regel keine Metrik angegeben werden kann, die einen sinnvollen Abstand zwischen den mehrdimensionalen Vergleichswerten mißt. Weitere Details zu Vergleichsfunktionen sind im Abschnitt 2.4 auf Seite 59 zu finden.

Definition 2.17 (Klassifikation). Eine Klassifikation $\mathcal{K} = \{\mathcal{K}_1, \ldots, \mathcal{K}_j, \ldots\}$ auf einer Menge M ist gegeben durch eine Entscheidungsregel (*engl.: decision rule*) $\delta : M \rightarrow [0,1]^\infty$, die jedem Element $c \in M$ eine Folge (bzw. einen unendlich-dimensionalen Vektor) $\delta(c) = (p_1(c), \ldots, p_j(c), \ldots)^T$ zuordnet, wobei für alle $c \in M$

$$\sum_{j=1}^{\infty} p_j(c) = 1. \tag{2.2}$$

erfüllt sei.

Der Wert der $p_j(c) \in [0,1]$ ist ein Maß dafür, wie stark ein Punkt $c \in M$ einer Klasse $\mathcal{K}_j \in \mathcal{K}$ zugehörig ist. Ein Punkt c ist sozusagen auf die Klassen der Klassifikation verteilt. $p_j(c) = 0$ bedeutet, daß c nicht in die Klasse $\mathcal{K}_j$ gehört, und $p_j(c) = 1$ heißt wegen (2.2), daß c ausschließlich in die Klasse $\mathcal{K}_j$ fällt. $0 < p_i(c) < p_j(c) < 1$ heißt schließlich, daß c in die Klassen $\mathcal{K}_i$ und $\mathcal{K}_j$ fällt, wobei c stärker in $\mathcal{K}_j$ als in $\mathcal{K}_i$ fällt.

Wir geben diese allgemeine Definition 2.17 an, damit verschiedene Klassifikationsmethoden einsetzbar sind. Als Interpretation der p_j sind zum Beispiel —je nach Art der verwandten Methode— Wahrscheinlichkeiten einer Bayes-Klassifikation oder Zugehörigkeitswerte einer Fuzzy-basierten Klassifikation möglich.

Wir erhalten folgende Spezialfälle:

- **Endliche Klassifikation:** $\mathcal{K}$ sei eine Klassifikation auf einer Menge M. $\mathcal{K}$ heißt endlich, wenn eine Zahl $max \in \mathbb{N}$ existiert, so daß gilt:

$$\forall c \in M \, \forall i > max : \; p_i(c) = 0. \tag{2.3}$$

  Die Entscheidungsregel $\delta : M \to [0,1]^\infty$ reduziert sich hier auf $\delta : M \to [0,1]^{max}$.

- **Disjunkte Klassifikation:** $\mathcal{K}$ sei eine Klassifikation auf einer Menge M. $\mathcal{K}$ ist eine disjunkte Klassifikation auf M, wenn jedes $c \in M$ in genau eine Klasse $\mathcal{K}_j$ fällt, d.h. wenn für alle $c \in M$ gilt:

$$\exists j \in \mathbb{N} : \quad p_j(c) = 1. \tag{2.4}$$

  In diesem Fall läßt sich die Entscheidungsregel vereinfachen: $\delta : M \to \mathbb{N}$ ordnet jedem Element $c \in M$ den Index $j = \delta(c)$ einer Klasse $\mathcal{K}_j \in \mathcal{K}$ zu.

Es folgt unmittelbar

Satz 2.18. *Es seien h_A, h_B Transformationsfunktionen, die die Attributmengen X^A, X^B zweier Datenbanken A und B in die gemeinsam ableitbare Attributmenge Y transformieren und $f : (\mathrm{dom}(Y))^2 \to R$ eine Vergleichsfunktion.*

Dann induziert jede Klassifikation auf R eine Klassifikation auf $A \times B$.

Beweis. Der Satz folgt mit Definition 2.17 der Klassifikation direkt aus den Definitionen 2.14 und 2.16; vgl. Abbildung 2.1. □

Die Abbildung 2.1 veranschaulicht den dreistufigen Prozeß der Identifizierung von Realwelt-Objekten und zeigt die verwandte Notation. Für die Klassifikation läßt sich sowohl überwachtes als auch nicht-überwachtes Lernen einsetzen. Der Diskussion von Klassifikationsverfahren ist ein eigenes Kapitel ab S. 69 gewidmet.

Bemerkung 2.19 *(Vorauswahl von Paaren).* Wenn eine Klassifikation $\mathcal{K}$ für Paare von Elementen einer Datenbank bekannt ist, müßte man alle Paare bilden und darauf die Klassifikation $\mathcal{K}$ anwenden. Von diesem Vorgehen ist

aus Effizienzgründen jedoch abzuraten, da (im Falle der Duplikat-Erkennung in einer Datenbank mit n Elementen) $\frac{n(n-1)}{2}$ verschiedene Paare zu bilden wären.[4] Da für die meisten Paare von vornherein klar ist, daß sie keine Duplikate sein können, muß man diese auch nicht bilden. Dies führt zur Idee der *Vorauswahl von Paaren* (*engl.: preselection*), die im 5. Kapitel untersucht wird.

2.3 Ableitbare Attribute - Der Schlüssel zur Identifizierung

Hier werden die *ableitbaren Attribute* und ihre Rolle im Prozeß der Objektidentifizierung ausführlich diskutiert. Die *ableitbaren Attribute* wurden mit der Definition 2.14 im Abschnitt 2.2 auf Seite 48 eingeführt.

Eine Transformationsabbildung $h_X(\cdot;\mathcal{I}) : \mathrm{dom}(X) \rightarrow \mathrm{dom}(Y)$ gemäß Definition 2.14 auf Seite 49 berechnet aus einem Wert $x \in \mathrm{dom}(X)$ einen Wert $y \in \mathrm{dom}(Y)$, möglicherweise unter Zuhilfenahme zusätzlicher Information $\mathcal{I}$, die wir als Parameter der Funktion $h_X(\cdot;\mathcal{I})$ auffassen. Wir setzen voraus, daß die Berechnung von Funktionswerten durch einen Algorithmus erfolgen kann. Solch ein Algorithmus muß folgende Eigenschaften erfüllen:

- **Deterministisch:** Bei identischer Information $\mathcal{I}$ wird für einen Wert $x \in \mathrm{dom}(X)$ immer derselbe Wert $y \in \mathrm{dom}(Y)$ berechnet,
- **Terminierend (Berechenbarkeit):** In endlicher Zeit wird ein Ergebnis zurückgegeben.

Beispiel 2.20 *(Standardisierung von Telefonnummern).* Wir betrachten eine Datenbank mit Immobilienanzeigen aus Berliner Tageszeitungen, die ein Attribut Telefonnummer enthalte. Die Telefonnummern enthalten oft Sonderzeichen: „-“, „/“, Leerzeichen und manchmal einen Zusatz wie „AB“, „Tel./Fax“, „Fax“ oder anderes wie „ab 8 Uhr“. In diesem Attribut sind außerdem manchmal Vorwahlen angegeben, bei Berliner Telefonnummern fehlt diese jedoch in der Regel. Wir können das standardisierte Attribut TelefonMitVorwahl, das ausschließlich die Ziffern mit Vorwahl enthält, mit dem Algorithmus `CleanPhoneNumberOne` in Abbildung 2.2 aus dem Attribut Telefonnummer ableiten.

Sofern außerdem ein Vorwahlverzeichnis zur Verfügung steht, lassen sich ein Attribut *Vorwahl* sowie ein Attribut *TelefonOhneVorwahl* aus dem abgeleiteten Attribut *Telefon* gewinnen. Zusätzlich läßt sich auch das Attribut *TelefonArt* mit dem Wertebereich $\mathrm{dom}((TelefonArt)) = \{Mobil, Festnetzanschluß\}$ anhand der angegebenen Vorwahl mit einer Liste der Vorwahlnummern für Mobilfunknetze als Kontext-Information ermitteln.

[4] Für zwei Datenbanken der Größe n_A, n_B wären offensichtlich

$$\frac{n_A(n_A-1)}{2} + n_A \cdot n_B + \frac{n_B(n_B-1)}{2}$$

Paare zu bilden, falls jede der Datenbanken selbst frei von Duplikaten ist, immerhin noch $n_A \cdot n_B$ Paare.

```
FUNC CleanPhoneNumberOne(IN: textPhoneNumber)

  /* INITIALIZATION */
  textPhoneNumberCleaned = EMPTY
  boolFirstDigit = TRUE

  FOR EACH char IN textPhoneNumber

    /* ONLY ADDING OF DIGITS - */
    /* IGNORES OTHER CHARS */
    IF IsDigit(char) THEN

      /* CHECK THE FIRST DIGIT ONCE */
      IF boolFirstDigit THEN
        boolFirstDigit = FALSE

        /* ADDING THE PRECALL "030" FOR BERLIN */
        /* IF NO PRECALL IS GIVEN  */
        IF NOT (char == "0") THEN
            textPhoneNumberCleaned = "030"
        END IF

      END IF

      /* ADDING THE DIGIT char */
      textPhoneNumberCleaned.CONCAT char

    END IF

  END FOR

  RETURN textPhoneNumberCleaned

END FUNC
```

Abbildung 2.2: Der Algorithmus `CleanPhoneNumberOne` aus Beispiel 2.20

Definition 2.21 (Verfeinerung von Transformationsfunktionen). Es sei X eine Attributmenge und Y_1, Y_2 seien mit h_1, h_2 aus X ableitbare Attribute, $h_i : \mathrm{dom}(X) \to \mathrm{dom}(Y_i), i = 1, 2.$ h_1 bezeichnen wir als Verfeinerung von h_2, Schreibweise $h_1 \succ h_2$, wenn gilt:

$$|\mathrm{dom}(Y_1)| \geq |\mathrm{dom}(Y_2)| \tag{2.5a}$$

$$\exists g : \mathrm{dom}(Y_1) \to \mathrm{dom}(Y_2) \ : \ h_2 = g \circ h_1 \tag{2.5b}$$

Umgekehrt bezeichnet man h_2 auch als Vergröberung von h_1.

Die Funktion g erfüllt die Aufgabe der Aggregation bzw. Gruppierung von Werten — jeder Wert in $\mathrm{dom}(Y_2)$ hat unter g einen oder mehrere Werte aus $\mathrm{dom}(Y_1)$ zum Urbild. Verfeinerungen von Transformationsfunktionen bieten die Möglichkeit, die Domäne eines Attributes sukzessiv zu verfeinern, ein immer engeres Gitter (oder Raster) auf die Attributwerte anzuwenden.

Satz 2.22 (Teilordnung). *Es sei X eine Attributmenge und $TF(X) = \{h : \mathrm{dom}(X) \to h(\mathrm{dom}(X))\}$ die Menge aller Transformationsfunktionen auf* $\mathrm{dom}(X)$.

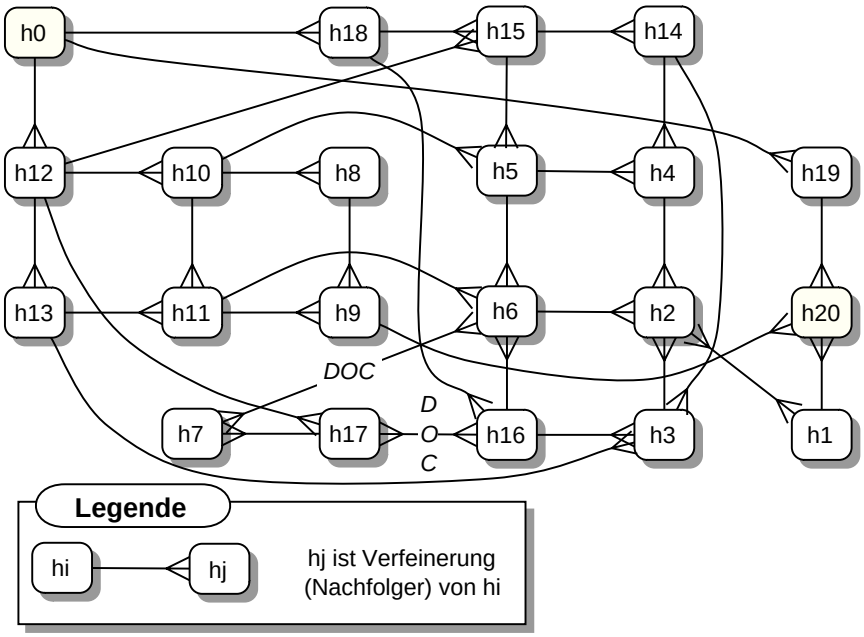


Abbildung 2.3: Die Verfeinerung von Transformationsfunktionen $h_j : \mathrm{dom}(X_i) \to \mathrm{dom}(Y_j)$ aus dem Beispiel 2.23. Die Abbildung zeigt die partielle Ordnung (Vorgänger–Nachfolger) auf $\{h_0, \ldots, h_{20}\} \subset TF(X_i)$.

Dann induziert $\prec$ eine partielle Ordnung auf $TF(X)$. für eine Attributmenge X. Diese Teilordnung besitzt ein minimales Element[5] $h_o : \mathrm{dom}(X) \to \{\perp\}, h_o(x) = \perp$.

Beweis. Sei $h \in TF(X)$ mit $h(\mathrm{dom}(X)) \neq \emptyset$ beliebig, aber fest. Offensichtlich ist dann (2.5a) wegen $|\{\perp\}| = 1 \leq |h(\mathrm{dom}(X))|$ erfüllt. Mit $g : h(\mathrm{dom}(X)) \to \{\perp\}, g(x) = \perp$ erhalten wir $h_o = g \circ h$ und damit (2.5a). h ist also eine Verfeinerung von h_o. □

Wir geben ein Beispiel für Verfeinerungen, aus dem sich ersehen läßt, daß Attribute, die auf den ersten Blick verschieden sind, sich durch Verfeinerung von Transformationsfunktionen ableiten lassen. Dergestalt erhält man eine partielle Ordnung der Transformationsfunktionen. Solch eine Ordnung läßt sich für die Gruppierung/Indexierung von Datensätzen bzw. Daten-Elementen ausnutzen, indem man zunächst nach den gröberen Attributen gruppiert und diese nach Bedarf weiter verfeinert. Dieses Vorgehen ermöglicht eine effiziente Verarbeitung großer Datenmengen unter Zuhilfenahme von geeigneten Indexstrukturen.

Beispiel 2.23. Es sei X_1 ein Attribut mit $\mathrm{dom}(X)_1 = \{\texttt{TEXT}\}$, mit Zeichenketten über einem Alphabet $\mathcal{A}$. Dieses Attribut könnte z.B. der Buchtitel in einem Buchkatalog sein. Mit u bezeichnen wir einen Datensatz aus *UR*, mit $X_1(u)$ den Wert des Attributes X_1 für u. Wir können aus X_1 mittels Transformationsfunktionen h_i folgende Attribute ableiten (in den Klammern ist eine kurze Variablenbedeutung zu finden):

[5] Genaugenommen existieren unendlich viele minimale Elemente $h_y : \mathrm{dom}(X) \to \{y\}, h_y(x) = y$ für beliebiges y, die aber f.a. $y \quad h_y \prec h_o \prec h_y$ erfüllen. Wir wählen eine dieser Funktionen, nämlich h_o als *das* minimale Element von $TF(X)$ aus.

- Y_o (NULL-Abbildung), welches durch $h_o : \mathrm{dom}(X)_1 \to \{\perp\}, h_o(x) = \perp$ gegeben ist, wenn $\perp$ das leere Wort bezeichne .

- Y_1 (Text mit Wörtern), welches alle in $X_1(u)$ enthaltenen Wörter enthalte,[6] die durch Freizeichen „ “ getrennt seien .

- Y_2 (Wortliste), welches die Liste aller in $X_1(u)$ enthaltenen Wörter enthalte.

- Y_3 (standardisierte Wortliste), ein aus Y_2 abgeleitetes standardisiertes Attribut, welches ebenfalls die Liste aller in $X_1(u)$ enthaltenen Wörter enthalte, jedoch sind alle Groß- in Kleinbuchstaben umgewandelt sowie Sonderzeichen (ä, ö, ü, ß) durch (ae, oe, ue, ss) ersetzt worden.

- Y_4 (Wortliste ohne Wiederholungen), wie Y_2, wobei mehrfach in $X_1(u)$ auftretende Wörter nur einmal in der Liste aufgeführt werden.

- Y_5 (Vektor für Wörter), welches aus einem Vektor $v \in \{0,1\}^n$ bestehe, der für jedes in $X_1(u)$ vorkommende Wort $v_j = 1$ und für jedes nicht vorkommende Wort $v_j = 0$ enthalte, wobei der Index $j \in \{1, \ldots, n\}$ die Position eines Wortes in einem Wortverzeichnis DIC der Länge n darstelle.

- Y_6 (Vektor für Worthäufigkeit), wie Y_5, nur daß in dem Vektor $v \in \mathbb{N}^n$ die j-te Komponente v_j die Häufigkeit $freq(w, X_1(u))$ des j-ten Wortes w des Wortverzeichnisses DIC im Attribut X_1 des Datensatzes u angebe.

- Y_7 (Vektor für IDF–TF-Wert von Wörtern), wie Y_6, nur daß für die in $X_1(u)$ enthaltenen Wörter w nicht die Häufigkeit $freq(w; X_1(u))$, sondern den reskalierten Wert ν enthält, der definiert ist durch

$$\nu := \big(1 + \log freq(w; X_1(u))\big) \cdot \log \frac{|DOC|}{freq(w; DOC)}$$

  wobei $freq(w; DOC)$ die Häufigkeit des Wortes w in einer Dokumentsammlung DOC bezeichne. Letztlich drückt die obige Formel, die als Term Frequency– Inverse Document Frequency Gewichtungsschema bekannt ist, aus, daß in der Dokumentsammlung seltene Wörter höher gewichtet werden als die häufigen Wörter. Wir haben hier $h_7(\cdot; DOC) : \mathrm{dom}(X)_1 \to \mathbb{R}^n_{\geq 0}$, d.h. die Dokumentsammlung DOC geht als Kontextinformation in die Ableitung mit ein.

- Y_8 (Vektor für Zeichen), welches aus einem Vektor $v \in \{0,1\}^{|\mathcal{A}|}$ bestehe, der für jedes in $X_1(u)$ vorkommende Zeichen $z \in \mathcal{A}$ eine 1 und sonst eine 0 in der j-ten Komponente enthalte, wobei der Index $j = 1, \ldots, |\mathcal{A}|$ mit der Position des Zeichens z in $\mathcal{A}$ ist.

- Y_9 (Vektor für Zeichenhäufigkeit), wie Y_8, nur daß die Häufigkeit der Zeichen in $X_1(u)$ in v angegeben wird, d.h. $v \in \mathbb{N}^{|\mathcal{A}|}$.

[6] Wörter sind hierbei die Zeichenketten in $X_1(u)$, die ausschließlich aus Buchstaben (a – z, A – Z, ä, ö, ü, ß, Ä, Ö, Ü) bestehen und durch andere Zeichen in $X_1(u)$ begrenzt werden. Mehrfach auftretende Wörter werden mehrfach in der Liste aufgeführt.

- Y_{10} (Vektor für Buchstaben), wie Y_8, jedoch nur für Buchstaben z in $\mathcal{A}$.
- Y_{11} (Vektor für Buchstabenhäufigkeit), wie Y_9, jedoch nur für Buchstaben z in $\mathcal{A}$.
- Y_{12}–Y_{17} (standardisierte Attribute): Anwendung der Standardisierung der Buchstaben ebenso wie in Y_3 (Ersetzung von Sonderzeichen), Zuordnung vgl. folgende Tabelle

Attribut	analog Attribut	kurze Beschreibung
Y_{12}	Y_{10}	Vektor für Buchstaben ($\in [0,1]^{\lvert\mathcal{A}\rvert}$)
Y_{13}	Y_{11}	Vektor für Buchstaben ($\in \mathbb{N}^{\lvert\mathcal{A}\rvert}$)
Y_{14}	Y_4	Wortliste ohne Wiederholungen
Y_{15}	Y_5	Vektor für Wörter ($\in [0,1]^n$)
Y_{16}	Y_6	Vektor für Worthäufigkeit ($\in \mathbb{N}^n$)
Y_{17}	Y_7	Vektor für TF–IDF-Wert ($\in \mathbb{R}^n$)

- Y_{18} (Textlänge), das die Länge des Textes (Anzahl der Zeichen in $X_1(u)$) angibt.
- Y_{19} (Sprache), das die Sprache des Textes angibt; falls sie nicht zu ermitteln ist, erhalten wir $Y_{19}(u) = \perp$.
- Y_{20} (Identität), das den Text unverändert übernimmt, h_{20} ist also die identische Abbildung.

Für die Transformationsfunktionen $h_o, \ldots, h_{20}$ gelten unter anderem folgende Verfeinerungen (die vollständige Teilordnung ist in der Abbildung 2.3 zu ersehen):

- $h_0 \prec h_i$ und $h_i \prec h_{20}$ für alle i,
- $h_3 \prec h_2$, $h_4 \prec h_2$,
- $h_5 \prec h_4$, es liegen unterschiedliche Repräsentationen der Werte vor. Wir haben jedoch $h_4 \not\prec h_5$, da die Reihenfolge der Wörter in der Liste eine Rolle spielt, und
- $h_6 \prec h_7$ und $h_7 \prec h_6$, wenn man die Dokumentsammlung *DOC* als Kontextinformation berücksichtigt.

2.3.1 Behandlung von NULL-Werten

Für die Untersuchung von Attributen spielen fehlende Werte —die als NULL-Werte bezeichnet werden— eine gewisse Rolle. Deshalb widmen wir der Behandlung von NULL-Werten unsere Aufmerksamkeit.[7]

Wir unterscheiden dabei grundsätzlich zwischen zwei Typen von NULL-Werten. Für das Realwelt-Objekt, für das in einem Attribut $X_i(a) = \perp$ einer Instanz $a \in A$ gilt, kann genau einer der folgenden beiden Fälle für

[7] Der Begriff NULL-Wert wurde in der Definition 1.17 auf Seite 27 eingeführt.

das Vorliegen des NULL-Wertes vorliegen (Die verwandten Bezeichnungen wurden von C.F. Codd eingeführt [Cod86]):

Entweder

- **APPLICABLE**—ANWENDBAR:
  Nicht-struktureller NULL-Wert. Die betreffende Eigenschaft existiert zwar für das entsprechende Realwelt-Objekt, es ist aber kein Wert für das Attribut vorhanden.[8]

oder

- **INAPPLICABLE**—NICHT ANWENDBAR:
  Struktureller NULL-Wert. Die betreffende Eigenschaft existiert nicht für das Realwelt-Objekt, auf den sich die Instanz a bezieht. Es kann kein (sinnvoller) Wert für das Attribut angegeben werden.[9]

Aufgrund der fehlenden Information kann es aber auch vorkommen, daß für einen NULL-Wert unklar ist, welcher der beiden Typen (APPLICABLE/INAPPLICABLE) vorliegt:[10]

- **NO INFORMATION**—KEINE INFORMATION:
  Es ist unklar, ob der NULL-Wert vom Typ APPLICABLE oder INAPPLICABLE ist. Es sind beide Typen möglich!

In der Literatur werden über 20 verschiedene Arten von NULL-Werten diskutiert, die jeweils einem der drei oben angegebenen Typen zugeordnet werden können.

Im Falle nicht-struktureller NULL-Werte (**APPLICABLE**) ist es manchmal möglich, einen Wert zu ermitteln. Dies kann beispielsweise durch das Ausnutzen funktionaler Abhängigkeiten für das betroffene Attribut geschehen.

Falls jedoch für ein struktureller NULL-Wert (**INAPPLICABLE**) vorliegt, darf kein Wert aus dem Wertebereich des Attributes dafür gesetzt werden.

Oft kommt es zu Überlagerungen beider Arten von NULL-Werten in einem Attribut einer Datenbank, ohne daß erkenntlich ist, welcher Typ vorliegt. Mit anderen Worten, es liegt der Typ **NO INFORMATION** vor. Dies sei an einem Beispiel erläutert:

Beispiel 2.24. Wir betrachten einen Bibliothekskatalog. Hier kann das Fehlen der ISBN-Nummer in einem Datensatz darauf hindeuten, daß entweder

a) keine ISBN-Nummer für diesen Eintrag existiert (z.B. bei Aufsätzen und älteren Büchern möglich) oder

[8] Eine alternative Bezeichnung für diesen Typ von NULL-Wert ist: **UNKNOWN**—UNBEKANNT

[9] Eine alternative Bezeichnung für diesen Typ von NULL-Wert ist: **DOES NOT EXIST**—EXISTIERT NICHT.

[10] Die Bezeichnung **NO INFORMATION** für diesen Typ von NULL-Wert wurde von Zaniolo eingeführt [Zan84, GZ88].

b) die ISBN-Nummer lediglich fehlt (z.B. bei unvollständiger Datenerfassung).

Sofern es sich um den Fall b) handelt, kann für andere Datensätze, die sich auf dasselbe Buch beziehen, eine ISBN-Nummer angegeben sein. Das ist im Fall a) jedoch nicht möglich, da es sich um einen strukturellen NULL-Wert handelt.

Für das Ersetzen nicht-struktureller NULL-Werte können verschiedene Strategien angewandt werden, die auch als *Imputationen* bezeichnet werden (siehe dazu beispielsweise die Arbeiten von Dempster, Laird und Rubin [DLR77], Vassiliou [Vas81], Little und Rubin [LR87] und Feelders [Fee99]). Wir gehen exemplarisch auf drei dieser Strategien näher ein, wobei wir uns an dieser Stelle auf Relationen beschränken.

1. **Ableitung aus anderen Attributen:** Unter Ausnutzung von (funktionalen) Abhängigkeiten bzw. Redundanzen läßt sich der fehlende Wert aus den verfügbaren Attributwerten des Datensatzes ermitteln, möglicherweise unter Verwendung zusätzlicher Informationen, wie z.B. die Ableitung der PLZ oder des Landes aus den Angaben für Ort und Straße/Hausnummer.

2. **Ersetzen durch den Wert eines anderen Datensatzes:** Der fehlende Wert von $X_i(a_0)$ wird mit dem Wert $x = X_i(a_l)$ eines anderen Datensatzes ersetzt, der in anderen (vorher festgelegten) Attributen X' mit dem aktuellen Datensatz übereinstimmt — sofern folgende Abhängigkeit gilt:

   $$\forall a : \big(X'(a_0) = X'(a) \wedge X_i(a) \neq \perp\big) \Longrightarrow X_i(a) = x$$

   Hier wird eine empirische funktionale Abhängigkeit für Attributwerte ausgenutzt, wobei die Korrektheit der Imputation stark von den Daten sowie der Wahl der Attributmenge X' abhängt.

3. **Mehrfache Ersetzung (*engl.: multiple imputation*):** Da oft nicht *ein Wert*, sondern mehrere Werte für einen NULL-Wert möglich sein können, kann anstatt eines Wertes auch eine Angabe mehrerer Werte (u.U. verbunden mit einer Bewertung) erfolgen. Für die Speicherung mehrerer Werte wird ein Mengen- oder Listen-Typ verwandt. Alternativ können mehrere Datensätze mit den unterschiedlichen Werten erzeugt werden, die jedoch als Duplikate in der Datenbank kenntlich zu machen sind. Dieses Vorgehen führt jedoch zu einer Erhöhung der Anzahl der Datensätze, die die Effizienz der Verarbeitung beeinträchtigen kann. Deshalb ist die erstgenannte Alternative im Regelfall vorzuziehen.

Bemerkung 2.25. Sofern eine Ersetzung eines NULL-Wertes nicht möglich ist, verbleibt die Möglichkeit, ihn als NULL-Wert weiterzugeben, um dann beim Vergleich von Datensätzen die NULL-Werte explizit zu berücksichtigen. Im Beispiel 2.24 wurden die Besonderheiten von fehlenden ISBN's beschrieben, eine Ersetzung ist hier nicht immer möglich, jedoch läßt sich der Fall

zweier aufeinandertreffender NULL-Werte bzw. eines NULL-Wertes mit einem Wert aus dem Wertebereich beim Vergleich separat behandeln. Insbesondere für strukturelle NULL-Werte ist dieses Herangehen sinnvoll, da das Fehlen eines Wertes eine identifizierende oder zumindest unterscheidende Information bergen kann. Zusätzliches Wissen über das Auftreten von NULL-Werten in Attributen kann durch Meta-Daten angegeben werden, beispielsweise mit semantischen Constraints, die im Kapitel 4.2 auf Seite 108 behandelt werden.

Beispiel 2.26. Bei der Auswertung einer anonymisierten medizinischen Datenbank mit Krankheitsbildern habe das Attribut *Anzahl_Schwangerschaften* in über 70% aller Datensätze keinen Wert. Die in den restlichen 30% angegebenen Werte sind überwiegend von 0 verschieden. Können wir den fehlenden Wert mit 0 ersetzen?
Eine Imputation eines Wertes ist generell nur dann zulässig, wenn man sicherstellen kann, daß es sich nicht um einen strukturellen NULL-Wert handelt. Für dieses Beispiel heißt das, $Anzahl\_Schwangerschaften$ darf nur dann mit einem Default-Wert (z.B. 0) ersetzt werden, wenn aus den anderen Attributwerten ersichtlich ist, daß es sich um eine Frau handelt. Für Männer kann das Attribut *Anzahl_Schwangerschaften* nicht sinnvoll gesetzt werden,[11] da ein struktureller NULL-Wert vorliegt.

Für weitergehende Informationen zu NULL-Werten und unvollständiger Information in Datenbanken sei auf die Arbeiten von Vassiliou [Vas79], Lipski [Lip79, Lip81], Biskup [Bis83], Imielinski [IL84], Paredaens [PBGG89], Grahne [Gra91], Candan et al. [CGS97] sowie Klein [Kle94, Kle97, Kle02] verwiesen.

2.4 Vergleichsfunktionen

In diesem Abschnitt wird auf die Definition und Auswahl von Vergleichsfunktionen eingegangen. Vergleichsfunktionen wurden in der Definition 2.16 eingeführt.

Der Vergleich der Werte und Referenzen von Datenbank-Objekten setzt in der Regel voraus, daß diese strukturgleich sind, d.h. daß die Datenbank-Objekte Instanzen derselben Klassen des globalen Schemas sind und daß Werte für dieselben Attribute angegeben sind. Aufgrund der Verschiedenheit der Datenbanken in Struktur und Umfang sowie Darstellung der Daten über eine Grundgesamtgheit von Realwelt-Objekten kommt es jedoch oft zu Strukturunterschieden und damit zu fehlenden Attributen und/oder Werten einzelner Attribute sowie zu Spezialfällen, die besonders berücksichtigt werden müssen.

Bevor wir auf den Vergleich von Attributwerten —der die Basis jedes Identifizierungsansatzes bildet— eingehen, stellen wir den Struktur- und Referenzstruktur-Vergleich von Datenbank-Objekten vor.

[11] Die Imputation $Anzahl\_Schwangerschaften = 0$ ist zwar rein formal für Männer ebenfalls korrekt, da sie keine Kinder gebären können — macht aber keinen Sinn. Insbesondere ist eine Auswertung bezüglich dieses Attributes nur für Frauen vernünftig.

2.4.1 Strukturvergleiche

Datenbank-Objekte können trotz der Tatsache, daß sie Instanzen ein und derselben Klasse sind, unterschiedliche (Sub-)Strukturen aufweisen, d.h. es können für ein Datenbank-Objekt Werte eines Attributes vorhanden sein, die für ein anderes Datenbank-Objekt fehlen, oder für Listen- oder Mengenwertige Attribute können unterschiedlich viele Werte gegeben sein. Ferner können Unterschiede in der Referenzierung auf andere Datenbank-Objekte vorliegen.

Für den Strukturvergleich zweier Datenbank-Objekte ist also zu ermitteln, ob für dieselben Attribute Werte (oder Referenzen) vorliegen.

Es sind drei Fälle zu unterscheiden: (1) Bei beiden Datenbank-Objekten liegen Werte für dieselben Attribute vor (Übereinstimmung), (2) Bei einem der Datenbank-Objekte fehlen Werte von Attributen, die bei dem anderen vorliegen (Enthaltensein), und (3) Es gibt Attribute, für die bei einem der zwei Datenbank-Objekte Werte angegeben sind, die aber bei dem jeweils anderen fehlen (Überschneidung).

Für eine Anwendung ist zu klären, inwieweit Strukturunterschiede auf die Verschiedenheit der zugrundeliegenden Realwelt-Objekte hindeuten, da in diesem Fall keine weitergehenden Vergleiche notwendig sind. Falls die Verschiedenheit der zugrundeliegenden Realwelt-Objekte nicht aus Strukturunterschieden ersichtlich ist, muß bei der Wahl von Vergleichsfunktionen darauf geachtet werden, daß die vorkommenden Abweichungen in der Struktur berücksichtigt werden, beispielsweise durch Hinzunahme entsprechender Vergleichswerte analog zur Berücksichtigung von NULL-Werten beim Vergleich, siehe Bemerkung 2.25 auf Seite 58.

Bei der Klassifikation für Paare von Datenbank-Objekten ist zudem sicherzustellen, daß trotz möglicher Strukturunterschiede für jedes Paar eine Entscheidung gefällt wird.

2.4.2 Referenzbaum-Vergleiche

In manchen Anwendungsfällen genügt es nicht, bei einem Vergleich zweier Datenbank-Objekte ausschließlich ihre Werte zu berücksichtigen, um eine Identifizierung vorzunehmen. Sofern diese Datenbank-Objekte über Referenzen zu anderen Datenbank-Objekten in der Datenbank verfügen, kann man diese in den Vergleich mit einbeziehen, analog zu den in Folgerung 1.30 auf Seite 38 dargelegten Identifizierungsansätzen, bei denen Referenzen berücksichtigt werden.

Bei einem an die tiefe Gleichheit von Datenbank-Objekten angelehnten Vergleich werden zwei Referenzbäume verglichen, was sehr aufwendig werden kann. Alternativ dazu können wir uns bei der Traversierung auf Objekt-Knoten der maximalen Referenzierungs-Tiefe n in den Referenzbäumen beschränken, d.h. wir gehen von der *Identifikation auf n-beschränkten Referenz-Bäumen* (siehe Seite 38) aus. Da aber schon ein Teil des n-beschränkten Referenzbaumes für die Objektidentifizierung genügen kann, können wir auch identifizierende Referenz-Strukturen (vgl. Definition 1.33 auf Seite 39) verwenden.

Eine Referenz-Vergleichsfunktion ist zusammengesetzt aus Wert-Vergleichen, die hierarchisch strukturiert sind, d.h. wir haben in der obersten Hierarchiestufe die Vergleichswerte der Attributwerte der beiden Datenbank-Objekte und in den tieferen Hierarchiestufen die Vergleichswerte von referenzierten Datenbank-Objekten. Wenn in einer Datenbank die Referenz-Struktur bekannt ist, läßt sich eine Referenz-Vergleichsfunktion als mehrdimensionale Wert-Vergleichsfunktion umsetzen. Das ist nicht zuletzt auch deshalb von Vorteil, da eine Klassifikation für Baum-artige Vergleichswerte schwieriger zu gewinnen ist. In dieser Vergleichsfunktion werden dann je Hierarchiestufe eine oder mehrere Dimensionen für die entsprechenden Wert-Vergleiche verwandt. Ferner kann eine Aggregation der Dimensionen in *eine* Dimension vorgenommen werden, beispielsweise durch eine kategorielle Codierung.

Wir erläutern dies an einem Beispiel.

Beispiel 2.27. Wir betrachten ein Personendaten-Register, in dem neben den Daten einer Person (Name, Vorname, Geburtsdatum, Adresse) Referenzen auf die Kinder sowie den Ehepartner einer Person geführt werden (sofern vorhanden und bekannt). Als Wert-Vergleichsfunktion sei die *Minimum-Edit-Distance* für die Namen und Vornamen mit Werten von 0 bis 3 sowie der Vergleich *identisch/nicht identisch* für Geburtsdatum und Adresse vorgesehen. Sofern Referenzen auf Ehepartner und/oder Kinder vorhanden sind, sollen diese in der gleichen Weise miteinander verglichen werden. Hierbei zeigt sich ein erstes Problem: Während eine Person (in Deutschland) nur einen Ehepartner haben kann, kann sie durchaus mehrere Kinder haben (und möglicherweise auch nicht mit demselben Partner). Diesen Besonderheiten muß bei dem Referenz-Vergleich Rechnung getragen werden. Also erhalten wir für den Vergleich zweier Datenbank-Objekte 4 Dimensionen für die Wert-Vergleiche, weitere 4 Dimensionen für die Wert-Vergleiche der referenzierten Ehepartner und bis zu $4 \cdot n_1 \cdot n_2$ Dimensionen, wobei n_1 die Zahl der referenzierten Kinder des einen und n_2 die erfaßte Kinderzahl des anderen Datenbank-Objektes bezeichnet. Die verwandten Dimensionen für die Kinder läßt sich auf $\min(n_1, n_2)$ verringern, falls nur der Vergleich des „am ehesten passenden Kindes“ angegeben werden soll. Dazu werden die Vergleichsergebnisse mittels einer Heuristik in eine Rangordnung gestellt. Die 4-dimensionalen Vergleichswerte der referenzierten Datenbank-Objekte können dann zusammengefaßt werden in einer nominalen Skala (z.B. 0: falls alle 4 Werte übereinstimmen, 1: falls lediglich die Adresse variiert, 2: falls lediglich das Geburtsdatum variiert, aber das Jahr übereinstimmt, 3: falls nur der Vorname ein wenig variiert, 4: sonst).

2.4.3 Wertvergleiche

2.4.3.1 Die Definition von Abstandsmaßen (Metriken) als Vergleichsfunktionen

Definition 2.28 (Metrik). $Y \subset X$ sei eine Attributmenge, $Y = (X_{i_1}, \ldots, X_{i_l})$, $X_{i_1}, \ldots, X_{i_l} \subset X = (X_1, \ldots, X_d)$.

Eine Abbildung $dist : \mathrm{dom}(Y) \times \mathrm{dom}(Y) \to [0, \infty)$ ist eine *Metrik*, genau

dann, wenn sie die folgenden Axiome für alle $x, y, z \in \mathrm{dom}(Y)$ erfüllt:

$$\text{(Reflexivität)} \qquad dist(x,y) = 0 \Longleftrightarrow x = y \tag{2.6a}$$

$$\text{(Symmetrie)} \qquad dist(x,y) = dist(y,x) \tag{2.6b}$$

$$\text{(Dreiecks-Ungleichung)} \qquad dist(x,y) \leq dist(x,z) + dist(z,y) \tag{2.6c}$$

Wir bezeichnen eine Metrik *dist* als *endlich*, wenn

$$\sup_{x,y \in \mathrm{dom}(Y)} dist(x,y) < \infty.$$

Eine endliche Metrik *dist* bezeichnen wir als (auf 1) *normiert*, wenn

$$\sup_{x,y \in \mathrm{dom}(Y)} dist(x,y) = 1.$$

Zu jeder endlichen Metrik $dist'$ läßt sich die normierte Metrik $dist := \frac{1}{c}\, dist'$ mit $c := \sup_{x,y \in \mathrm{dom}(Y)} dist'(x,y)$ bilden.

Definition 2.29 (Aggregation von Metriken). Es seien $Y^{(1)}, \ldots, Y^{(l)}$ paarweise disjunkte Attributmengen $Y^{(1)} \subset X$, $Y^{(i)} \cap Y^{(j)} = \emptyset$ und $dist_i : \mathrm{dom}(Y^{(i)}) \times \mathrm{dom}(Y^{(i)}) \to [0,\infty)$ seien Metriken. Dann ist mit $dist := \Phi(dist_1, \ldots, dist_l)$ für eine Funktion $\Phi : \big([0,\infty)\big)^l \to [0,\infty)$ genau dann eine aggregierte Metrik auf $\mathrm{dom}(Y^{(i)}) \times \mathrm{dom}(Y^{(i)})$ gegeben, wenn *dist* die Axiome (2.6) erfüllt.

Beispiel 2.30 *(Aggregation von Metriken).* Wir geben Beispiele für die Aggregation von Metriken gemäß Definition 2.29. Sei $d = (d_1, \ldots, d_l)^T \in \big([0,\infty)\big)^l$.

Mit $\Phi(d) = \sum_{i=1}^{l} w_i d_i$ ist die *lineare (gewichtete) Aggregation* von Metriken gegeben, $w_i \in \mathbb{R}_{>0}$.

Die *quadratische Aggregation* ist gegeben durch

$$\Phi(d) = d^T \mathbf{C} d = \sum_{i=1}^{l} \sum_{j=1}^{l} c_{ij} d_i d_j$$

mit einer positiv semi-definiten linearen Abbildung (Matrix) $\mathbf{C} : \mathbb{R}^l \to \mathbb{R}^l$ ($\mathbf{C}$ wird auch als quadratische Form bezeichnet).[12]

Falls die Matrix $\mathbf{C}$ als Inverse der für eine Stichprobe von Datensatz-Paaren (d.h. ihrer mehrdimensionalen Distanz-Werte) geschätzten Kovarianzmatrix ist, erhalten wir den Spezialfall der Mahalanobis-Distanz [Mah36].[13]

Bemerkung 2.31 *(Ähnlichkeitsmaß).* Es sei $dist : (\mathrm{dom}(Y))^2 \to [0,1]$ eine normierte Metrik. Dann ist durch

$$\mathrm{sim}(x,y) := 1 - dist(x,y) \tag{2.7}$$

[12] Eine lineare Abbildung $\mathbf{C} : \mathbb{R}^l \to \mathbb{R}^l$ ist *positiv semi-definit*, wenn $d^T \mathbf{C} d \geq 0$ für alle $d \in \mathbb{R}^l$ gilt.

[13] Bei der Mahalanobis-Distanz werden Korrelationen zwischen den verschiedenen Dimensionen korrigiert, so daß diese den aggregierten Wert nicht dominieren. Außerdem wird eine Reskalierung der Werte vorgenommen. Die grundlegende Annahme hierbei ist die einer mehrdimensionalen Gauß'schen Normalverteilung für die mehrdimensionalen Distanz-Werte.

ein (normiertes) Ähnlichkeitsmaß auf $\mathrm{dom}(Y)$ gegeben.

Während ein Ähnlichkeitsmaß die Symmetrie $\mathrm{sim}(x,y) = \mathrm{sim}(y,x)$ und Reflexivität ($\mathrm{sim}(x,y) = 1 \iff x = y$) erfüllt, läßt sich kein Analogon zur Dreiecks-Ungleichung finden.

Auf Nullwerte erweiterte Metrik

Sei X_i ein Attribut, in dem Nullwerte zugelassen sind (eine Diskussion von Nullwerten ist im Abschnitt 2.3.1 auf Seite 56 zu finden). Weiter sei $dist : (\mathrm{dom}(X_i))^2 \to [0,\infty)$ eine Metrik, die wir auf $\mathrm{dom}(X_i) \cup \{\perp\}$ fortsetzen wollen, die also auch für Nullwerte definiert werden soll. Die Fortsetzung soll die Axiome einer Metrik ebenfalls erfüllen.

Wegen (2.6a) würde $dist(\perp,\perp) = 0$ gelten — für Nullwerte ist aber die Eigenschaft $dist(\perp,\perp) > 0$ sinnvoll, da *Nichtwissen* über den Wert zweier Datensätze nicht minimale Distanz =0 ergeben sollte.[14] Demgemäß können wir (2.6a) ändern in

$$\begin{aligned} \text{(Reflexivität)} \qquad & \forall x,y \in \mathrm{dom}(X_i) : \big(dist(x,y) = 0 \iff x = y\big) \\ & \wedge\, dist(\perp,\perp) > 0 \;\wedge\; dist(x,\perp) > 0 \end{aligned} \tag{2.8a}$$

Da (2.6b) trivialerweise auch für $X_i(a) = \perp$ gilt, können wir (2.6b) einfach ersetzen durch

$$\text{(Symmetrie)}\ \forall x,y \in (\mathrm{dom}(X_i) \cup \{\perp\}) : dist(x,y) = dist(y,x). \tag{2.8b}$$

Schwieriger gestaltet sich die Dreiecksungleichung für Nullwerte: Aus der um $\perp$ ergänzten Dreiecksungleichung

$$\begin{aligned} \text{(Dreiecks-Ungleichung)} \qquad & \forall x,y,z \in \mathrm{dom}(X_i) \cup \{\perp\} : \\ & dist(x,z) \le dist(x,y) + dist(x,z) \end{aligned} \tag{2.8c}$$

folgt $c := dist(\perp,\perp) \le 2\, dist(\perp,x)$ für alle $x \in \mathrm{dom}(X_i)$. Wir erhalten

$$\forall x \in \mathrm{dom}(X_i) :\ dist(\perp,x) \ge \frac{c}{2}.$$

Wir fassen die Ergebnisse zusammen:

Definition 2.32 (NULL-Wert erweiterte Metrik). Sei X_i ein Attribut, in dem Nullwerte zugelassen sind und $dist : (\mathrm{dom}(X_i))^2 \to [0,\infty)$ sei eine Metrik. Sei $c \in \mathbb{R}, c \ge 0$ beliebig, aber fest. Dann ist $dist : (\mathrm{dom}(X_i) \cup \{\perp\})^2 \to [0,\infty)$ eine für *NULL-Werte erweiterte Metrik* wenn $dist$ die Axiome (2.6) erfüllt und außerdem für alle $x,y \in \mathrm{dom}(X_i)$ gilt:

$$dist(\perp,\perp) = c,\ dist(\perp,x) \ge \tfrac{1}{2}c \tag{2.9a}$$

$$dist(\perp,x) = dist(x,\perp) \tag{2.9b}$$

$$dist(x,y) \le dist(\perp,x) + dist(\perp,y) \tag{2.9c}$$

[14] Würden wir $dist(\perp,\perp) = 0$ fordern, kann die Definition 2.28 analog für $\perp$ angewandt werden, d.h. wir erhielten (2.6) mit $\mathrm{dom}(X) \cup \{\perp\}$ statt $\mathrm{dom}(X)$.

Zur Definition von Metriken für Attribute mit NULL-Werten stehen also zwei Alternativen zur Verfügung:

- Wir setzen $c := 0$ und fordern $dist(\perp, \perp) = 0$, $dist(\perp, x) \geq 0$ sowie $dist(x, y) \leq dist(\perp, x) + dist(\perp, y)$ für alle $x, y \in \mathrm{dom}(X_i)$, d.h. $\perp$ wird als Element des Wertebereiches angesehen, oder
- Wir wählen $c > 0$ beliebig und haben (2.9) zu erfüllen (falls $\mathrm{dom}(X_i)$ dicht ist, erhalten wir außerdem $dist(\perp, x) = \frac{1}{2}c$ für alle $x \in \mathrm{dom}(X_i)$).

2.4.3.2 Ordinale und kategorielle Vergleichsfunktionen

Definition 2.33. Eine eindimensionale Vergleichsfunktion $f : (\mathrm{dom}(Y))^2 \to R$ heißt *ordinal*, wenn eine totale Ordnung $(<)$ der Vergleichswerte $r \in R$ existiert, so daß gilt:

$$r < r' \iff \quad r \text{ ist } \textit{besser} \text{ als } r' \ ,$$

wobei das Prädikat „ist *besser* als“ eine Interpretation bezüglich des Grades der Übereinstimmung/der Ähnlichkeit zweier Paare von Datensätzen $(x, y), (x', y') \in (\mathrm{dom}(Y))^2$ mit $r = f(x, y)$ und $r' = f(x', y')$ darstellt. In diesem Fall werden die Vergleichswerte bezüglich dieser Ordnung aufgezählt (sie bilden eine Ordinalskala).

Falls diese Ordnung auf R mit einer Metrik gemäß Definition 2.28 angegeben werden kann, heißt die Vergleichsfunktion *metrisch* (Sie bildet eine metrische Skala).

Sofern keine totale Ordnung $(<)$ R existiert, ist f eine *kategorielle* Vergleichsfunktion (die Vergleichswerte bilden eine Nominalskala).

Eine mehrdimensionale Vergleichsfunktion heißt *metrisch, ordinal* bzw. *kategoriell*, wenn alle ihrer Komponenten metrisch/ordinal/nominal skaliert sind. Eine mehrdimensionale Vergleichsfunktion heißt gemischt skaliert, wenn verschiedene Skalen in den einzelnen Komponenten vorkommen.

Es ist offensichtlich, daß Abstandsmaße metrische Vergleichsfunktionen sind.

2.4.3.3 Verfeinerung von Vergleichsfunktionen

Analog zur Verfeinerung von Transformationsfunktionen (vgl. S. 53) führen wir die Verfeinerung von Vergleichsfunktionen ein.

Definition 2.34 (Verfeinerung von Vergleichsfunktionen). X sei eine Attributmenge. Eine Vergleichsfunktion $f' : (\mathrm{dom}(X))^2 \to R'$ ist eine *Verfeinerung* einer Vergleichsfunktion $f : (\mathrm{dom}(X))^2 \to R$ (Schreibweise $f' \succ f$), wenn gilt:

$$\dim R' \geq \dim R \tag{2.10a}$$

$$\exists g : R' \to R : \quad f = g \circ f' \tag{2.10b}$$

Umgekehrt bezeichnet man f auch als *Vergröberung* von f'.

Bei der Verfeinerung einer Vergleichsfunktion können zwei Fälle auftreten:

- **Gruppierung**: Es ist $\dim R = \dim R'$. Die Vergleichswerte einer oder mehrerer Komponenten f'_i von f' werden jeweils in f_i zusammengefaßt. Wir erhalten für alle i Komponenten $f_i \prec f'_i$ sowie $|f_i| \leq |f'_i|$, wobei $|\cdot|$ die Kardinalität der Menge der Vergleichswerte bezeichnet.
- **Projektion/Aggregation**: Es ist $\dim R < \dim R'$. Es wird eine Dimensionsreduktion vorgenommen, die Vergleichswerte der Komponenten f'_i von $f' = (f'_1, \ldots, f'_k)$ werden abgebildet auf $f = (f_1, \ldots, f_l), 0 < l < k$.

Beispiel 2.35. Ein typisches Beispiel für die Gruppierung ist die Reduzierung der Vergleichswerte einer Komponente, z.B. die Diskretisierung einer metrisch skalierten Vergleichsfunktion $f' : R \to [0, 1]$ mittels Aufteilung in eine Ordinalskala, z.B. ist $f \prec f'$ mit

$$f(x, y) = \begin{cases} 0 & f'(x, y) = 0, \\ 1 & 0 < f'(x, y) \leq \frac{1}{2} \\ 2 & f'(x, y) > \frac{1}{2}. \end{cases}$$

Beispiel 2.36. Ein einfaches Beispiel für die Projektion ist die Summation mehrdimensionaler Distanzen mit $g : \mathbb{R}^k \to \mathbb{R}, g : x \mapsto \sum_{i=1}^{k} x_i$:

$$f = g \circ f' = g((f'_1, \ldots, f'_k)) = \sum_{i=1}^{k} f'_i.$$

Beispiel 2.37. Wir betrachten ein Attribut X_1 das den Wertebereich $\mathrm{dom}(X_1)$ = {`TEXT`} habe und Zeichenketten über einem Alphabet $\mathcal{A}$ enthalte, etwa den Titel eines Buches in einem Bibliothekskatalog. Für dieses Attribut haben wir im Beispiel 2.23 auf Seite 54 ableitbare Attribute $Y_j, j = 1, \ldots, 20$ angegeben. Dann können wir folgende Vergleichsfunktionen $f : Y \to f(Y)$ für $Y = Y_1, \ldots, Y_{20}$ definieren:

$$\begin{aligned}
\texttt{f1}(x, y) = \texttt{CountSameWords}(x_4, y_4) &:= \texttt{length}(x_4 \cap y_4) \\
\texttt{f2}(x, y) = \texttt{PortionSameWords}(x_4, y_4) &:= \frac{\texttt{length}(x_4 \cap y_4)}{\max(\texttt{length}(x_4), \texttt{length}(y_4))} \\
\texttt{f3}(x, y) = \texttt{CountSameWords2}(x_5, y_5) &:= \sum_j (x_5)_j \cdot (y_5)_j \\
\texttt{f4}(x, y) = \texttt{TF_IDF_Similarity}(x_7, y_7) &:= \sum_j (x_7)_j \cdot (y_7)_j
\end{aligned}$$

$$\texttt{f5}(x, y) = \texttt{PSWLDiscrete}(x_4, y_4) := \begin{cases} 0 & : 0.9 \leq \texttt{PortionSameWords}(x_4, y_4) \leq 1, \\ 1 & : 0.7 \leq \texttt{PortionSameWords}(x_4, y_4) < 0.9, \\ 2 & : 0.5 \leq \texttt{PortionSameWords}(x_4, y_4) < 0.7, \\ 3 & : \text{otherwise.} \end{cases}$$

mit $\texttt{f1} = \texttt{CountSameWords}, \texttt{f2} = \texttt{PortionSameWords} : Y_4 \to [0,1]$,
$\texttt{f3} = \texttt{CountSameWords2} : Y_5 \to \mathbb{N}$,
$\texttt{f4} = \texttt{TF\_IDF\_Similarity} : Y_7 \to [0,\infty) \subset \mathbb{R}$ und
$\texttt{f5} = \texttt{PSW\_Discrete} : Y_4 \to \{0,1,2,3\}$.

Wir erhalten folgende Verfeinerungen:

- $\texttt{f5} \prec \texttt{f2}$, da der Wertebereich diskretisiert wird
- $\texttt{f1} \prec \texttt{f3}, \texttt{f3} \prec \texttt{f1}$, da sogar $\texttt{f1}(x,y) = \texttt{f3}(x,y)$ für alle $(x,y) \in (domY)^2$ gilt.
- $\texttt{f1} \prec \texttt{f2}, \texttt{f2} \prec \texttt{f1}$ bei Benutzung der `length`-Funktion
- Wir erhalten $\texttt{f3} \not\prec \texttt{f5}, \texttt{f5} \not\prec \texttt{f3}$, da beide Funktionen zwar dasselbe Skalarprodukt verwenden, aber auf verschiedene Attribute angewandt werden.

2.5 Die Software-Architektur: Mediator-Wrapper-Architektur mit Identifizierung

Als Software-Architektur zur Umsetzung des dargestellten Modells für die Objekt-Identifizierung verwenden wir das von Wiederhold 1992 eingeführte Zwei-Schichten-Modell [Wie92], bestehend aus einer Mediator- und Wrapper-Schicht. In diesem Modell gibt es je zu integrierender Datenbank (mindestens) einen Wrapper, der folgende Aufgaben hat:

- Kommunikation mit der Datenbank (Verbindung herstellen, Abfrage senden, Ergebnis-Daten empfangen),
- Extraktion der gewünschten Daten aus der Antwort, gegebenenfalls Ausführung von zusätzlicher Kommunikation,
- Transformation der Daten in ein vorher festgelegtes lokales Schema und Daten-Format.

Der Mediator übernimmt in diesem Zwei-Schichten-Modell:

- Die Kommunikation mit Benutzern bzw. einem Interface (Verbindung herstellen, Abfrage empfangen, Antwort senden),
- Abfragezerlegung, Abfrageplanung sowie Weitergabe einzelner Teil-Abfragen an die Wrapper,
- Zusammenfügen der Ergebnis-Daten der einzelnen Wrapper zu einer Antwort (Homogenisierung unter Transformation der Daten aus den lokalen Schemata in ein globales Schema), gegebenenfalls unter Verwendung von Datenbereinigung und Objektidentifizierung (siehe folgende Punkte),
- Datenbereinigung, bei der Inkonsistenzen in den Daten (soweit als möglich) aufgelöst werden (z.B. durch Standardisierung),

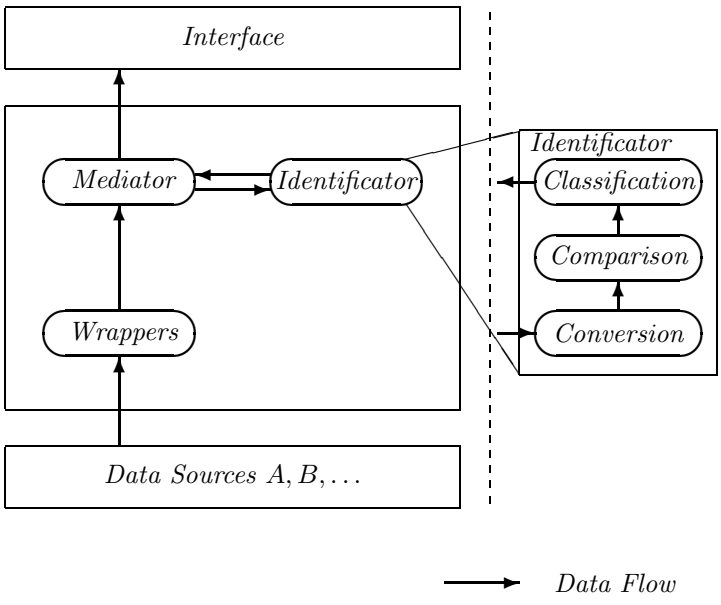


Abbildung 2.4: Die Software-Architektur

- Objektidentifizierung, bei der diejenigen Daten, die sich auf äquivalente Realwelt-Objekte beziehen, erkannt und zusammengefaßt werden.

Die Aufgabe der Objektidentifizierung wird in der Architektur durch das Modul *Identificator* übernommen, das —basierend auf einer vorher gelernten Klassifikation— Duplikate aus einer Menge von Datensätzen herausfiltert. Die Ausgabe besteht aus einer Liste von Paaren, die als Duplikat klassifiziert wurden. Gegebenenfalls wird die Liste für jedes Paar ergänzt um eine Bewertung (*engl.: score*) zwischen 0 und 1.

Beispiel 2.38 *(BOA — Berliner Wohnungssuchmaschine).* Für die prototypisch in PHP realisierte Berliner Wohnungssuchmaschine wurden Wohnungsanzeigen aus den Online-Editionen der Tageszeitungen *Tagesspiegel* und *Berliner Morgenpost* abgefragt und in einer lokalen Datenbank gespeichert. Dazu gab es je Online-Edition einen Wrapper, der mit den Abfrageparametern *Bezirk* und *Wohnungsgröße* aufgerufen werden konnte. Im Wrapper waren umgesetzt:

- Herstellen einer Verbindung zur Online-Edition über HTTP,
- Senden von Abfragen mit *Bezirk* und *Wohnungsgröße* an die Online-Datenbank durch automatisches Ausfüllen von HTML-Formularen (Aufruf der HTTP-Methoden GET oder POST je nach Online-Datenbank),
- Extraktion der Daten der Wohnungsanzeigen aus der Ergebnis-Seite, bei mehreren Ergebnis-Seiten Abruf der Folgeseiten und Extraktion der Daten aus ihnen,
- Speichern der Daten in einer temporären Tabelle (unter Angabe der Datenbank und einem Zeit-Stempel).

Der Mediator diente dazu, die beiden Wrapper mit den entsprechenden Parametern zu starten und ihre Ergebnisse zusammenzufassen. Durch Daten-Extraktion konnten weitere Attribute aus dem Anzeigentext gewonnen werden, z.B. Etage und Ausstattungsmerkmale. Durch Abgleich mit einem Straßenregister konnte zudem die Straße extrahiert werden.

Kapitel 3

Klassifikation

Was rumpelt und pumpelt
in meinem Bauch herum?

Ich meinte, es wären sechs Geißlein,
so sind's lauter Wackerstein!

Aus „Der Wolf und die sieben Geißlein“

Inhalt dieses Kapitels. Die Entscheidung, ob zwei Datensätze sich auf ein und dasselbe Objekt beziehen, soll durch eine Klassifikation gefällt werden. Die Unterschiede zwischen der überwachten und nicht überwachten Klassifikation werden erläutert. Weitere Gesichtspunkte wie die Skalen des verwendeten Merkmalsraumes einer Klassifikation oder der Einfluß von Abhängigkeiten im mehrdimensionalen Fall werden diskutiert. Schließlich werden verschiedene Klassifikationsverfahren detailliert dargestellt und hinsichtlich ihrer Eignung untersucht.

3.1 Das Lernen einer Klassifikation

Für den Vergleichsraum R soll eine Entscheidungsregel δ gefunden werden, so daß für jeden möglichen Vergleichswert $r \in R$ zweier Datensätze entschieden werden kann, ob sie sich auf ein und dasselbe *Realwelt-Objekt* oder auf zwei verschiedene *Realwelt-Objekte* beziehen.

Zur Gewinnung der Entscheidungsregel δ lassen sich sowohl *überwachte* als auch *nicht-überwachte Lernverfahren* einsetzen. Im Folgenden werden beide Lernverfahren unter dem Begriff *Verfahren zur Klassifikation einer Stichprobe* zusammengefaßt. Die Klassifikation wurde in Definition 2.17 auf Seite 50 eingeführt.

Eine Klassifikation wird auf einer Stichprobe S gelernt. Die Stichprobe S besteht aus einer Tabelle von N^S Datensätzen s^k, $k = 1, \ldots, N^S$ mit Attributen $X_1, \ldots, X_k$, aus der eine Auswahl von d Attributen als *Einflußgrößen* zum Lernen einer Klassifikation verwandt wird. Wir gehen in den folgenden Abschnitten auf die Besonderheiten des überwachten und nicht-überwachten Lernens ein.

3.1.1 Überwachtes Lernen

Das überwachte Lernen (*engl.: supervised learning*) zeichnet sich durch folgende Eigenschaften aus:

- Es gibt eine feste Anzahl von *Einflußgrößen* $X_1, \ldots, X_d$, $d \in \mathbb{N}$ und eine *Zielgröße* X^* (es können auch mehrere Zielgrößen $X_1^*, \ldots, X_j^*$ gegeben sein),
- Die Klassen einer Klassifikation sind *a priori* durch den Wertebereich $\mathrm{dom}(X^*)$ der *Zielgröße* X^* bestimmt, $\mathcal{K} = \{\mathcal{K}_1, \ldots, \mathcal{K}_j\}$, $j = |\mathrm{dom}(X^*)|$,
- Die Stichprobe wird durch den Supervisor bezüglich der *Zielgröße* X^* bewertet, sofern der Wert von X^* nicht schon bekannt ist, und
- Die Klassenzugehörigkeit bzw. der Wert $x^* \in \mathrm{dom}(X^*)$ der *Zielgröße* X^* kann für alle Werte $(x_1, \ldots, x_d) \in \mathrm{dom}(X_1 \times \ldots \times X_d)$ angegeben werden (Vorhersage der Zielgröße).[1]

Der Hauptunterschied zum nicht-überwachten Lernen besteht darin, daß eine manuelle Bewertung der Daten durch einen Supervisor vorgenommen werden muß.

Überwachte Lernverfahren sind z.B.:

1. Entscheidungsbaum-Verfahren
2. Statistische Tests (wie z.B. im Record Linkage verwandt)
3. Nächste-Nachbarn Klassifikation
4. Neuronale Netze
5. Support Vector Machines [Vap98]
6. (Lineare und Nichtlineare) Diskrimination
7. Instance Based Learning [AKA91]

3.1.2 Nicht-überwachtes Lernen

Das nicht-überwachte Lernen (*engl.: unsupervised learning*) zeichnet sich durch folgende Eigenschaften aus:

- Es gibt eine feste Anzahl von *Einflußgrößen* $X_1, \ldots, X_d$, $d \in \mathbb{N}$ und keine *Zielgröße*,
- Art und Anzahl der Klassen einer Klassifikation sind nicht *a priori* bestimmt (können aber oft durch Parameter beeinflußt werden),

[1] Für eine nicht-disjunkte Klassifikation gemäß Definition 2.17 ergeben sich Klassenzugehörigkeiten zu mehreren Klassen.

- Da die Klassifikation nicht auf einer *Zielgröße* beruht, bedarf es in der Regel einer Interpretation der ermittelten Klassen,[2]
- Die Klassenzugehörigkeiten können für alle Vektoren

$$(x_1, \ldots, x_d) \in \mathrm{dom}(X_1 \times \ldots \times X_d)$$

angegeben werden.

Nicht-überwachte Lernverfahren sind z.B.:

1. Klassifikation mit Assoziationsregeln,
2. Abstandsmaß-basiertes Clustering wie bei der Nearest-Neighbor Klassifikation verwandt,
3. Bayes-basierte Klassifikation,
4. Fuzzy- und Neuro-Fuzzy Clustering.

Bemerkung 3.1. Nicht-überwachte Lernverfahren lassen sich auch für bewertete Stichproben mit einer Zielgröße X^* verwenden (die Zielgröße kann, aber muß nicht als *zusätzliche* Einflußgröße mit in den Lernprozeß einbezogen werden). Aus einer gelernten Klassifikation $\mathcal{K}$ ohne Vorhersage der Zielgröße läßt sich dann eine Vorhersage für die Zielgröße ermitteln, indem man die gewonnenen Klassen hinsichtlich der Zielgröße interpretiert.

Für die Vorhersage kann man beispielsweise die Verteilung der Werte der Zielgröße in den einzelnen Klassen verwenden (geschätzt durch die Häufigkeiten in der Lernstichprobe); mit diesem Vorgehen erhält man eine nicht-disjunkte Klassifikation.

Im Abschnitt 3.3.3.2 auf Seite 86 wird die Gewinnung einer Klassifikation aus Assoziationsregeln beschrieben.

3.1.3 Die Klassifikation einer Stichprobe von Datensatzpaaren

Aus einer Datensatz-Stichprobe S sei eine Datensatzpaar-Stichprobe $P \subset S \times S$ gezogen, die einen bestimmten Anteil von dubletten Paaren enthalte (Die Besonderheiten der Stichprobenziehung werden detailliert im Abschnitt 6.2.2 auf Seite 152 beschrieben). Die Vergleichsfunktionen $f_i : (\mathrm{dom}(X))^2 \rightarrow R_i, i = 1, \ldots, d$ seien implementiert. Dann können die Vergleichswerte $r_i = f_i(a, b)$ für Datensatzpaare berechnet und der Vergleichswert r_i für jedes Datensatzpaar in der Datensatzpaar-Stichprobe P in einem vorher angefügten Attribut ri gespeichert werden. Die Datensatzpaar-Stichprobe P genügt also folgender Tabellenstruktur:

$$P(PairID, ID\_of\_a, ID\_of\_b, Same, r1, \ldots, rd) \tag{3.1}$$

[2]Die Interpretation kann z.B. durch einen Supervisor vorgenommen werden, wobei die Bewertung von einigen wenigen Klassen einen geringeren Aufwand erfordert als die Bewertung einer gesamten Stichprobe.

Das Attribut *Same* der Tabelle P hat den Wertebereich $\mathrm{dom}(Same) = \{0, 1\}$. Mit dem Attribut *Same* wird erfaßt, ob es sich bei den Datensätzen a, b des Paares $pair = (a, b)$ um ein Duplikat handelt, d.h. ob sie sich auf *äquivalente* Realwelt-Objekte beziehen ($a \equiv b$): $Same(pair)$ = 'Yes' bzw., falls sich a und b auf *nicht äquivalente* Realwelt-Objekte beziehen ($a \not\equiv b$) $Same(pair)$ = 'No'. Der Wert von *Same* wird bei der Stichprobenziehung gesetzt, je nachdem ob es sich um ein dublettes Paar handelt, oder ob nicht (diese Bewertung entspricht dem Vorgehen beim überwachten Lernen, sie ist für die Ermittlung der Korrektheit auch für nicht-überwachte Lernverfahren notwendig).

Die Stichprobe P wird (randomisiert) in eine *Lernstichprobe* L und in eine *Teststichprobe* T aufgeteilt, wobei eine Quotierung der dubletten Paare proportional zum Anteil dubletter Paare in P angewandt wird. Die Lernstichprobe L wird zum Lernen einer Klassifikation für Paare $(a, b) \in A \times B$ verwandt—sie stellt den Input eines Klassifikationsverfahrens dar.

Die gelernte Klassifikation wird anschließend auf die Teststichprobe T angewandt, d.h. für jedes Paar in T wird der Wert der Zielgröße *Same* bestimmt und mit dem wahren Wert verglichen, so daß wir unverzerrte Fehlerraten der Klassifikation erhalten.[3]

3.2 Überblick über geeignete Methoden für die Klassifikation

In Kapitel 2 wurde dargestellt, wie sich das Problem der *Objektidentifizierung* auf ein Klassifikationsproblem zurückführen läßt. Für die Qualität einer Klassifikation ist die sinnvolle Wahl eines Klassifikationsverfahrens jedoch entscheidend. Grundsätzlich läßt sich *jedes Klassifikationsverfahren* einsetzen, die Auswahl wird jedoch durch die Art der verwandten Vergleichsfunktionen beschränkt, denn je nach Skalierung der einzelnen Dimensionen des Vergleichsraumes R sind nur bestimmte Verfahren einsetzbar.

Für den Fall kategoriell-wertiger Vergleichsfunktionen sind Regel-erzeugende Methoden wie Entscheidungsbäume und Assoziationsregel-Verfahren einsetzbar.

Sofern eine Verteilungsfunktion (z.B. für kategorielle Werte die Multinomialverteilung) geschätzt werden kann, so kommen statistische Verfahren ins Spiel, wie der monotone Likelihood-Quotienten-Test im klassischen Record-Linkage-Ansatz (Abschnitt 3.3.2).

Für kardinale Skalen —die bei Verwendung von Metriken als Vergleichsfunktionen entstehen— läßt sich die Nächste-Nachbarn-Klassifikation verwenden (Abschnitt 3.3.4). Als Nächste-Nachbarn-Klassifikationsmethode fassen wir auch die *Nicht-lineare Klassifikation* auf, bei der eine Aggregation von Metriken vorgenommen wird, beispielsweise mit einer Quadratischen Form. Ein weiterer Spezialfall der Nächste-Nachbarn-Klassifikation ist die *Sorted Neighbourhood Method*, auf die ebenfalls eingegangen wird.

[3] Alternative Ansätze zur Bestimmung von Fehlerraten sind *Cross Validation* oder *Bootstrapping*, vgl. Efron [Efr79] und Freedman [Fre81].

Es sei erwähnt, daß eine Vergröberung der Skalierung vorgenommen werden kann, beispielsweise um Methoden (wie Record Linkage) einsetzen zu können, die nur für nominale Skalen verwendbar sind. Dazu wird dann eine kardinale Skalierung durch eine Zerlegung in Intervalle in eine Ordinalskala vergröbert, und diese kann dann —unter Vernachlässigung der Ordnung— als Nominalskala verwandt werden. Es sei aber bemerkt, daß jede Vergröberung zu einem Verlust an Information führen kann. Deshalb ist zu empfehlen, die Wahl der Vergleichsfunktionen in Abhängigkeit vom anzuwendenden Verfahren vorzunehmen.

Im Folgenden wird eine Auswahl verwendbarer Klassifikationsverfahren mit ihren Besonderheiten dargestellt. Die Angabe der Skalen der Dimensionen des Vergleichsraumes (kardinal, ordinal, nominal oder gemischt) sowie der Anzahl der Vergleichsfälle je Dimension liefert das Kriterium für die Einsetzbarkeit eines Verfahrens.

1. Decision Tree — Entscheidungsbaum
   - Skalen: beliebig gemischt
   - Vergleichswerte je Dimension: beliebig viele ($\mathbb{R}_+$ zulässig!)
   - Fehlerraten: schwer kalkulierbar, nicht skalierbar
   - Bewertung eines Vergleichswertes: nicht möglich
   - zusätzliche Annahmen: keine
   - Besonderheiten: Vorhersage der Klassenzugehörigkeit für jeden möglichen Vergleichswert
2. Record Linkage — statistischer Test
   - Skalen: nominal, ordinal[4]
   - Vergleichswerte je Dimension: wenige diskrete Fälle (2–10)

[4]Durch Modellrestriktionen ist es möglich, ordinale Skalen bei der Schätzung zu berücksichtigen.

Tabelle 3.1: Die Eignung von Klassifikationsmethoden für verschiedene Skalen

	Skalen des Merkmalsraumes[1]				
Klassifikationsmethoden	nominal	ordinal	kardinal[2]	stetig	gemischt[3]
Record Linkage	++	+	+	−−	−
Assoziationsregeln	++	++	+	−−	−
Entscheidungsbäume	++	++	++	++	++
k-nächste Nachbarn	−−	−	++	++	−−
nicht-lineare Klassifikation	−	0	++	++	−−

[1]Die Eignung variiert von (−−) ungeeignet über (0) mäßig geeignet bis hin zu (++) gut geeignet

[2]diskrete und endlich viele kardinale Werte

[3]gemischte Skalen (z.B. nominal/stetig)

- Fehlerraten: skalierbar
- Bewertung eines Vergleichswertes: durch Wahrscheinlichkeitsverteilung
- zusätzliche Annahmen: Abhängigkeitsstruktur von Attributen bzw. Attributgruppen (Dimensionen)
- Besonderheiten: Vorhersage der Klassenzugehörigkeit für jeden möglichen Vergleichswert

3. Klassifikation mit Assoziationsregeln

   - Skalen: nominal
   - Vergleichswerte je Dimension: endlich viele diskrete Fälle (2–20)
   - Fehlerraten: skalierbar
   - Bewertung eines Vergleichswertes: durch passende Regeln
   - zusätzliche Annahmen: keine
   - Besonderheiten: Vorhersage der Klassenzugehörigkeit für jeden möglichen Vergleichswert

4. Nächste-Nachbarn-Klassifikation (Aggregation von mehrdimensionalen Abstandsmaßen)

   - Skalen: nur kardinal
   - Vergleichswerte je Dimension: beliebig viele ($\mathbb{R}_+$ zulässig!)
   - Fehlerraten: kaum skalierbar
   - Bewertung eines Vergleichswertes: durch Abstand zum Nullpunkt[5]
   - zusätzliche Annahmen: implizite Wahrscheinlichkeitsverteilungsannahme (z.B. Normalverteilung)
   - Besonderheiten: Abhängigkeiten über Kovarianz modellierbar (zusätzliches Schätzproblem: Kovarianz-Matrix)
   - Bemerkung: bekannter Spezialfall: Mahalanobis-Distanz

Weitere Verfahren, die sich zum Gewinnen einer Klassifikation eignen[6]:

- Nicht-lineare Diskrimination
- Neuronale Netze,
- Support Vector Machines,
- Fuzzy-basierte Klassifikation und Neuro-Fuzzy-Clustering,
- Pattern Matching,
- Inductive Learning sowie
- Bayes-basierte Klassifikation.

[5] Genaugenommen wird die Klassifikation für Datensätze vorgenommen und nicht für Vergleichswerte von Paaren.

[6] Diese Verfahren wurden nicht in die Untersuchung einbezogen, wir zeigen die Verwendung für obige vier Klassifikationsmethoden, die Anwendung anderer Verfahren ist analog möglich.

3.3 Klassifikationsmethoden

In den nächsten Abschnitten wird dargestellt, wie sich eine Auswahl von Klassifikations-Verfahren für die Klassifikation von Vergleichswerten einsetzen lassen.

3.3.1 Entscheidungsbäume

Entscheidungsbaum-Verfahren eignen sich sehr gut, um aus einer Beispielsammlung Entscheidungsregeln abzuleiten (z.B. aus einer Lernstichprobe von Paaren von Datensätzen mit einem klassifizierenden Attribut *Same*). Bekannte Verfahren sind C4.5 oder CART[7] (vgl. Quinlan [Qui93], Breiman et al. [BFOS84], Michie, Taylor und Spiegelhalter [MST94]) Entscheidungsbaum-Verfahren sind auch in den meisten *Data-Mining* Software-Paketen enthalten. Wir verwenden die frei verfügbare Software von Borgelt [Bor03b].

Das grundlegende Prinzip von Entscheidungsbaum-Verfahren besteht darin, eine Lernstichprobe anhand von Attributwerten schrittweise zu zerlegen, so daß jede Zerlegung die Aufteilung bezüglich eines vorgegebenen Klassenattributes verbessert. Die Methode der Partitionierung wird sukzessive auf die erhaltenen Zerlegungen der Lernstichprobe angewandt, bis ein Abbruchkriterium erfüllt ist.

Eine Lernstichprobe L wird mittels Achsen-paralleler Schnitte aufgeteilt, wobei die Achsen in unserem Anwendungsfall dem durch die d Dimensionen $R_1, \ldots, R_d$ aufgespannten d-dimensionalen Vergleichsraum entsprechen. Die Beispiele $(a, b) \in L$ werden anhand ihrer Vergleichswerte $f_j(a, b)$, $j = 1, \ldots, d$ aufgeteilt; für ordinal und kardinal skalierte (Vergleichs-)Attribute R_j durch Bedingungen der Form $f_j(a, b) \diamond c$, $\diamond \in \{\leq, <, =, >, \geq\}$), während für nominal skalierte Attribute nur Bedingungen der Form $f_j(a, b) = c$ verwendbar sind.

Die *Verbesserung der Aufteilung* wird mit einem Auswahlmaß gemessen, beispielsweise dem unten angegebenen Informationsgewinn. Gewählt wird jeweils diejenige Aufteilung, für die das gewählte Auswahlmaß maximal ist.

Als *Abbruchkriterium* kann eine Beschränkung der maximalen Tiefe des induzierten Entscheidungsbaumes vorgenommen werden oder eine Einhaltung einer Mindestanzahl von Beispielen in den Knoten des Baumes gefordert werden.

Der *Informationsgewinn* basiert auf der *Shannonschen Entropie* [Sha48]. Die *Entropie* wird für eine Menge von Beispielen L, für die jeweils eine Klasse k des klassifizierenden Attributes R_0 (in unserem Fall ist $R_o = Same$) angegeben ist,[8] wie folgt berechnet:

$$\text{Entropy}(R_0; L) = -\sum_{k \in \text{dom}(R_0)} p_k^L \log_2 p_k^L, \tag{3.2}$$

[7] CART ist das Akronym für Classification and Regression Tree.

[8] Wir interpretieren das klassifizierende Attribut R_0 hierbei als Zufallsvariable, die Entropie ist ein Maß für die Ungenauigkeit von Realisierungen der Zufallsvariable R_0 in L.

mit der Konvention $0 \log_2 0 = 0$ und der abkürzenden Schreibweise $p_k^L := \mathbf{P}\{\, R_0(pair) = k \mid pair \in L \,\}$. Wir schätzen die Wahrscheinlichkeit p_k^L durch die relativen Häufigkeiten des Auftretens der Klasse k in L, d.h.

$$\hat{p}_k^L = \frac{|\,\{R_0(pair) = k,\, pair \in L\}\,|}{|L|}\,.$$

Der Informationsgewinn (*engl.: information gain*) IG einer Aufteilung von L in $L_1, \ldots, L_n$ (d.h. $L = \bigcup_{i=1}^n L_i, L_i \cap L_j = \emptyset, i \neq j$) ist gegeben durch:

$$\begin{aligned} IG(L, \{L_j\}_{j=1,\ldots,n}) &:= \text{Entropy}(R_0; L) - \sum_{j=1}^{n} \text{Entropy}(R_0; L_j) \qquad (3.3) \\ &= -\sum_{k \in \text{dom}(R_0)} p_k^L \log_2 p_k^L + \sum_{j=1}^{n} \sum_{k \in \text{dom}(R_0)} p_k^{L_i} \log_2 p_k^{L_i}. \end{aligned}$$

Weitergehende Informationen zu Auswahlmaßen (z.B. zum *Informationsgewinn-Verhältnis* und *Gini-Index*) sind in der Arbeit von Borgelt und Kruse [BK98] zu finden.

3.3.2 Record Linkage

Der von Howard Newcombe, Ivan Fellegi und Alan Sunter in den 50ern und 60ern entwickelte Ansatz des *Record Linkage* wird für die Zusammenführung großer Datenbestände seit über 30 Jahren angewandt.[9] Hauptsächlich wird *Record Linkage* für die Identifizierung von Personen in Statistischen und Adress-Datenbanken eingesetzt, eine weitere Anwendung ist die Zusammenführung medizinischer Datensätze beispielsweise in einem Krebsregister. Die Vielfalt der Anwendungsgebiete und den Stand der Forschung findet man in den Proceedings der beiden Konferenzen in Arlington, Virginia ([KA85], [AJ97]). Über die verfügbare Software für *Record Linkage* findet man Informationen in der Arbeit von Winkler [Win01b].

Der *Record Linkage* Ansatz basiert auf einem statistischen Test, dem *Likelihood-Quotienten-Test*, der zweiseitig angewandt wird. Als Teststatistik $\lambda(r)$ wird der Quotient der Wahrscheinlichkeiten des Auftretens von Werten einer Vergleichsfunktion $f : (\text{dom}(X))^2 \to R$, $f(a, b) = r$ verwandt:

$$\lambda(r) := \frac{\mathbf{P}\{\, f(a,b) = r \mid a \equiv b \,\}}{\mathbf{P}\{\, f(a,b) = r \mid a \not\equiv b \,\}}, \qquad (3.4)$$

wobei es sich um bedingte Wahrscheinlichkeiten handelt. Die Konditionierung mit $a \equiv b$ bzw. mit $a \not\equiv b$ bedeutet, daß das betreffende Paar (a, b) sich auf *dasselbe* bzw. auf *verschiedene Realwelt-Objekte* bezieht. Da üblicherweise im *Record Linkage* von endlich vielen (und mithin diskreten) Vergleichswerten $r \in R$, $|R| < \infty$ ausgegangen wird, wird eine Multinomial-Verteilung zugrundegelegt. Es könnten aber auch andere Verteilungssannahmen gemacht werden. Die Parameter der beiden diskreten Wahrscheinlichkeits-Verteilungen

$$\mathbf{P}\{\, f(a,b) = r \mid a \equiv b \,\}\,, \quad \mathbf{P}\{\, f(a,b) = r \mid a \not\equiv b \,\} \qquad (3.5)$$

[9] Die grundlegenden Arbeiten zum Record Linkage sind [NKAJ59] und [FS69].

werden auf einer Lernstichprobe L von Datensatz-Paaren (a, b) geschätzt, für die angegeben ist, ob es sich um Duplikate handelt ($a \equiv b$) oder ob nicht ($a \not\equiv b$). Weiter sei eine d-dimensionale Vergleichsfunktion $f = (f_1, \ldots, f_d)$ deren Vergleichswerte für Paare (a, b) im Vergleichsraum R liegen. Eine Schätzung der Wahrscheinlichkeits-Verteilungen (3.5) kann aus den relativen Häufigkeiten des Auftretens eines Vergleichswertes $r \in R$ für Paare in L vorgenommen werden:

$$\hat{\mathbf{P}}\{\, r \mid a \equiv b \,\} := |\,\{(a,b) \in L \mid (\forall j : f_j(a,b) = r_j) \wedge a \equiv b\}\,| / c_0, \qquad (3.6a)$$

$$\hat{\mathbf{P}}\{\, r \mid a \not\equiv b \,\} := |\,\{(a,b) \in L \mid (\forall j : f_j(a,b) = r_j) \wedge a \not\equiv b\}\,| / c_1, \qquad (3.6b)$$

mit $c_0 := |\,\{(a,b) \in L \mid a \equiv b\}\,|$ und $c_1 := |\,\{(a,b) \in L \mid a \not\equiv b\}\,|$. Diese Wahrscheinlichkeiten lassen sich jedoch gemäß (3.6) nur dann hinreichend genau schätzen, wenn der Stichprobenumfang $|L\,|$ der Lernstichprobe groß gewählt wird. Erfahrungsgemäß benötigt man für die Schätzung einer Multinomialverteilung mit m Fällen mindestens $3m$ Beispiele (vgl. z.B. [Chr90]). Für einen d-dimensionalen Vergleichsraum $R = R_1 \times \cdots \times R_d$ ist die Kardinalität m gegeben durch: $m = \prod_{i=1}^{d} M_i = \prod_{i=1}^{d} |R_i|$. Die Kardinalität von R kann schon für Vergleichsfunktionen mit wenigen Vergleichswerten recht groß werden, so daß der benötigte Stichprobenumfang die Zahl vorhandenen dubletten Datensatzpaaren übersteigen kann.[10] Der Schätzung der Wahrscheinlichkeiten (3.6) aus einer Stichprobe kommt daher eine besondere Bedeutung zu. In der Regel sind dazu zusätzliche Annahmen notwendig, z.B. über die Unabhängigkeit einzelner Dimensionen des Vergleichsraumes oder über das Monotonieverhalten der Verteilungsfunktion. Ein Ansatz zur verbesserten Schätzung der Wahrscheinlichkeitsverteilungen (3.6) wird im Abschnitt 3.3.2.1 dargestellt.

Wir bilden den Quotienten dieser beiden Verteilungen als Teststatistik:

$$\hat{\lambda}(r) \;:=\; \frac{\hat{\mathbf{P}}\{\, f(a,b) = r \mid a \equiv b \,\}}{\hat{\mathbf{P}}\{\, f(a,b) = r \mid a \not\equiv b \,\}}$$

Dieser Quotient $\hat{\lambda}(r)$ wird auch als *Wettquotient* (*engl.: odds*) für einen Vergleichswert $r = (r_1, \ldots, r_d) \in R$ bezeichnet, da das Verhältnis

$$\hat{\mathbf{P}}\{\, r \mid a \equiv b \,\} : \hat{\mathbf{P}}\{\, r \mid a \not\equiv b \,\}$$

dubletter Paare zu nicht dubletten Paaren widerspiegelt, die einen Vergleichswert $r \in R$ haben. Der Quotient (3.4) ist ein Indikator dafür, wie gut ein Datensatzpaar $(a, b) \in A \times A$ mit Vergleichswert $r \in R$ zueinander paßt:

- $\hat{\lambda}(r) > 1$: Es ist wahrscheinlicher, daß sich a und b auf ein und dasselbe Realwelt-Objekt beziehen, als das Gegenteil,
- $\hat{\lambda}(r) \approx 1$: Die Wahrscheinlichkeit ist für beide Fälle genauso hoch,
- $\hat{\lambda}(r) < 1$: Es ist wahrscheinlicher, daß sich a und b auf verschiedene Realwelt-Objekte beziehen.

[10] Es sei bemerkt, daß zur Schätzung der Wahrscheinlichkeiten für die nicht-dubletten Paare gemäß (3.6b) sich zumeist eine beliebig große Lernstichprobe gewinnen läßt.

Diesen drei Fällen entsprechen drei mögliche Entscheidungen: Entweder als *Duplikat* oder *Nicht-Duplikat* akzeptieren oder als *potentielles Duplikat* zu belassen, so daß es noch einer weiteren Prüfung bedarf. Diese Vorgehensweise entspricht einem zweifachen bzw. zweiseitigen monotonen Likelihood-Quotienten-Test (für statistische Testtheorie siehe z.B. [MGB74, Leh94]). Dazu stellen wir zunächst die Hypothese H_0 und die Gegenhypothese H_1 für (a, b) auf:

$$H_0 : \ (a \not\equiv b), \qquad H_1 : \ (a \equiv b). \tag{3.7}$$

Der zweiseitige monotone Likelihood-Quotienten-Test wird dann wie folgt durchgeführt:

$$\begin{aligned} \hat{\lambda}(r) > \hat{\lambda}_0 : &\quad H_0 \text{ wird zum Niveau } \alpha_0 \text{ verworfen} \\ \hat{\lambda}(r) < \hat{\lambda}_1 : &\quad H_1 \text{ wird zum Niveau } \alpha_1 \text{ verworfen} \\ \text{sonst} : &\quad H_0 \text{ und } H_1 \text{ können nicht verworfen werden} \end{aligned}$$

wobei die Grenzen $\hat{\lambda}_0, \hat{\lambda}_1$ mit $0 < \hat{\lambda}_1 \leq \hat{\lambda}_0 < \infty$ wie folgt bestimmt werden (vgl. [FS69], [Nei98]). Es seien $\alpha_0, \alpha_1 > 0$ beliebig, aber fest. $\hat{\lambda}_0$ ergibt sich dann als der minimale Wert für $\lambda(\cdot)$, bei dem

$$\mathbf{P}\{\, H_0 \text{ verworfen} \mid H_0 \,\} < \alpha_0$$

gilt. Entsprechend ist $\hat{\lambda}_1$ der maximale Wert für $\lambda(\cdot)$, bei dem

$$\mathbf{P}\{\, H_1 \text{ verworfen} \mid H_1 \,\} < \alpha_1$$

gilt. Um $\hat{\lambda}_0$ und $\hat{\lambda}_1$ zu bestimmen, betrachten wir die Teilmengen des Vergleichsraumes R, für die jeweils eine der beiden Hypothesen verworfen wird, die Ablehnungsbereiche $R^*(\hat{\lambda}_0), R_*(\hat{\lambda}_1) \subset R$. Aufgrund der Monotonie des Likelihood-Quotienten $\hat{\lambda}(r)$ erhalten wir für ein beliebiges λ:

$$R^*(\lambda) \ := \ \Big\{ r \in R \mid \hat{\lambda}(r) > \lambda \Big\}, \tag{3.9a}$$

$$R_*(\lambda) \ := \ \Big\{ r \in R \mid \hat{\lambda}(r) < \lambda \Big\}. \tag{3.9b}$$

Die Schwellwerte $\hat{\lambda}_0$, $\hat{\lambda}_1$ ergeben sich dann aus

$$\begin{aligned} \hat{\lambda}_0 \ &:= \ \min \ \Big(\lambda \in \mathbb{R} \,\Big|\, \hat{\mathbf{P}}\Big\{ H_0 \text{ verworfen} \mid \hat{\lambda}(r) > \lambda \Big\} < \alpha_0 \Big) \\ &= \min \ \Big(\lambda \in \mathbb{R} \,\Big|\, \sum_{r \in R^*(\lambda)} \hat{\mathbf{P}}\{\, f(a,b) = r \mid a \not\equiv b \,\} < \alpha_0 \Big) \end{aligned} \tag{3.10a}$$

$$\begin{aligned} \hat{\lambda}_1 \ &:= \ \max \ \Big(\lambda \in \mathbb{R} \,\Big|\, \hat{\mathbf{P}}\Big\{ H_1 \text{ verworfen} \mid \hat{\lambda}(r) < \lambda \Big\} < \alpha_1 \Big) \\ &= \max \ \Big(\lambda \in \mathbb{R} \,\Big|\, \sum_{r \in R_*(\lambda)} \hat{\mathbf{P}}\{\, f(a,b) = r \mid a \equiv b \,\} < \alpha_1 \Big) \end{aligned} \tag{3.10b}$$

Typischerweise ergibt sich ein Unsicherheits-Intervall $I = [\hat{\lambda}_1, \hat{\lambda}_0]$ mit $1 \in I$, für das keine der beiden Hypothesen verworfen werden kann. Das eigentliche Problem besteht bei diesem Test darin, daß er eine geringe Güte hat, d.h.

die Wahrscheinlichkeit, daß keine der beiden Hypothesen verworfen werden kann ist recht groß, sofern α_0 und α_1 sehr klein sind. Wir erhalten

$$\hat{\beta}_0 := \hat{\mathbf{P}}\{\, H_0 \text{ nicht verworfen} \mid \neg H_0 \,\} = \sum_{r \in R \setminus \{R^*(\hat{\lambda}_0)\}} \hat{\mathbf{P}}\{\, f(a,b) = r \mid a \equiv b \,\}$$

$$\hat{\beta}_1 := \hat{\mathbf{P}}\{\, H_1 \text{ nicht verworfen} \mid \neg H_1 \,\} = \sum_{r \in R \setminus \{R_*(\hat{\lambda}_1)\}} \hat{\mathbf{P}}\{\, f(a,b) = r \mid a \not\equiv b \,\}$$

sowie für den Fall, daß keine der beiden Hypothesen verworfen werden kann:

$$\begin{aligned} \hat{\beta}_0^* &:= \hat{\mathbf{P}}\{\, \text{keine Entscheidung} \mid \neg H_0 \,\} = \hat{\beta}_0 - \alpha_1 \\ &= \sum_{r \in R'} \hat{\mathbf{P}}\{\, f(a,b) = r \mid a \equiv b \,\} \\ \hat{\beta}_1^* &:= \hat{\mathbf{P}}\{\, \text{keine Entscheidung} \mid \neg H_1 \,\} = \hat{\beta}_1 - \alpha_0 \\ &= \sum_{r \in R'} \hat{\mathbf{P}}\{\, f(a,b) = r \mid a \not\equiv b \,\} \end{aligned}$$

mit $R' := R \setminus \{R^*(\hat{\lambda}_0) \cup R_*(\hat{\lambda}_1)\} = \left\{ r \in R \mid \hat{\lambda}_1 \leq \hat{\lambda}(r) \leq \hat{\lambda}_0 \right\}$.

3.3.2.1 Die Schätzung der Multinomialverteilung

Für den Likelihood-Quotienten-Test ist die korrekte Schätzung der Wahrscheilichkeitsverteilungen

$$\mathbf{P}\{\, f(a,b) = r \mid a \equiv b \,\}, \qquad \mathbf{P}\{\, f(a,b) = r \mid a \not\equiv b \,\} \tag{3.11}$$

notwendig.

Im Falle nominaler Werte $r \in R, R = R_1 \times \cdots \times R_d$ sind für (3.11) die Parameter zweier Multinomialverteilungen zu schätzen, der in (3.6) angegebene Schätzer basiert auf den relativen Häufigkeiten des Auftretens eines Vergleichswertes r und ist nur bei hinreichend großen Stichproben verwendbar.

Sofern weitere Informationen über die Multinomialverteilung bekannt sind, etwa das Monotonieverhalten bei ordinaler (Teil-)Skalierung einer Dimension R_i oder Abhängigkeiten oder Unabhängigkeiten von Dimensionen R_i, R_j, so läßt sich dieses Wissen ausnutzen und damit der in (3.6) angegebene Schätzer verbessern.

Schätzung unter der Unabhängigkeitsannahme

Es sei $R = R' \times R''$ eine Zerlegung von R mit $R' = R_{i_1} \times \cdots \times R_{i_l}$, $\{i_1, \ldots, i_l\} \subset \{1, \ldots, d\}$ und $R'' = R \setminus \{R'\}$. Falls die Unabhängigkeitsannahmen

$$\mathbf{P}\{\, r' \,\} = \mathbf{P}\{\, r' \mid r''_* \in R'' \,\}, \quad \mathbf{P}\{\, r'' \,\} = \mathbf{P}\{\, r'' \mid r'_* \in R' \,\} \tag{3.12}$$

für alle $r'_* \in R', r''_* \in R''$ erfüllt sind, lassen sich die Wahrscheinlichkeitsverteilungen für R' und R'' einzeln auf derselben Stichprobe schätzen und die Wahrscheinlichkeitsverteilung für R ergibt sich dann als Produkt:

$$\mathbf{P}\{\, r = (r', r'') \,\} := \mathbf{P}\{\, r' \,\} \cdot \mathbf{P}\{\, r'' \,\}. \tag{3.13}$$

Durch Ausnutzung von Unabhängigkeitsannahmen kann eine konsistente Schätzung von $\mathbf{P}\{\, r \,\}$ auch für kleinere Stichproben vorgenommen werden, beispielsweise für $d = 2$ und $|R_1| = |R_2| = 20$ sind unter der Unabhängigkeitsannahme 60 – 100 Beispiele ausreichend, während ohne Unabhängigkeitsannahme mindestens ein Stichprobenumfang von 1.200 zur Schätzung von $\mathbf{P}\{\, r \,\}$ erforderlich ist.

Die Schätzung der Multinomialverteilung mit log-linearen Modellen

Im vorangegangenen Abschnitt haben wir uns mit der Schätzung unter der Unabhängigkeitannahme von zwei *disjunkten Teilmengen* von Einflußfaktoren befaßt. In diesem Abschnitt wird die Schätzung der Multinomialverteilung unter der Annahme von Abhängigkeiten zwischen Attributen beschrieben. Dazu nutzen wir die *log-linearen Modelle*. Wir beschränken uns auf eine starke Vereinfachung in der Notation. Diese Notation erlaubt es uns, in aller Kürze auf die zur Schätzung der Multinomialverteilung notwendige Methodik einzugehen. Für weitergehende Informationen sei auf die Arbeiten [BFH75, DLS80, Rec89, Chr90, WL90, Whi90, Lau96, LN98] verwiesen.

Die Grundidee ist folgende: L sei eine Stichprobe mit d Attributen (Variablen, Dimensionen, Faktoren) $R = (R_1, \ldots, R_d)$ und jedes Attribut R_i sei nominal skaliert und habe einen endlichen Wertebereich, $|\mathrm{dom}(R_i)| = n_i < \infty$. Die Anzahl der Zellen der Multinomialverteilung ist dann durch $m = \prod_{i=1}^{d} n_i$ gegeben. Wir betrachten R als multinomial verteilte Zufallsvariable, die Verteilung läßt sich darstellen als

$$\mathbf{P}\{\, R \,\} = c \cdot \underbrace{\prod_{i=1}^{d} \phi_i(R_i)}_{\text{1-Faktor-Terme}} \cdot \underbrace{\prod_{i=1}^{d-1} \prod_{j=i+1}^{d} \phi_{ij}(R_i, R_j)}_{\text{2-Faktor-Terme}} \cdots \underbrace{\phi_{123\cdots d}(R_1, \ldots, R_d)}_{d\text{-Faktor-Term}}, \tag{3.14}$$

mit Abbildungen $\phi_i : \mathrm{dom}(R_i) \to [0,1], \phi_{ij} : \mathrm{dom}(R_i) \times \mathrm{dom}(R_j) \to [0,1], \ldots,$ $\phi_{123\cdots d} : \mathrm{dom}(R) \to [0,1]$. Logarithmieren von (3.14) liefert mit $u_\cdot = \log(\phi_\cdot)$

$$\log\left(\mathbf{P}\{\, R \,\}\right) = \tag{3.15}$$

$$u + \underbrace{\sum_{i=1}^{d} u_i}_{\substack{\text{1-Faktor-} \\ \text{-Interaktionen}}} + \underbrace{\sum_{i=1}^{d-1} \sum_{j=i+1}^{d} u_{ij}}_{\substack{\text{2-Faktor-} \\ \text{-Interaktionen}}} + \underbrace{\sum_{i=1}^{d-2} \sum_{j=i+1}^{d-1} \sum_{k=j+1}^{d} u_{ijk}}_{\substack{\text{3-Faktor-} \\ \text{-Interaktion}}} + \cdots + \underbrace{u_{123\cdots d}}_{\substack{d\text{-Faktor-} \\ \text{-Inter-} \\ \text{aktion}}},$$

Definition 3.2 (log-lineares Modell). Wir bezeichnen (3.15) als log-lineares Modell $\mathcal{M}$ einer d-dimensionalen multinomial verteilten Zufallsvariable R. $\mathcal{M}$ ist durch die in ihm enthaltenen Interaktionen bestimmt, Schreibweise $\mathcal{M} \subset \{u, u_1, \ldots, u_d, u_{11}, \ldots, u_{(d-1)d}, u_{111}, \ldots, u_{(d-2)(d-1)d}, \ldots, u_{12\cdots d}\}$, wobei nur die von Null verschiedenen Interaktionen $u_\cdot$ in $\mathcal{M}$ enthalten sind.
Mit $\mathcal{M}^{(0)}$ bezeichnen wir das saturierte log-lineare Modell, bei dem alle Terme $u_\cdot$ enthalten sind.

Ein Modell $\mathcal{M}^{(2)}$ bezeichnen wir als reduziertes Modell von $\mathcal{M}^{(1)}$, wenn $\mathcal{M}^{(1)} \neq \mathcal{M}^{(2)}$ und $\mathcal{M}^{(1)} \cap \mathcal{M}^{(2)} = \mathcal{M}^{(2)}$, d.h. wenn alle in $\mathcal{M}^{(2)}$ enthaltenen Terme auch in $\mathcal{M}^{(1)}$ enthalten sind.

Das saturierte Modell kann naturgemäß den Daten einer Stichprobe am besten angepaßt werden, jedoch ist eine große Zahl an Parametern zu schätzen, die oft den Stichprobenumfang überschreitet.[11] Daher versucht man die Zahl der zu schätzenden Parameter zu verringern, indem ein hinreichend gut angepaßtes reduziertes *log-lineares Modell* gewählt wird. Die Zahl der Parameter läßt sich dadurch verringern, daß nicht *jede* Kombination von Faktoren als Interaktion zugelassen wird, indem in einem Auswahlverfahren kleinere Modelle mit *weniger* Interaktionen bezüglich ihrer Anpassung an das saturierte Modell —respektive ihrer Anpassung an die Daten der Stichprobe— untersucht werden.

Definition 3.3 (hierarchisches Modell). Ein log-lineares Modell $\mathcal{M}$ ist *hierarchisch*, wenn für alle $l \geq 2$ gilt:

$$u_{i_1 i_2 \cdots i_l} \in \mathcal{M} \Longrightarrow u_{j_1 j_2 \cdots j_{l-1}} \in \mathcal{M} \text{ für alle } \{j_1, \ldots, j_{l-1}\} \subset \{i_1, \ldots, i_l\}\,.$$

In einem *hierarchischen log-linearen Modell* sind alle Interaktionen von Kombinationen der Faktoren, die miteinander interagieren, enthalten. Im einzelnen heißt das, daß falls in einem Modell k Faktoren $\{R_{i_1}, \ldots, R_{i_k}\}$ Interdependenzen haben, alle Kombinationen von 1 bis $(k-1)$ Faktoren aus $\{R_{i_1}, \ldots, R_{i_k}\}$ auch in dem Modell enthalten sind. Daher läßt sich ein hierarchisches Modell durch die maximalen Interaktionen von Faktoren beschreiben, die wir in eckigen Klammern zusammenfassen. Beispielsweise läßt sich ein Modell mit den Faktoren $R_1, \ldots, R_6$, in dem jeweils die Faktoren $\{R_1, R_2\}$, $\{R_2, R_3, R_4\}$ sowie $\{R_4, \ldots, R_6\}$ miteinander interagieren, kurz als [12][234][456] schreiben.

Definition 3.4 (graphisches Modell). Ein hierarchisches log-lineares Modell $\mathcal{M}$ ist *graphisch*, wenn gilt

$$\forall l \geq 3 : \left(u_{i_1 i_2 \cdots i_l} \in \mathcal{M} \Longleftrightarrow u_{j_1 j_2} \in \mathcal{M} \text{ für alle } j_1, j_2 \in \{i_1, \ldots, i_l\}\right).$$

Die *graphischen Modelle* bilden eine Teilklasse der hierarchischen log-linearen Modelle. Bei den *graphischen Modellen* betrachtet man nur die Zwei-Faktor-Interaktionen, wobei die k-Faktor-Interaktionen eines äquivalenten hierarchischen Modells im graphischen Modell enthalten sind, wenn *alle* Zwei-Faktor-Kombinationen der k Faktoren als Interaktionen auftreten. Solch eine Menge von Interaktionen bildet eine sogenannte *Clique* von Faktoren. Graphische Modelle lassen sich veranschaulichen, indem die Faktoren die Knoten und die Interaktionsterme die Kanten eines Graphen bilden. Für ein hierarchisches log-lineares Modell existiert nur dann ein äquivalentes graphisches Modell, wenn die maximalen Interaktionen der Faktoren genau den Cliquen eines graphischen Modells entsprechen. Das oben

[11] Ein saturiertes Modell enthält *mindestens* einen Parameter je Zelle und hat dadurch die beste Anpassung an die Stichprobe. Dieses Modell ist jedoch in der Regel überparametrisiert (*engl.: overfitted*), da typischerweise die Zusammenhänge in Daten weniger komplex sind — im saturierten Modell interagiert *jede* Kombination der Faktoren.

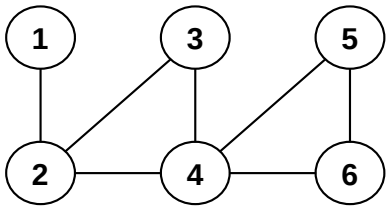


Abbildung 3.1: Das graphische Modell [12][23][24][34][45][46][56], das äquivalent zum hierarchischen log-linearen Modell [12][234][456] ist.

beschriebene Modell [12][234][456] ist äquivalent zu dem graphisches Modell [12][23][24][34][45][46][56], da es die Cliquen $\{1,2\}, \{2,3,4\}$ und $\{4,5,6\}$ enthält, vgl. Abbildung 3.1. Das hierarchische Modell [12][234][134] ist jedoch nicht graphisch, da es alle Zwei-Faktor-Interaktionen enthält, aber nicht [1234] — das Modell [1234] ist graphisch, es ist äquivalent zum graphischen Modell [12][13][14][23][24][34], da die Faktoren $\{1,2,3,4\}$ eine Clique bilden.

Die Anpassung eines Modells an eine Stichprobe wird mit einem Anpassungstest überprüft. Es läßt sich einerseits die Anpassung an das saturierte Modell prüfen (*engl.: Goodness of Fit*) und andererseits kann auch die Verbesserung der Anpassung eines erweiterten Modells gegenüber einem kleineren Modell geprüft werden, d.h. es wird getestet, ob die Anpassung eines Modells, das zusätzliche Interaktionen enthält, signifikant besser ist, als die des Modells mit weniger Interaktionen. Für beide Tests lassen sich die *Pearsons'sche Chi-Quadrat Statistik* X^2 oder die *Likelihood Ratio Statistik* G^2 verwenden. Beide Statistiken sind asymptotisch χ^2 verteilt, so daß ein Test für ein α-Niveau,[12] z.B. $\alpha = 0,05$, und (den jeweilig zu bestimmenden) Freiheitsgraden *df* (*engl.: degrees of freedom*) mit dem entsprechenden Wert $\chi^2(1-\alpha, df)$ verglichen wird. Die Nullhypothese wird zum Niveau α verworfen, wenn die Statistik X^2 bzw. G^2 den Wert $\chi^2(1-\alpha, df)$ überschreitet. Obwohl sich beide Statistiken asymptotisch (d.h. mit wachsendem Stichprobenumfang) ähnlich verhalten, wird die Likelihood Ratio Statistik oft vorgezogen. Die Präferenz dieser Statistik wird von Christensen [Chr90, S. 45] mit dem *Likelihood Prinzip* begründet: *Die Entscheidung soll abhängig gemacht werden von den relativen Werten der Likelihood Funktion.*

Zunächst beschreiben wir den Test eines Modells gegen das saturierte Modell. Es bezeichne $m^{(0)}_{i_1 i_2 \cdots i_d}$ die Anzahl der beobachteten Fälle je Zelle $i_1 i_2 \cdots i_d$ und entspricht der Schätzung der Häufigkeit durch das saturierte Modell $\mathcal{M}^{(0)} = [123 \cdots d]$, das alle k-Faktor-Interaktionen enthält (1-Faktor- bis d-Faktor-Interaktionen). $m^{(1)}_{i_1 i_2 \cdots i_d}$ bezeichne die Schätzung der Häufigkeit durch ein kleineres Modell $\mathcal{M}^{(1)}$. Dann können wir die Nullhypothese H_0 aufstellen:

$$H_0 : \; m^{(0)}_{i_1 i_2 \cdots i_d} = m^{(1)}_{i_1 i_2 \cdots i_d} \text{ für alle } i_1, i_2, \ldots, i_d \tag{3.16}$$

[12] Wir verwenden die für Tests übliche Notation des α-Niveaus und weisen darauf hin, daß es sich in diesem Zusammenhang bei α *nicht* um den Klassifikationsfehler unerkannter Duplikate handelt, der z.B. in der Formel (3.8) auf Seite 78 verwandt wird.

Für den Test bestimmen wir die *Likelihood Ratio Statistik* G^2 durch

$$G^2 = G^2(\mathcal{M}^{(1)}) = 2 \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \hat{m}^{(1)}_{i_1 i_2 \cdots i_d} \log \left(\frac{\hat{m}^{(1)}_{i_1 i_2 \cdots i_d}}{\hat{m}^{(0)}_{i_1 i_2 \cdots i_d}} \right). \tag{3.17}$$

Damit G^2 berechenbar ist, setzen wir den logarithmischen Term = 0, falls $\hat{m}^{(1)}_{i_1 i_2 \cdots i_d} = 0$ ist. Außerdem können wir für alle leeren Zellen einen positiven Wert für die Häufigkeit setzen, wenn die verwandte Software dies erfordert, d.h. für $m^{(0)}_{i_1 i_2 \cdots i_d} = 0$ setzen wir dann $m^{(0)}_{i_1 i_2 \cdots i_d} := c$ mit $c \in (0, 1]$, z.B. $c = \frac{1}{2}$ oder $c = 1$.

Den Wert G^2 vergleichen wir mit $\chi^2(1-\alpha, df)$, wobei sich die Freiheitsgrade df aus der Zahl der frei wählbaren Parameter in $\mathcal{M}^{(1)}$ ergeben, beispielsweise erhalten wir $df\big([1][2]\cdots[d]\big) = 1 + \sum_{i=1}^{d}(n_i - 1) = (\sum_{i=1}^{d} n_i) - d + 1$ für das Modell der vollständigen Unabhängigkeit aller Faktoren, oder für ein Modell mit einigen 2-Faktor-Interaktionen $df\big([12][23][4]\cdots[d]\big) = 1 + (n_1 - 1)(n_2 - 1) + (n_2 - 1)(n_3 - 1) + \sum_{i=4}^{d}(n_i - 1)$. Die Freiheitsgrade des saturierten Modells $\mathcal{M}^{(0)} = [123\cdots d]$ ist gleich der Zahl der Parameter aller k-Faktor-Terme, $k = 1, \ldots, d$,

$$df(\mathcal{M}^{(0)}) = 1 + \underbrace{\sum_{i=1}^{d}(n_i - 1)}_{\text{1-Faktor-Terme}} + \underbrace{\sum_{i=1}^{d-1}\sum_{j=i+1}^{d}(n_i - 1)(n_j - 1)}_{\text{2-Faktor- Terme}} + \ldots + \underbrace{\prod_{i=1}^{d}(n_i - 1)}_{d\text{-Faktor-Term}}.$$

Alternativ zu dem Wert G^2 können wir die Hypothese H_0 mit der *Chi-Quadrat Statistik* X^2 testen, die bestimmt ist durch die Summe der relativen quadratischen Abweichungen beider Wahrscheinlichkeitsverteilungen (df wird wie oben bestimmt):

$$X^2 = X^2(\mathcal{M}^{(1)}) = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \frac{\big(\hat{m}^{(0)}_{i_1 i_2 \cdots i_d} - \hat{m}^{(1)}_{i_1 i_2 \cdots i_d}\big)^2}{\hat{m}^{(1)}_{i_1 i_2 \cdots i_d}}. \tag{3.18}$$

Die Bewertung eines log-linearen Modells kann dann mit einer Statistik-Software erfolgen. Wir verwenden das Programm *LEM*, das als DOS- und Windows-Version frei erhältlich ist [Ver97, Ver96].[13] Neben dem oben beschriebenen Anpassungstest liefert *LEM* auch die Schätzung der Wahrscheinlichkeitsverteilung für das gewählte log-lineare Modell.

Bei der Modellselektion schränkt man zunächst die Klasse der zulässigen Modelle ein, beispielsweise beschränkt man sich auf die Klasse der hierarchischen oder der graphischen Modelle. Innerhalb einer Klasse von Modellen sucht man dann nach einem *minimalen* Modell, das eine gute Anpassung an die Daten der Stichprobe hat *und* sich gut interpretieren läßt. Für graphische Modelle ist die Interpretation aus dem zugehörigen Graphen zu entnehmen: Es gelten *konditionale Unabhängigkeitsannahmen* für die Knoten (respektive Variablen) die *nicht* über Kanten miteinander verbunden sind. Im einzelnen heißt das, das die Wahrscheinlichkeitsverteilung eine Variable

[13] Einige Statistik-Programm-Pakete verfügen über Methoden zur Analyse log-linearer Modelle, beispielsweise *SPSS*, [SPS].

nur von den Variablen abhängt, die direkt mit ihr verbunden sind; gegeben die Werte dieser Variablen ist diese Variable unabhängig von allen anderen Variablen.

Für die Modellselektion wird ein Ausgangsmodell (*engl.: initial model*) gewählt und dieses schrittweise modifiziert. Es gibt mehrere Strategien [Chr90, S. 115ff]:

- **Forward Selection:** Start mit einem kleinen Modell, z.B. dem leeren Modell (Gleichverteilung). Schrittweises Hinzufügen je eines Terms in das Ausgangsmodell, bis sich keine signifikante Verbesserung der Anpassung mehr erreichen läßt.

- **Backward Elimination:** Start mit einem großen Modell, z.B. mit dem saturierten Modell. Schrittweises Entfernen je eines Terms aus dem Ausgangsmodell, bis sich die Anpassung signifikant verschlechtert.

- **Composite Methods:** Start mit einem beliebigen Modell. Schrittweises Hinzufügen oder Entfernen von Termen, bis eine gute Anpassung erreicht wird.

Die Wahl eines hinzuzufügenden bzw. zu entfernenden Terms wird von einem Test abhängig gemacht: Wir testen ein Modell $\mathcal{M}^{(s)}$ gegen ein reduziertes Modell $\mathcal{M}^{(r)}$, in dem (mindestens) ein Term weniger enthalten ist. Der Test ergibt sich aus dem oben beschriebenen Anpassungstest, wobei wir in der Statistik (3.17) $\mathcal{M}^{(0)} := \mathcal{M}^{(s)}$ und $\mathcal{M}^{(1)} = \mathcal{M}^{(r)}$ setzen. Die mit diesen Setzungen aus (3.17) berechnete *Likelihood-Ratio-Statistik* G^2 erhält man auch aus den Statistiken des Tests der beiden Modelle gegen das saturierte Modell, es gilt

$$G^2 = G^2(\mathcal{M}^{(s)}) - G^2(\mathcal{M}^{(r)}). \tag{3.19}$$

Die Freiheitsgrade *df* ergeben sich aus der Differenz der Freiheitsgrade beider Modelle, $df = df(\mathcal{M}^{(s)}) - df(\mathcal{M}^{(r)})$, — *df* entspricht genau der Zahl der im reduzierten Modell fehlenden Parameter. Unter der H_0 wird dann das größere Modell $\mathcal{M}^{(s)}$ verworfen, falls $G^2 > \chi^2(1-\alpha, df)$ gilt, da in diesem Fall das reduzierte Modell eine hinreichend gute Anpassung an die Daten hat.

Bemerkung 3.5. Analog zum hier verfolgten Ansatz der Modellselektion wird in der Literatur zum *Record Linkage* beschrieben, wie eine verbesserte Schätzung der Multinomialverteilung erreicht werden kann, vgl. [Win93, Win95, MR93, LR94]. Bei der Schätzung werden Interaktionen von bis zu 3 Faktoren mit berücksichtigt. Es handelt sich um einen Spezialfall der in dieser Arbeit vorgestellten expliziten Modellselektion: Die Multinomialverteilung wird für ein (hierarchisches log-lineares) Modell geschätzt, das alle Interaktionen bis zur dritten Stufe enthält.[14] Dazu wird ein modifizierter

[14] Das verwandte Modell enthält die Haupteffekte (Ein-Faktor-Interaktionen), sowie alle Zwei- und Drei-Faktor-Interaktionen und den konstanten Term.

EM-Algorithmus angewandt.[15]Dieses Vorgehen entspricht der Anpassung eines hierarchischen Modells mit allen drei-Faktor Interaktionen, beispielsweise für eine Stichprobe mit fünf Variablen dem Modell

$$\mathcal{M}^{(1)} = [123][124][125][134][135][145][234][235][245][345].$$

3.3.3 Klassifikation mit Assoziationsregeln

3.3.3.1 Assoziationsregeln

Es sei eine Lernstichprobe L von Datensatzpaaren gegeben, $L = (PairID, r_1, \ldots, r_d, Same)$. Für jedes Paar seien die Vergleichswerte $r_1, \ldots, r_d$ von d Vergleichsfunktionen berechnet und notiert, und das Attribut $Same$ hat den Wert 0, wenn ein Paar aus dubletten Datensätzen besteht und sonst den Wert 0. Wir bilden den um das Attribut erweiterten Vergleichsraum R^s, d.h. wir setzen $R_0 := Same$ und $R^s := R_0 \times R$ und erhalten damit $R^s = (R_0, R_1, \ldots, R_d)$. Aus der Lernstichprobe L lassen sich dann Assoziationsregeln ableiten, die Abhängigkeiten modellieren, die zwischen Werten $r_{i_1}, \ldots, r_{i_k}$ der Attribute $R_{i_1}, \ldots, R_{i_k}$ bestehen, $i_1, \ldots, i_k \in \{0, \ldots, d\}$.

Eine allgemeine Assoziationsregel läßt sich wie folgt formulieren:

rule : `if` *Condition* `then` *Conclusion*
`with confidence` *conf* `and support` *supp*

Die Confidence ist ein Maß für die Glaubwürdigkeit einer Regel, sie gibt an, wie hoch die Wahrscheinlichkeit ist, daß die *Conclusion* erfüllt ist, wenn die *Condition* gilt. Der Support mißt die a priori Wahrscheinlichkeit einer Regel, d.h. welche relative Häufigkeit für Datensätze zu erwarten ist, die sowohl die *Condition* als auch die *Conclusion* erfüllen. Die Confidence *conf* und der Support *supp* einer Regel sind folgendermaßen definiert:

$$conf := \mathbf{P}\{\, Conclusion \mid Condition \,\} \tag{3.20a}$$
$$supp := \mathbf{P}\{\, Condition \wedge Conclusion \,\} \tag{3.20b}$$

Wir beschränken uns auf Assoziationsregeln mit konjunktiver *Condition* und einelementiger *Conclusion*, d.h. die *Condition* hat die Form

$$Condition \in \left\{ [\bigwedge_{i=1}^{k} R_{j_i} = r_{j_i}] \mid r_{j_i} \in \mathrm{dom}(R_{j_i}) \right\}$$

für einen Teilraum $R' = (R_{j_1}, \ldots, R_{j_k})$ von R^s, $j_i \in 0, \ldots, d$, und

$$Conclusion \in \{[R_j = r_j] \mid r_j \in \mathrm{dom}(R_j)\}$$

für ein Attribut $R_j \in R^s \backslash \{R'\}$. Wir erhalten

rule : `if` $[\bigwedge_{i=1}^{k} R_{j_i} = r_{j_i}]$ `then` $[R_j = r_j]$
`with confidence` *conf* `and support` *supp*

[15]EM steht für Expectation-Maximization, vgl. Dempster et al. [DLR77].

Die Confidence $conf$ und der Support $supp$ einer Regel $rule$ lassen sich für eine Lernstichprobe L folgendermaßen schätzen, $I := \{j_i \mid i = 1, \ldots, k\}$, $j \notin I$:

$$conf(rule, L) := \frac{|\{(a,b) \in L \mid (\forall i \in I : f_i(a,b) = r_i) \wedge f_j(a,b) = r_j\}|}{|\{(a,b) \in L \mid \forall i \in I : f_i(a,b) = r_i\}|}, \quad (3.21a)$$

$$supp(rule, L) := \frac{|\{(a,b) \in L \mid \forall i \in (I \cup j) : f_i(a,b) = r_i\}|}{|L|}. \quad (3.21b)$$

Für eine Lernstichprobe L lassen sich z.B. mit dem Apriori-Algorithmus Assoziationsregeln generieren [AIS93]. Als Parameter werden übergeben: $min\_supp \in (0,1), min\_conf \in (0,1]$. min_supp und min_conf geben die untere Grenze für den Support $supp$ und die Confidence $conf$ an, die von den zu generierenden Assoziationsregeln nicht unterschritten werden sollen. Die generierten Regeln fassen wir in der Menge $Rules$ zusammen:

$$\begin{aligned} &Rules(L; min\_supp, min\_conf) \quad (3.22)\\ &\quad := \{rule \mid supp(rule, L) \geq min\_supp \wedge conf(rule, L) \geq min\_conf\}. \end{aligned}$$

Für alle Lernstichproben L gilt folgende Monotonieeigenschaft:

$$(s_1 \geq s_2 \wedge c_1 \geq c_2) \Longrightarrow Rules(L; s_1, c_1) \subset Rules(L; s_2, c_2).$$

Offensichtlich läßt sich $Rules(L; s_1, c_1)$ aus $Rules(L; s_2, c_2)$ ableiten durch Selektion der Regeln deren Confidence $\geq c_1$ und Support $\geq s_1$ ist.

3.3.3.2 Das Gewinnen einer Klassifikation aus Assoziationsregeln (AssoClass)

Um aus einer Menge von Assoziationsregeln eine Klassifikation zu gewinnen, muß für widersprüchliche Regeln bezüglich eines Entscheidungsattributes (z.B $Same$) eine Bewertungs- oder Auswahlstrategie gefunden werden, die zu einer Klassifikation führt. Wir stellen im Folgenden einen neuen Ansatz dafür vor, bei dem für jeden Datensatz eine Bewertung der für ihn zutreffenden Regeln vorgenommen wird. Eine Bewertungsstrategie muß jedoch nur angewandt werden, falls die auf einen Datensatz zutreffenden Regeln bezüglich eines Entscheidungsattributes nicht widerspruchsfrei sind.

Typischerweise werden sehr viele Regeln für eine Lernstichprobe L erzeugt, so daß ein Auswahlverfahren notwendig wird. Es gibt unterschiedliche Methoden, Regeln auszuwählen. Beispielsweise können durch die Berechnung von *Reducts* redundante Regeln gefunden und eliminiert werden (vgl. Delic et. al [DLN02b, DLN02a]):

Definition 3.6. [Redukt] Eine Regel $rule_2 \in Rules(L; s, c)$ ist *redundant*, wenn es eine Regel $rule_1 \in Rules(L; s, c)$ gibt, für die gilt:

- $conf_1 \leq conf_2$,
- $Conclusion_1 = Conclusion_2$, und
- $Condition_1 \Longrightarrow Condition_2$, d.h. $rule_1$ enthält alle Bedingungen (Attribute und Werte) aus $rule_2$.

Falls die Regel $rule_2$ redundant ist, bezeichnet man die Regel $rule_1$ als Redukt (Verkürzung) (*engl.: reduct*) von $rule_2$.

Redundante Regeln enthalten Bedingungen, deren Entfernung aus der Regel keine Verringerung der Confidence zur Folge haben—diese Bedingungen sind überflüssig. Für jede Regel mit $conf = 1$, sind alle Regeln redundant, die zusätzliche Bedingungen enthalten, da sich die Confidence nicht mehr erhöhen läßt.

Definition 3.7 (passende Regel für Vergleichswerte). Es sei $I \subset \{1, \ldots, d\}$, $j \in \{1, \ldots, d\} \setminus I$, und $r = (r_1, \ldots, r_d) \in R$.

Eine Regel mit $R_j \neq Same$, der Form `if` $[\bigwedge_{i \in I} R_i = r_i^*]$ `then` $[R_j = r_j^*]$ paßt für r (*engl.: the rule matches* r), wenn gilt:

$$\forall i \in I \cup \{j\} : \; r_i^* = r_i. \tag{3.23}$$

Sei $r_0^* \in \{0, 1\}$. Eine Regel `if` $[\bigwedge_{i \in I} R_i = r_i^*]$ `then` $[Same = r_0^*]$ paßt für r, falls gilt:

$$\forall i \in I : \; r_i^* = r_i. \tag{3.24}$$

Die für einen Vergleichswert $r \in R$ passenden Regeln fassen wir in der Menge $Rules_r$ zusammen,

$$Rules_r := \{rule \in Rules(L; s, c) \mid rule \text{ paßt für } r\}. \tag{3.25}$$

Da für die Klassifikation von Datensatzpaaren in Abhängigkeit von ihrem Vergleichswert r als dublett oder nicht-dublett das Attribut *Same* entscheidend ist, teilen wir die passenden Regeln $Rules_r$ in drei Teilmengen auf: Die Regeln bei denen das Attribut *Same* die *Conclusion* ist ($Rules_r(\rightarrow Same)$), diejenigen Regeln bei denen das Attribut *Same* in der *Condition* enthalten ist ($Rules_r(Same \rightarrow)$) und schließlich alle anderen Regeln, in denen das Attribut *Same* nicht vorkommt ($Rules_r(\neg Same)$).

Bemerkung 3.8 *(Reskalierung).* Falls die Assoziationsregeln aus einer Tabelle extrahiert wurden, bei der eine unterschiedliche Anzahl von Datensätzen $pair = (a, b)$ mit $a \equiv b$ und $a \not\equiv b$ ist in der erzeugten Regelmenge ein Ungleichgewicht der Anzahl von Regeln mit $Same(a, b) =$ 'Yes' und $Same(a, b) =$ 'No' vorhanden. Wir definieren diese Anteile durch

$$c^0 := |\{(a, b) \in L \mid a \equiv b\}| = |\{(a, b) \in L \mid Same(a, b) = \text{'Yes'}\}| \tag{3.26a}$$

$$c^1 := |\{(a, b) \in L \mid a \not\equiv b\}| = |\{(a, b) \in L \mid Same(a, b) = \text{'No'}\}| \tag{3.26b}$$

Es ist zumeist $c^0 < c^1$. Sei also $c^0 < c^1$. Um eine Ausgewogenheit der Regeln zu erreichen, bestimmen wir einen Skalierungsfaktor für den minimalen Support der Regeln mit $Same =$ 'No' durch c^1/c^0 und entfernen dann alle Regeln mit $Same =$ 'No' als Folgerung aus der Regelmenge $Rules_r(\rightarrow Same)$,[16] die den (reskalierten) minimalen Support

$$min\_supp^* = \frac{c^1}{c^0} \cdot \text{min\_supp}$$

[16] Falls $c^1 = c^0$ gilt, ist keine Reskalierung notwendig; für $c^0 > c^1$ ist entsprechend die Menge der Regeln mit $Same =$ 'Yes' als Folgerung zu reduzieren.

nicht einhalten.

Falls $min\_supp = 5\%$ für die Regelerzeugung verwandt wurde, erhalten wir somit $min\_supp^* = \frac{c^1}{c^0} \cdot 0.05$.

Die Zahl der passenden Regeln $Rules_r(\rightarrow Same)$ läßt sich verkleinern, indem man die Regeln eliminiert, deren $Condition$ eine Bedingung $[R_i = r_i]$ enthält, die als $Conclusion$ in einer Regel $rule^* \in Rules_r(\neg Same)$ enthalten ist. Das Enthaltensein der Bedingung $[R_i = r_i]$ als $Conclusion$ entspricht einer empirischen Abhängigkeit des Attributwertes r_i von R_i von anderen Attribut-Attributwert-Kombinationen. Man entfernt alle Regeln, für die solch eine Abhängigkeit gilt, sofern eine Mindestwert der Confidence von der Regel $rule^*$ erreicht wird, d.h. falls $conf(rule^*) \geq conf_0$. Das liefert uns

Definition 3.9 (Abhängigkeitsfreie Regelmenge). Es sei $Rules_r(\rightarrow Same)$, $Rules_r(\neg Same) \subset Rules(L; s, c)$ und sowie $conf_0 \in (\frac{1}{2}, 1]$, $conf_0 \geq c$. Eine Regel $rule' \in Rules_r(\rightarrow Same)$ habe die Form

$rule$: `if` $[\bigwedge_{i=1}^{k} R_{j_i} = r_{j_i}]$ `then` $[Same = r_0]$

für $r_0 \in \{0, 1\}$. Wir definieren $Rules'_r(\rightarrow Same) \subset Rules_r(\rightarrow Same)$ durch

$$Rules'_r(\rightarrow Same) := \{rule \in Rules_r(\rightarrow Same) \mid \forall\, rule^* \in Rules_r(\neg Same) \quad (3.27)$$
$$\forall i = 1, \ldots, k : conf(rule^*) \geq conf_0 \wedge Conclusion(rule^*) \neq [R_{j_i} = r_{j_i}]\}$$

Dann ist $Rules'_r(\rightarrow Same)$ frei von *empirischen funktionalen Abhängigkeiten* zwischen den Attributen $R_1, \ldots, R_d$ mit einer Confidence von $conf_0$.

Für einen Vergleichswert $r \in R$ verwenden wir die Regeln in $Rules'_r(\rightarrow Same)$ zur Klassifikation. Oft erhalten wir widersprüchliche Regeln für einen Vergleichswert r, d.h. es gibt Regeln, die $Same =$ 'No' als $Conclusion$ haben und es gibt andere Regeln, nach denen $Same =$ 'Yes' gilt. Deshalb ist eine Auswahl aus den passenden Regeln und eine Bewertung bzw. Aggregation der ausgewählten Regeln für die Definition einer Entscheidungsfunktion δ notwendig.

Definition 3.10 (Klassifikation aus Assoziationsregeln). Aus den Regeln in $Rules(\rightarrow Same)$ bilden wir eine Klassifikation $\mathcal{K} = \{\mathcal{K}_0, \mathcal{K}_1\}$, indem wir jedem Vergleichswert $r \in R$ einen Vektor $p(r) := (p_0, p_1) \in [0, 1]^2$ zuordnen. Sei (a, b) ein Datensatzpaar mit dem Vergleichswert $r \in R$. Dann definieren wir p_0 als den Zugehörigkeitwert von (a, b) in die Klasse $\mathcal{K}_0$ der als dublett klassifizierten Paare, während $p_1 := (1 - p_0)$ den Zugehörigkeitwert von (a, b) in die Klasse $\mathcal{K}_1$ nicht-dublett klassifizierter Paare bezeichnet. Die Bestimmung der Zugehörigkeitswerte p_0, p_1 erfolgt mittels einer Entscheidungsfunktion $\delta : R \rightarrow [0, 1]^2$.

Für eine Klassifikation lassen sich unterschiedliche *Kriterien* für die Auswahl von Regeln und anschließender Zusammenfassung (Aggregation) je Vergleichswert $r \in R$ anwenden. Aus einer Regelmenge lassen sich also —je nach verwandtem Kriterium— unterschiedliche Klassifikationen gewinnen, die wir mit dem Index i versehen, $\mathcal{K}^{(i)}$ für die i-te Entscheidungsfunktion $\delta^{(i)}$.

Definition 3.11 (Entscheidungsfunktion (Kriterien)). Seien $s_1 \in (0,1)$, $c_1 \in (\frac{1}{2}, 1]$ und $Rules_r \subset Rules(L; s_1, c_1)$ sei eine Menge von Regeln, die für den Vergleichswert $r \in R$ gelten, und $Rules'_r(\rightarrow Same) \subset Rules_r$ sei eine Menge von Regeln, bei denen $Same$ das Entscheidungsattribut ist, die frei von empirischen funktionalen Abhängigkeiten ist.

Die Regeln in $Rules'_r(\rightarrow Same)$ seien absteigend nach der Confidence geordnet, Regeln mit gleicher Confidence sind absteigend nach ihrem Support geordnet, falls Confidence und Support von Regeln übereinstimmen, ist die längere Regel[17] zuerst aufzuzählen, und sonst sind sie zufällig geordnet. Die Position einer Regel in der Aufzählung sei mit $position(rule)$ festzustellen, wobei wir mit $rule_j$ die j-te Regel bezeichnen, d.h. es gilt $j = position(rule_j)$.

Im Folgenden sei der Vergleichswert $r \in R$ beliebig, aber fest. Die Anzahl von Regeln mit $Same$ als Entscheidungsattribut ist $count := |Rules'_r(\rightarrow Same)|$ und die Teilmengen der positiven und negativen Regeln für das $Same$-Attribut sind gegeben durch

$$\begin{aligned} Rules_r^{POS} &:= \left\{ rules \in Rules'_r(\rightarrow Same) \mid Conclusion = [Same = \text{'Yes'}] \right\}, \\ Rules_r^{NEG} &:= \left\{ rules \in Rules'_r(\rightarrow Same) \mid Conclusion = [Same = \text{'No'}] \right\}. \end{aligned}$$

Die Entscheidungsfunktionen $\delta^{(i)}$ der Klassifikation $\mathcal{K}^{(i)}$ ist gegeben durch:

$$\delta^{(i)}(r) := \begin{pmatrix} \delta_1^{(i)} \\ \delta_2^{(i)} \end{pmatrix} = \begin{pmatrix} \delta_1^{(i)} \\ 1 - \delta_1^{(i)} \end{pmatrix}. \tag{3.28}$$

Für $Rules'_r(\rightarrow Same) \neq \emptyset$ wird die Entscheidungsfunktion $\delta_1^{(i)}$ nach dem i-ten hier aufgeführten Kriterium berechnet:[18]

1. **Rank:** Summe der inversen Ränge der positiven Regeln, normiert mit $c = \frac{count(count-1)}{2}$:

$$\delta_1^{(1)}(r) := \frac{1}{c}\Big(\sum count - position(rule) \mid rule \in Rules_r^{POS} \Big)$$

2. **Confidence:** Summe der Confidence der positiven Regeln, normiert mit $c = \sum_{j=1}^{count} conf(rule_j)$:

$$\delta_1^{(2)}(r) := \frac{1}{c}\Big(\sum conf(rule) \mid rule \in Rules_r^{POS} \Big)$$

3. **Count:** Anzahl der positiven Regeln, normiert mit $c = count$:

$$\delta_1^{(3)}(r) := \frac{1}{c} \cdot \left| \left\{ rule \in Rules_r^{POS} \right\} \right|$$

[17] Die Länge einer Regel ist durch die Anzahl der Bedingungsattribute bestimmt.

[18] Wir geben fünf Beispiele für Entscheidungsfunktionen an, es existieren weitere Möglichkeiten, beispielsweise indem man für die Regeln gleicher Länge jeweils eine Bewertung definiert und diese Bewertungen dann geeignet aggregiert.

4. **BestRules (l beste Regeln):** $l \in \mathbb{N}$ sei ein fester Parameter. Falls die ersten l Regeln der Aufzählung eine übereinstimmende *Conclusion* ([$Same$ = 'Yes'] oder [$Same$ = 'No']) haben, ergibt sich eine klare Entscheidung. Andernfalls wird das *Rang-Kriterium* angewandt, d.h. wir berechnen die Summe der Ränge der positiven Regeln, normiert mit $c = \frac{count(count-1)}{2}$:

$$\delta_1^{(4)}(r;l) := \begin{cases} 0 & rule_1, \ldots, rule_l \in Rules_r^{POS}, \\ 1 & rule_1, \ldots, rule_l \in Rules_r^{NEG}, \\ \delta_1^{(1)}(r) & \text{sonst.} \end{cases}$$

5. **NonConflicts:** Wir entscheiden nur, falls alle Regeln der Aufzählung eine übereinstimmende *Conclusion* ([$Same$ = 'Yes'] oder [$Same$ = 'No']) haben. Andernfalls wird der Wert auf 0.5 gesetzt:[19]

$$\delta_1^{(5)}(r;l) := \begin{cases} 0 & rule_1, \ldots, rule_l \in Rules_r^{POS}, \\ 1 & rule_1, \ldots, rule_l \in Rules_r^{NEG}, \\ 0.5 & \text{sonst.} \end{cases}$$

6. **DecisionRules:** $conf_0 \in (0, 1]$ sei ein Parameter, der so gewählt ist, daß die Regelmenge

$$\left\{ rule \in Rules'(\rightarrow Same) \mid conf(rule) \geq conf_0 \right\}$$

widerspruchsfrei auf L ist (das ist z.B. für $conf_0 = 1$ erfüllt).[20] Formal heißt das, daß für alle Werte $r \in R$ die von Datensätzen $pair \in L$ angenommen werden, gelten muß:

$$\begin{array}{l} \left(\nexists\, rule \in Rules_r^{POS} : conf(rule) \geq conf_0 \right) \\ \vee \\ \left(\nexists\, rule \in Rules_r^{NEG} : conf(rule) \geq conf_0 \right). \end{array}$$

Dann können wir einen Entscheidungsbaum aus den Regeln mit $conf_0 \leq conf(rule)$ erzeugen, wir setzen:

$$\delta_1^{(6)}(r;l) := \begin{cases} 0 & \exists\, rule \in Rules_r^{POS} : conf(rule) \geq conf_0, \\ 1 & \exists\, rule \in Rules_r^{NEG} : conf(rule) \geq conf_0, \\ 0.5 & \text{sonst.} \end{cases}$$

Für $Rules'_r(\rightarrow Same) = \emptyset$ ergibt sich $\delta_1^{(i)}(r) := 0.5$ für alle i, da keine passenden Regeln für den Vergleichswert $r \in R$ in $Rules'_r(\rightarrow Same)$ vorhanden sind.

[19] Diese Aggregationsmethode läßt sich aus den δ-Werten der Aggregationsmethoden **Rank** sowie **Confidence** ableiten.

[20] Genaugenommen eignet sich *jede* widerspruchsfreie Teilmenge der Regelmenge $Rules'(\rightarrow Same)$ zur Klassifikation, wir stellen hier nur eine Möglichkeit dar, solch eine Teilmenge zu erzeugen.

Definition 3.12 (Entscheidungsbaum aus Assoziationsregeln). Seien $\lambda_L, \lambda_U \in (0,1)$ mit $\lambda_L \leq \lambda_U$. Dann definieren wir die diskrete Entscheidungsfunktion $\delta_D^{(i)}$ durch

$$\delta_D^{(i)}(r) := \begin{cases} 0 & \delta_1^{(i)}(r) < \lambda_L, \\ 1 & \delta_1^{(i)}(r) \geq \lambda_U, \\ 2 & \text{sonst.} \end{cases} \tag{3.29}$$

Mit $\delta_D^{(i)}$ erhalten wir die disjunkte Klassifikation $\mathcal{K}_D^{(i)} = \{\mathcal{K}_0, \mathcal{K}_1, \mathcal{K}_2\}$, wobei die Klassen $\mathcal{K}_0$ und $\mathcal{K}_1$ für die als dublett bzw. nicht-dublett klassifizierten Paare stehen und $\mathcal{K}_2$ für die nicht (klar) entscheidbaren Paare steht. Die Klassifikation $\mathcal{K}_D^{(i)}$ stellt einen Entscheidungsbaum dar, der durch die Entscheidungsfunktion $\left\{\delta_D^{(i)}(r) \mid r \in R\right\}$ bestimmt ist.

3.3.4 Nächste-Nachbarn-Klassifikation (NN–Klassifikation)

Die Nächste-Nachbarn-Klassifikation läßt sich zur Klassifikation einer Datensatzstichprobe S verwenden, im Gegensatz zu den anderem in diesem Kapitel besprochenen Verfahren ist keine Ziehung einer Stichprobe von Datensatzpaaren P notwendig. Es handelt sich um ein nicht-überwachtes Verfahren, das auch als *NN-Clustering* bezeichnet wird. Zu einem Klassifikationsverfahren für Datensatzpaare wird es durch die Festlegung einer *maximalen Abweichung* d_L, deren Unterschreiten einer Metrik für zwei Datensätze einer Klassifikation als dublett entspricht (Die Metrik wird durch Aggregation von mehrdimensionalen metrischen Vergleichsfunktionen gebildet). Da immer in der Umgebung eines Datensatzes nach ähnlichen Datensätzen (im Sinne der Metrik) gesucht wird, bezeichnen wir diese Art der Klassifikation auch als Umgebungs-Suche.

Bemerkung 3.13. Wie wir im Folgenden sehen werden, handelt es sich bei der Umgebungs-Suche um eine Selektionsmethode, bei der Datensätze gefiltert werden, die —im Sinne der verwandten Metrik— nah zueinander sind. Obzwar man die zueinander nächsten Datensätze als Duplikate klassifizieren kann, empfiehlt sich die Kombination mit einer anderen Klassifikation, beispielsweise mit einem Entscheidungsbaum. Die Umgebungssuche ist ein Beispiel für eine Vorauswahl von Datensatzpaaren, und zwar basierend auf einem metrischen Selektor. Auf die Vorauswahl werden wir im Kapitel 5 näher eingehen.

Es sei für eine Attributmenge $Y \subset X$ der Datenbank A eine Metrik $dist : (\mathrm{dom}(Y))^2 \to [0, \infty)$ definiert und es seien $k \in \mathbb{N}_+$, $\varepsilon \in \mathbb{R}_+$. Für die Suche nach Datensätzen in der Nähe eines Punktes $y \in \mathrm{dom}(Y)$, z.B. $y = Y(a_0), a_0 \in A$, gibt es zwei Suchstrategien:

- **Suche nach den k–Nächsten Nachbarn (*engl.: k–Nearest Neighbor Search, k–NN-Search*):** Selektion von k Datensätzen $a_1, \ldots, a_k \in A$, für die gilt

  $$\max_{i=1,\ldots,k} \left(dist(y, Y(a_i))\right) \leq \min \left(dist(y, Y(a_j)) \mid a_j \in A \backslash \{a_1, \ldots, a_k\}\right).$$

  Wir definieren dann k–NN$(y) := \{a_1, \ldots, a_k\}$.

- **Umgebungs-Suche (*engl.: Neighborhood-Search*):** Auswahl von allen Datensätzen aus A, die sich in der ε-Umgebung $U_\varepsilon(y) := \{x \in \mathrm{dom}(Y) \mid dist(x,y) < \varepsilon\}$ befinden,

$$U_\varepsilon^A(y) := \{a_j \in A \mid dist(y, Y(a_j)) < \varepsilon\}\,.$$

Beide Strategien werden auch als *Ähnlichkeitssuche* bezeichnet, wobei die Ähnlichkeit zweier Datensätze $a, b \in A$ durch den Abstand $dist(a,b)$ bestimmt ist: Die Ähnlichkeit ist umso größer, je kleiner der Abstand $dist(a,b)$ ausfällt. Um mit der Ähnlichkeitssuche eine Klassifikation $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3\}$ zu gewinnen, kann man eine untere und eine obere Grenze $d_L, d_U \in \mathbb{R}_+ \cup \{\infty\}$,$d_L \leq d_U$ festlegen und erhält damit für $a, b \in A$ eine Entscheidungsregel für die Umgebungs-Suche

$$\delta^{(1)}(a,b) := \begin{cases} 1 & dist(Y(a), Y(b)) < d_L, \\ 2 & dist(Y(a), Y(b)) \geq d_U, \\ 3 & \text{sonst.} \end{cases} \tag{3.30}$$

Alternativ ergibt sich eine Entscheidungsregel für die k–NN-Suche:

$$\delta^{(2)}(a,b) := \begin{cases} 1 & b \in k\text{–NN}(Y(a)), \\ 2 & \text{sonst.} \end{cases} \tag{3.31}$$

Die k–NN-Suche läßt sich jedoch nicht als Klassifikation in dublette und nicht-dublette Datensätze verwenden, da *für jeden* Datensatz genau k Datensätze gefunden werden, unabhängig davon, wie groß der Abstand zu ihnen ist. Wir können aber beide Entscheidungsregeln zur Entscheidungsregel der NN-Klassifikation zusammenfassen:

$$\delta(a,b) := \begin{cases} 1 & b \in k\text{–NN}(Y(a)) \wedge dist(Y(a), Y(b)) < d_L, \\ 2 & b \notin k\text{–NN}(Y(a)) \vee dist(Y(a), Y(b)) \geq d_U, \\ 3 & \text{sonst.} \end{cases} \tag{3.32}$$

Mit den Werten $d_L = d_U = \infty, k = 1, \ldots, |A|$ erhält man aus (3.32) die Entscheidungsregel (3.31) der k–NN-Suche, während für $k = |A|, 0 < d_L \leq d_U < \infty$ die Entscheidungsregeln (3.32) und (3.30) zusammenfallen. Für beide Klassifikationen enthält die Klasse $\mathcal{K}_1$ die als dublett klassifizierten Paare, $\mathcal{K}_2$ die als nicht-dublett klassifizierten Paare und $\mathcal{K}_3$ die unentscheidbaren Paare (wobei $\mathcal{K}_3$ für $d_L = d_U$ in (3.30) keine Paare enthält).

Bemerkung 3.14. Aufgrund der Schwierigkeit, für Realwelt-Daten Metriken zu definieren, die klar zwischen dubletten und nicht-dubletten Paaren diskriminieren können, liefert eine NN–Klassifikation $\mathcal{K}$ oft nicht akzeptable Fehlerraten. Man kann jedoch die Klassifikation $\mathcal{K}$ mit einer Klassifikation $\mathcal{K}'$ für Datensatzpaare kombinieren, um die Korrektheit zu erhöhen (vgl. dazu die vorhergegangenen Abschnitte, z.B. Entscheidungsbaum-Verfahren, Record Linkage und Klassifikation mit Assoziationsregeln). Es besteht aber ebenso die Möglichkeit, die Klassifikation $\mathcal{K}'$ durch ein Regelsystem für Vergleichswerte von Attributen X_i aus X angegeben werde, das z.B. durch einen Experten aufgestellt wird (vgl. dazu den folgenden Abschnitt zur *Sorted*

Neighborhood Method). Die Klassifikation $\mathcal{K}$ entspricht im Falle der Kombination mit einer weiteren Klassifikation einer Vorauswahl (oder Vorklassifikation) (*engl.: preselection*) von Datensatzpaaren. Weitere Möglichkeiten der Vorauswahl werden im Abschnitt 5 auf Seite 123 vorgestellt.

Bemerkung 3.15. Die k–NN- und die Umgebungs-Suche sind für große Datenmengen sehr aufwendig. Im Extremfall müssen für eine Suche alle Datensätze $a \in A$ bezüglich des Wertes $dist(y, Y(a))$ untersucht werden. Um die Effizienz zu erhöhen, versucht man die Zahl der Lesezugriffe zu verringern, d.h. je Suche nur eine Auswahl der Elemente $a \in A$ in den Arbeitsspeicher zu laden. Für numerische, kardinal skalierte Wertebereiche gibt es eine breite Palette von Indexstrukturen, die eingesetzt werden können, falls als Metrik jeweils der Absolutbetrag der Differenz zweier Werte gewählt wird (l_q-Metriken), d.h. falls $dist(x, y) := \left(\sum_{i=1}^{d} |x_{j_i} - y_{j_i}|^q\right)^{\frac{1}{q}}$, $q \in [0, \infty)$ oder $dist(x, y) := \max\left(|x_{j_i} - y_{j_i}| : i = 1, \ldots, d\right)$ für eine Attributmenge $Y = (X_{j_1}, \ldots, X_{j_d})$. In diesem Fall ist der effiziente Lesezugriff für die Umgebungs- und k-NN–Suche durch die vom Datenbank-Management-System zur physischen Speicherung der Daten eingesetzten Indexstrukturen wie B-Trees, R*-Trees oder Bitmaps zumeist gewährleistet, so daß die zusätzliche Verwendung einer Indexstruktur in der Regel nicht notwendig ist. Bei der Verwendung von anderen (nicht-l_q) Metriken kann ein effizienter Lesezugriff für die k–NN- und die Umgebungs-Suche nicht garantiert werden, so daß auf zusätzliche, externe Indexstrukturen zurückgegriffen werden muß. Für metrisch skalierte Wertebereiche wurden ebenfalls effiziente Indexstrukturen entwickelt (z.B. der M-Tree, vgl. Ciaccia et al. [CPZ97]), diese Indexstrukturen befinden sich jedoch noch im Forschungsstadium und die Implementierung muß für eine spezifische Anwendung selbst vorgenommen werden. Auf die Besonderheiten bei der Verwendung von Indexstrukturen wird im Abschnitt 5.2 auf Seite 133 eingegangen.

In den folgenden Unterabschnitten wird die *Sorted Neighborhood Method* für das Finden von Duplikaten in Datenbanken dargestellt und es werden zwei Verallgemeinerungen dieser Methode vorgestellt.

3.3.4.1 Sorted Neighborhood Method (Methode der sortierten k-Nächsten Nachbarn)

Eine Attributmenge Y sei ein Sortierschlüssel, der aus den Attributen $X = (X_1, \ldots, X_k)$ von A ableitbar sei, $X \vdash Y$. Auf dem Wertebereich $\mathrm{dom}(Y)$ der Attributmenge Y existiere eine totale Ordnung $<_Y$. Dann können wir die Metrik $dist : (\mathrm{dom}(Y))^2 \to [0, \infty)$,

$$dist(Y(a), Y(b)) := \begin{cases} 0 & a = b, \\ 1 + |\{a' \in A : a <_Y a' <_Y b\}| & a <_Y b, \\ dist(Y(b), Y(a)) & a >_Y b. \end{cases} \tag{3.33}$$

definieren.[21] Die Metrik $dist$ entspricht dem Abstand der Datensätze bezüglich der Ordnung $<_Y$. Dann läßt sich die *Sorted Neighborhood Method* als

[21] man sieht leicht ein, daß es sich um eine Metrik handelt, die Reflexivität und Symmetrie sind konstruktionsgemäß erfüllt, während die Dreiecks-Ungleichung gilt, da durch $<_Y$ eine lineare Ordnung auf A induziert ist.

Spezialfall der k-NN-Suche mit der Metrik (3.33) auffassen.[22]

Es sei $\mathcal{K}' = \{\mathcal{K}'_1, \mathcal{K}'_2\}$ eine disjunkte Klassifikation mit den Klassen $\mathcal{K}'_1$ für dublette Paare und $\mathcal{K}'_2$ für nicht-dublette Paare. Die Klassifikation $\mathcal{K}'$ wird in der Literatur als *Equational Theory* bezeichnet (vgl. z.B. die Arbeiten von Hernandez [Her96] und Hernandez und Stolfo [HS98]), die z.B. in Form eines Regelsatzes *Rules* kodiert werden kann. Mit dieser *Equational Theory* läßt sich dann für ein gegebenes Datensatzpaar entscheiden, ob es als dublett zu klassifizieren ist, oder ob nicht.

Im Folgenden setzen wir voraus, daß die Datensätze $a_i \in A$ gemäß dem Sortierschlüssel Y geordnet sind, d.h. es gilt

$$\forall i, j = 1, \ldots, |A| : \ (i < j \Longrightarrow a_i <_Y a_j).$$

Wir verwenden die Entscheidungsregel (3.31) mit $d_L = d_U = \infty$. k ist die Zahl der Datensätze, die gleichzeitig im Arbeitsspeicher gehalten werden, k wird als *scan window size* bezeichnet. Die Bezeichnung *scan window* für die Datensätze $a_{i+1}, \ldots, a_{i+k}$, $i = 0, \ldots, (|A| - k)$, rührt daher, daß die Tabelle A sequentiell gelesen —sozusagen gescannt— wird. Dabei wird jeweils der erste Datensatz a_{i+1} aus dem *scan window* (d.h. im Arbeitsspeicher) gelöscht, und dafür wird der nächste Datensatz a_{i+k+1} gelesen, vgl. Abbildung 3.2. Der Datensatz a_{i+k+1} wird mit allen Datensätzen $a_{i+2}, \ldots, a_{i+k}$ verglichen, und diese Datensatzpaare (a_j, a_{i+k+1}), $j = i + 2, \ldots, i + k$, werden mit $\mathcal{K}'$ als dublett oder nicht-dublett klassifiziert. Folglich werden alle Datensätze $a_j \in A$, $k - 1 < j < |A| - k + 1$ mit den $(2k - 2)$ Datensätzen $a_{j-k-1}, \ldots, a_{j-1}, a_{j+1}, \ldots, a_{j+k}$ verglichen, es wird also eine $(2k - 2)$–NN-Suche für sie durchgeführt.[23] Im Anschluß an den Durchlauf wird der Transitive Abschluß $\bar{\mathcal{K}}_1$ der gefundenen dubletten Datensatzpaare $(a, b) \in \mathcal{K}_1$ gebildet,[24] und mit $\bar{\mathcal{K}}_1$ erhält man formal die Klassifikation $\bar{\mathcal{K}} = \left\{\bar{\mathcal{K}}_1, ((A \times A) \backslash \bar{\mathcal{K}}_1)\right\}$ der Datensatzpaare $(a, b) \in A \times A$ in dublette und nicht-dublette Paare.

Bemerkung 3.16 *(Aufwandsbetrachtung).* Dank der *linearen Ordnung* $<_Y$ und der Metrik (3.33) muß bei der gleichzeitigen k-NN-Suche für alle Datensätze $a \in A$ jeder Datensatz nur einmal in den Arbeitsspeicher geladen werden.[25] Es bezeichne $n = |A|$ die Anzahl der Datensätze in A. Das Sortieren der Datensätze nach dem Sortierschlüssel Y hat eine Komplexität von $O(n \log n)$. Insgesamt sind $k \cdot n$ Vergleiche von Datensätzen durchzuführen (für einen Vergleich sowie für das Laden von Datensätzen nehmen

[22] Genaugenommen fällt durch die Wahl der Metrik (3.33) die k-NN-Suche mit der Umgebungs-Suche zusammen, da wir mit $d_L = d_U = k$ aus (3.30),(3.31) die Identität der Entscheidungsfunktionen erhalten, $\delta^{(1)} = \delta^{(2)}$.

[23] Die am Anfang und Ende der Sortierung $<_Y$ verbleibenden Datensätze $a_j \in A$, $j = 1, \ldots, (k-1), (|A| - k + 1), \ldots, |A|$ werden mit *mindestens* $k - 1$ Datensätzen verglichen, es wird für sie also zumindest eine $(k-1)$–NN-Suche durchgeführt.

[24] Der Transitive Abschluß $\bar{Z}$ einer binären Relation $Z \subset A \times A$ ist definiert durch (vgl. Definitionen 4.2 und 4.3 auf Seite 104):

$$\bar{Z} := \{(a_{i_0}, a_{i_1}) \mid \exists l \in \mathbb{N}_+ : \ \{(a_{i-1}, a_i) \mid i = 1, \ldots, l\} \subset Z \wedge i_0, i_1 \in \{0, \ldots, l\}\}.$$

[25] Einzelne Daten müssen jedoch mehrmals geladen werden, wenn der verfügbare Arbeitsspeicher nicht für die Daten von k Datensätzen ausreicht.

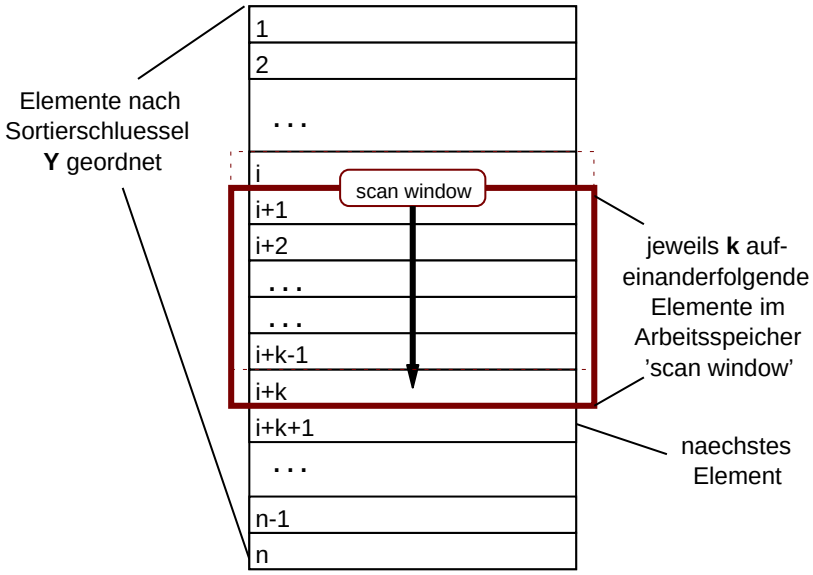


Abbildung 3.2: Sorted Neighborhood Method

wir einen konstanten Aufwand $O(1)$ an), wir erhalten eine Aufwand dafür in der Größenordnung von $O(k \cdot n)$. Es ergibt sich eine Komplexität für die *Sorted Neighborhood Method* von

$$O(n \log n) + O(k \cdot n) \overset{k \ll n}{=} O(n \log n), \tag{3.34}$$

wenn k deutlich kleiner als n gewählt wird.

Beispiel 3.17 *(Sortierschlüssel für Adreßdaten).* Wir betrachten eine Tabelle A mit Personendaten, die Attribute für die Adresse (PLZ, Ort, Straße, Hausnummer) und den Namen, Vornamen sowie das Geburtsdatum von Personen sowie eine laufende Nummer ID für die Datensätze $a \in A$ enthält:

$$A = A(\underline{ID}, PLZ, Ort, Strasse, HausNr, Name, Vornamen, GebDat)$$

Als Sortierschlüssel Y können wir beispielsweise eine phonetische Kodierung des Namens verwenden,

$$\texttt{SoundexName}(a) = \texttt{SoundexCode}(a.Name).$$

Alternativ kann man auch zusammengesetzte Sortierschlüssel Y definieren (+ bezeichne das Zusammenfügen zweier Zeichenketten), wir geben zwei Beispiele:

$\texttt{SoundexNameVorname}(\text{a}) =$
$\texttt{SoundexCode}(\text{a.Name}) + \texttt{SoundexCode}(\text{a.Vorname}),$

$\texttt{PLZNameInitialYear}(\text{a}) = \text{a.PLZ} + \texttt{Initial}(\text{a.Name}) + \texttt{Year}(\text{a.GebDat}).$

Bemerkung 3.18 *(Sortierschlüssel).* Ein Sortierschlüssel sollte so gewählt werden, daß er invariant ist für typische in den Daten vorkommende Fehler, d.h. daß der Wert des Sortierschlüssels identisch ist für zwei Elemente mit unterschiedlichen, verfälschten Daten. Z.B. sollten die Namen *Müller* und *Mueller* in zwei Datensätzen, die sonst völlig identisch seien, nicht auf einen

unterschiedlichen Sortierschlüsselwert führen.
Bei der Wahl eines Sortierschlüssels Y ist allerdings zu beachten, daß die Zahl von Elementen mit identischem Sortierschlüsselwert bezüglich der Größe k des *scan windows* nicht zu groß ist, d.h. daß

$$(2k-2) \geq \max_{y \in \mathrm{dom}(Y)} \left(\left| \{a \in A \mid Y(a) = y\} \right| \right) \tag{3.35}$$

erfüllt ist. Andernfalls würden Elemente mit identischem Sortierschlüsselwert nicht miteinander verglichen und könnten somit auch nicht als Duplikate erkannt werden.
Während die eingangs gemachte Überlegung zur Wahl von Sortierschlüsseln mit geringer Selektivität führt, zeigt die zweite Betrachtung, daß eine zu geringe Selektivität ebenfalls zu Problemen führt. Im Anwendungsfall sollte man einen Sortierschlüssel Y dergestalt zusammensetzen und gegebenenfalls modifizieren, so daß Y die Bedingung (3.35) erfüllt.

3.3.4.2 Multi-Way Sorted Neighborhood Method (Methode der mehrfach sortierten k-Nächsten Nachbarn)

Zur Verringerung der Fehlerraten der Klassifikation mit der Sorted Neighborhood Method wird diese Methode mehrfach mit unterschiedlichen Sortierschlüsseln $Y^{(1)}, \ldots, Y^{(l)}$ angewandt. Analog zum Vorgehen bei Sorted Neighborhood Method wird im Anschluß der transitive Abschluß aller als dublett klassifizierten Paare gebildet. Dadurch läßt sich zwar die Zahl gefundener Duplikate erhöhen, was zur Verringerung des α-Fehlers der Klassifikation führt, möglicherweise führt aber das mehrfache Suchen auch zu einer Erhöhung der Zahl fälschlicherweise als dublett klassifizierter Paare und damit des β-Fehlers. Die Verbesserung der Klassifikation hängt von der Güte der gewählten *Equational Theory* ab, der Klassifikation $\mathcal{K}'$ für Datensatzpaare.

Es seien l Sortierschlüssel $Y^{(1)}, \ldots, Y^{(l)}$ aus X abgeleitet, d.h. $X \vdash Y^{(i)}, i = 1, \ldots, l$. Weiter seien die Metriken $dist_i : (\mathrm{dom}(Y)^{(i)})^2 \to [0, \infty), i = 1, \ldots, l$ gemäß (3.33) gebildet, wobei $dist_i$ jeweils durch die Ordnung $<_{Y^{(i)}}$ auf A induziert sei. Dann können wir für $Y := Y^{(1)} \times \cdots \times Y^{(l)}$ die aggregierte Metrik $dist : (\mathrm{dom}(Y))^2 \to [0, \infty)$ durch

$$dist(Y(a), Y(b)) := \max_{i=1,\ldots,l} \left(dist_i(Y^{(i)}(a)), Y^{(i)}(b)) \right), \tag{3.36}$$

die sogenannte Maximums-Metrik, definieren.

Mit der gemäß (3.36) gebildeten Metrik ist die *Multi-Way Sorted Neighborhood Method* eine Umgebungs-Suche mit den Grenzen $d_L = d_U = k$, wenn die *scan window size* für alle Sortierschlüssel k beträgt. Der Vorteil dieses Vorgehens gegenüber dem mehrfachen Sortieren und mehrfachen Durchlaufen der Datenbank A liegt darin, daß sich durch die Verwendung einer geeigneten Indexstruktur (siehe dazu obige Bemerkung 3.15) die Performanz des Verfahrens deutlich erhöhen läßt.

Bemerkung 3.19 *(Aufwandsbetrachtung).* Für das sequentielle Durchführen der *Multi-Way Sorted Neighborhood Method* sind bei der Verwendung von

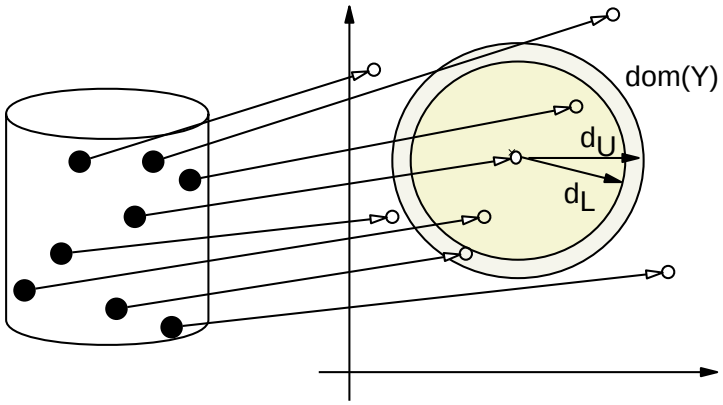


Abbildung 3.3: Umgebungs-Suche (verwandt bei der Generalized Neighborhood Method)

l Sortierschlüsseln l Sortiervorgänge für A und l Durchläufe über alle Datensätze in A notwendig. Der Aufwand je Sortierschlüssel bewegt sich nach Bemerkung 3.16 in der Größenordnung $O(n \log n)$. Das führt auf die Komplexität

$$O(l \cdot (n \log n)). \tag{3.37}$$

Durch die Verwendung einer effizienten Indexstruktur kann der Faktor l in (3.37) eliminiert werden, da der Aufbau einer Indexstruktur in der Regel den Aufwand $O(n \log n)$ und der Zugriff auf die indizierten Daten mit deutlich geringerem, in der Regel sublinearen Aufwand $O(\log_c n)$ zu bewerkstelligen ist, wobei der Parameter $c \geq 2$ von der gewählten Indexstruktur abhängt, für einen balancierten redundanzfreien binären Baum ist beispielsweise $c = 2$.

3.3.4.3 Generalized Neighborhood Method (Verallgemeinerte Methode der Nächsten Nachbarn)

Die beiden in den Abschnitten 3.3.4.1 und 3.3.4.2 dargestellten Verfahren lassen sich als Spezialfälle einer *Generalized Neighborhood Method* auffassen, bei der eine *beliebige* Metrik für den Wertebereich einer Attributmenge Y verwandt wird. Als Entscheidungsregel wird (3.32) verwandt, wobei sie durch geeignete Wahl der Werte k, d_L und d_U parametrisiert wird.

Für die *Generalized Neighborhood Method* benötigt man:

- Eine Metrik $dist : (\mathrm{dom}(Y))^2 \to [0, \infty)$ für eine Attributmenge Y von A, $X \vdash Y$,
- Werte für die Parameter k, d_L und d_U der Entscheidungsfunktion (3.32),
- Eine disjunkte Klassifikation $\mathcal{K}' = \{\mathcal{K}'_1, \mathcal{K}'_2\}$ mit den Klassen $\mathcal{K}'_1$ für dublette Paare und $\mathcal{K}'_2$ für nicht-dublette Paare (*Equational Theory for pairs*), sowie
- Einen effizienten Zugriff auf die Elemente $a \in A$ in der Umgebung eines Elements $a_0 \in A$ bezüglich der Metrik $dist$, beispielsweise durch den Einsatz einer Indexstruktur.

Bemerkung 3.20 *(Aufwandsbetrachtung).* Aus Effizienzgründen empfiehlt es sich, die k–NN-Klassifikation (3.32) mit einem relativ großen k und kleiner unterer Schranke d_L zu verwenden, damit die Anzahl der maximal mit einem Datensatz a_0 zu vergleichenden Datensätze durch k beschränkt ist und nur diejenigen Datensätze gelesen werden müssen, die bezüglich der Metrik *dist* einen Abstand von maximal d_L zu a_0 haben. Aus dem eben Gesagten läßt sich auch die Komplexität dieses Vorgehens ablesen: Lediglich im Extremfall sind analog zur *Sorted Neighborhood Method* $k \cdot n$ Vergleiche auszuführen ($n = |A|$), durch die untere Schranke werden im Normalfall aber deutlich weniger als k Vergleiche je Datensatz benötigt, wodurch sich der durchschnittliche Aufwand verringert. Diese Aufwandsbetrachtung ist jedoch nur dann korrekt, wenn eine effiziente Indexstruktur eingesetzt wird, denn *ohne* Indexstruktur muß zu a_0 für *jeden* Datensatz $a \in A$ der Abstand $dist(Y(a_0), Y(a))$ bestimmt werden, das entspricht einer Komplexität von $O(n^2)$. Wir gehen davon aus, daß eine Indexstruktur verwandt wird, die sublinearen Aufwand (bezüglich n), $O(k \log_c n) \overset{k \ll n}{=} O(\log_c n)$, für das Lesen der k–Nächsten Nachbarn zu $y \in \mathrm{dom}(Y)$ hat, mit einem $c \geq 2$ (siehe Bemerkung 3.19). Dann erhalten wir eine Komplexität in der Größenordnung des Erzeugens der Indexstruktur,

$$O(n \log n). \tag{3.38}$$

Wird die Entscheidungsfunktion (3.30) der Umgebungs-Suche gewählt (d.h. $k = \infty$ in (3.32)), läßt sich der Aufwand nur mit Zusatzinformationen abschätzen. Die Zahl der notwendigen Vergleiche je Datensatz a_0 ergibt sich dann aus der Anzahl $k(a_0) = |\{a \in A : dist(Y(a_0), Y(a)) < d_L\}|$, wobei $k(a_0)$ sehr stark von der gewählten Metrik *dist* abhängt, sowie von der unteren Grenze d_L und der Verteilung der Werte von $Y(a)$ für $a \in A$. Mit Verteilung der Werte ist an dieser Stelle der Sachverhalt gemeint, inwieweit sich die Umgebungen $U^A_{d_L}(a) := \{b \in A : dist(Y(b), Y(a)) < d_L\}$ überlappen, beispielsweise sind viele Vergleiche notwendig, falls $\sum_{a \in A} |U^A_{d_L}(a)| = \sum_{a \in A} k(a) > k_0|A|$, für ein $k_0 \approx \frac{|A|}{2}$ gilt. Sofern jedoch eine obere Schranke $k_1 > \sum_{a \in A} k(a)/|A|$ mit $k_1 \ll n = |A|$ angegeben werden kann, erhalten wir wiederum die Komplexität

$$O(n \log n). \tag{3.39}$$

Teil II

Evaluation von Objekt-identifizierungs-Verfahren

Vorbemerkung zum II. Teil

Im zweiten Teil dieser Arbeit setzen wir uns mit der methodischen Umsetzung der Objektidentifizierung auseinander. Schwerpunkte bilden hierbei die Beschreibung der Datenqualität hinsichtlich der Objektidentifizierung und Fragen des Entwurfs einer Realisierung. Im Kapitel 6 wird schließlich ein Vergleich von Klassifikations-Verfahren für verschiedenen Datenbestände durchgeführt.

Im zweiten Teil der Arbeit beschränken wir uns auf Datenbestände, die in einer universalen Relation darstellbar sind. Als Identifikationsansatz für die Evaluation der Verfahren legen wir die Wert-basierte Identifikation zugrunde.

Kapitel 4

Datenqualität

Königstochter, jüngste,
mach mir auf,

weißt du nicht, was gestern
du zu mir gesagt,
bei dem kühlen Brunnenwasser?

Königstochter, jüngste,
mach mir auf,

Aus „Froschkönig“

Even though quality cannot be defined,
you know what it is.

Robert Pirsig

Inhalt dieses Kapitels. Es werden semantische Constraints für die Qualität von Daten wie z.B. die Korrektheit einer Attributmenge oder die Anzahl der zu erwarteten Duplikate definiert, die die Komplexität von Objektidentifizierungs-Problemen beschreiben. Diese können von Experten eingeschätzt oder auf einer Stichprobe geschätzt werden.

Wir geben Kriterien für die Qualität von Daten an, mit der die Komplexität von Objektidentifizierungs-Problemen beschrieben werden kann. Der vorgestellte Ansatz ist in seinen Grundzügen in der Arbeit [NJLN03] beschrieben und wird hier detailliert behandelt.

Annahme 4.1. Wir machen folgende die Notation und den Datenzugriff vereinfachende Annahmen:

1. Wir gehen davon aus, daß die in einer Datenbank A für die Objekt-Identifizierung relevanten Daten in der Instanz einer (universalen) Relation enthalten sind. Die Tabelle, die eine Instanz einer universalen Relation ist, werden wir im folgenden ebenfalls mit A bezeichnen (da die Datenbank für uns nur aus einer einzigen Tabelle besteht, unterscheiden wir diese nicht von der Datenbank).

2. Sofern die Objekt-Identifizierung für Daten aus mehreren Datenbanken vorgenommen werden soll, führen wir die für die Objekt-Identifizierung relevanten Daten in der Tabelle A zusammen, wobei wir in der Tabelle ein zusätzliches Attribut vorhalten, in dem die Herkunft für jeden Datensatz erfaßt wird (beispielsweise durch Angabe des Namens der originären Datenbank).

3. Weiterhin versehen wir die Tabelle A mit einem Attribut ID, das für jeden Datensatz a in A einen unveränderlichen und eindeutigen (numerischen) Wert habe (das Attribut ID ist ein Surrogat-Schlüssel für A), wobei für alle Datensätze $a \in A, b \in A \backslash \{a\}$ entweder $a.ID < b.ID$ oder $a.ID > b.ID$ gelte.

4. Wir ordnen die Datensätze in A aufsteigend, so daß wir eine Aufzählung $A = (a_1, \ldots, a_N)$ erhalten, $N = |A|$. Das ID-Attribut induziert eine totale Ordnung auf A, d.h. wir können für $a, b \in A$ die Ordnung $<_{ID}$ definieren durch:[1]

$$a <_{ID} b \ :\Longleftrightarrow a.ID < b.ID. \tag{4.1}$$

   Wir schreiben $a < b$ statt $a <_{ID} b$, wenn eine Verwechslung mit anderen Ordnungen auf A ausgeschlossen ist.

4.1 Die Same-Relation

In diesem Abschnitt führen wir die Same-Relation ein. Eine Same-Relation ist eine binäre Relation über einer Tabelle A, die Paare von Elementen aus A enthält, die Duplikate sind (bzw. als solche betrachtet werden).

Definition 4.2 (Transitiver Abschluß einer binären Relation). Der geordnete transitive Abschluß (*engl.: transitive closure*) $TC(Z)$ einer binären Relation $Z \subset A \times A$ ist gegeben durch:

$$\begin{aligned} TC(Z) \ := \ & \{(a,b) \in A \times A \mid \exists l \in \mathbb{N} : \{(a_{i-1}, a_i) \mid i = 1, \ldots, l\} \subset (Z \cup Z^T) \ \wedge \\ & \wedge \exists i_0, i_1 \in \{0, \ldots, l\} : (a,b) = (a_{i_0}, a_{i_1}) \ \wedge \ a < b\} \end{aligned} \tag{4.2}$$

wobei $Z^T := \{(b,a) \in Z\}$.

Definition 4.3 (Symmetrischer Transitiver Abschluß einer binären Relation). Der symmetrische transitive Abschluß (*engl.: symmetric transitive closure*) $STC(Z)$ einer binären Relation $Z \subset A \times A$ ist gegeben durch:

$$\begin{aligned} STC(Z) &:= TC(Z) \cup (TC(Z))^T \cup \{(a,a) \in A \times A\} \\ &= \{(a,b) \in A \times A \mid \exists l \in \mathbb{N} : \{(a_{i-1}, a_i) \mid i = 1, \ldots, l\} \subset (Z \cup Z^T) \ \wedge \\ &\qquad \wedge \exists i_0, i_1 \in \{0, \ldots, l\} : (a,b) = (a_{i_0}, a_{i_1})\} \end{aligned} \tag{4.3}$$

wobei $Z^T := \{(b,a) \in Z\}$.
Offensichtlich gilt $TC(Z) = \{(a,b) \in STC(Z) \mid a < b\}$.

[1] Die auf A induzierte Ordnung durch das ID-Attribut wählen wir aus reiner Praktikabilität als Ordnung für A, wir könnten auch jede andere Ordnung auf A verwenden, die unveränderlich und eindeutig ist.

Definition 4.4 (Minimalität einer transitiven Relation). Z sei eine transitive binäre Relation über A. Dann ist eine Relation $Z^* \subset STC(Z)$ *minimal bezüglich der Transitivität*, wenn gilt

$$|Z^*| = \min_{Z' \subset STC(A)} \big(|Z'| \,:\, STC(Z') = STC(Z)\big). \tag{4.4}$$

Definition 4.5 (Same-Relation). Für die Tabelle A wird eine Same-Relation *Same* gebildet, die aus Paaren von Datensätzen $(a, b) \in A \times A$ besteht, die Informationen über *äquivalente* Realwelt-Objekte $o \in O$ enthalten, Schreibweise $a \equiv b$, d.h. es gilt

$$(a, b) \in Same \Longrightarrow a \equiv b. \tag{4.5}$$

Eine Same-Relation ist *vollständig* für A, wenn für alle $a, b \in A$ gilt:

$$a \equiv b \Longleftrightarrow (a, b) \in STC(Same). \tag{4.6}$$

Ein Element $b \in A$ bezeichnen wir als *Duplikat* des Elementes $a \in A$, wenn sich a und b auf äquivalente Realwelt-Objekte beziehen, d.h. falls $a \equiv b$ gilt.

Hierbei wird ein abstraktes Konzept der *Äquivalenz von Realwelt-Objekten* zugrundegelegt, das durch eine Äquivalenzrelation für diese Realwelt-Objekte vorgegeben werden muß. Wir nehmen an, daß diese Äquivalenzrelation mit einer festen Anzahl von *Eigenschaften* der Realwelt-Objekte beschrieben werden kann. Diese Äquivalenzrelation induziert dann eine Zerlegung von A in Äquivalenzklassen von Datensätzen, wobei jede dieser Äquivalenzklassen Datensätze enthält, die sich auf äquivalente Realwelt-Objekten beziehen.

Bemerkung 4.6. In Abhängigkeit vom Anwendungsfall kann ein Unterschied zwischen der *Äquivalenz* und der *Identität* von Realwelt-Objekten bestehen. Infolgedessen muß die Äquivalenz für jede Anwendung aus Anwendersicht festgelegt werden.
Dies sei am Beispiel eines Bibliothekskataloges erläutert: Als abstraktes Konzept der Äquivalenz von Büchern erweist es sich als sinnvoll, die *Auflage eines Buches* zu wählen. Mehrfach vorhandene Bücher einer Auflage werden in diesem Fall als äquivalent betrachtet — für Bibliotheksbenutzer ist es in der Regel unerheblich, welches der in Form und Inhalt identischen Bücher einer Auflage er ausleiht. Für die Verwaltung der ausgeliehenen Bücher ist die Unterscheidung zwischen äquivalenten und nicht identischen Büchern jedoch notwendig, um jedem Nutzer die von ihm entliehenen Bücher eindeutig zuordnen zu können.

$Same'$ sei eine vollständige Same-Relation für $A' \subset A$. Wir bilden für $a \in A'$ die Äquivalenzklasse der Duplikate von a in A', $D(a; A') \subset A'$. $D(a; A')$ enthält genau die Elemente aus A', die laut $Same'$ Duplikate von a in A' sind und ist definiert durch:

$$D(a; A') := \big\{b \in A' \mid a \equiv b\big\} = \big\{b \in A' \mid (a, b) \in STC(Same')\big\}. \tag{4.7}$$

Folgerung 4.7. *Sei $Same' \subset A' \times A'$ eine vollständige Same-Relation für $A' \subset A$ sowie $a \in A'$. Dann gilt für alle Elemente $b, b' \in D(a; A')$ offensichtlich $b \equiv b'$, sowie*

$$\forall a \in A' \, \forall b \in A' : \big(b \in D(a; A') \Longrightarrow D(a; A') = D(b; A)\big). \tag{4.8}$$

Falls $D(a; A') = \{a\}$ gilt, lassen sich aus Same$'$ keine (echten) Duplikate von a in A' ableiten.

Beweis. (4.8) folgt direkt aus (4.7), da jedes Element einer Äquivalenzklasse diese repräsentiert. □

Um eine Same-Relation *Same* frei von Redundanzen zu halten, modifizieren wir sie dergestalt, daß sie für jede verschiedene[2] Menge von Duplikaten $D(a; A') \subset A'$, $a \in A'$, $n = |D(a; A')|$ nur $(n-1)$ Paare enthält (Aus $D(a; A')$ lassen sich $\binom{n}{2}$ geordnete Paare von Datensätzen bilden).

Definition 4.8 (Optimierte Same-Relation). Aus einer Same-Relation *Same* für $A' \subset A$ bilden wir eine *optimierte Same-Relation* $Same^{opt}$ für A' wie folgt.
Zunächst entfernen wir alle mehrfach auftretenden Duplikatmengen aus der Menge von Duplikatmengen $\{D(a; A') \mid a \in A'\}$ und erhalten damit eine disjunkte Zerlegung von A' in Duplikatmengen, d.h. es gibt ein $A'' \subset A$, so daß

$$A' = \bigcup_{a \in A''} D(a; A') \tag{4.9}$$

mit $D(a; A') \cap D(b; A') = \emptyset$ für alle $a, b \in A'', a \neq b$ gilt. Die *optimierte Same-Relation* $Same^{opt}$ für A' konstruieren wir dann wie folgt:

Für alle $a \in A''$ mit $D(a; A') \neq \{a\}$ bilden wir die Indexmenge $I(a) := \{i \mid a_i \in D(a; A') \wedge a_i \neq a\}$ und setzen $i_0 := \min(i \in I(a))$. Wir speichern dann für die Duplikatmenge $D(a; A')$ die Paare (a_{i_0}, a_j), $j \in I(a) \backslash \{i_0\}$ in $Same^{opt}$.

Satz 4.9. *Same$'$ sei eine vollständige Same-Relation für $A' \subset A$.*

Dann ist die gemäß Definition 4.8 aus Same gebildete optimierte Same-Relation $Same^{opt}$ minimal, vollständig und eindeutig bestimmt. Der transitive Abschluß von $Same^{opt}$ gemäß (4.2) *vereinfacht sich zu:*

$$\begin{aligned} TC(Same^{opt}) := \{(a,b) \in A' \times A' \mid (a,b) \in Same^{opt} \vee \\ \vee (\exists a_0 \in A' : a < b \wedge (a_0, a) \in Same^{opt} \wedge (a_0, b) \in Same^{opt})\}. \end{aligned} \tag{4.10}$$

Beweis. Die Vollständigkeit folgt aus

$$\begin{aligned} \{(a,b) \mid a \equiv b\} \quad = \quad & \{(a,b) \mid (a = b) \vee (a,b) \in Same^{opt} \vee (b,a) \in Same^{opt} \vee \\ & \vee (\exists a_0 \in A : (a_0, a) \in Same^{opt} \wedge (a_0, b) \in Same^{opt})\}. \end{aligned}$$

Die Same-Relation $Same^{opt}$ ist minimal, da die Anzahl der Tupel in der Same-Relation, $|Same^{opt}|$, der Anzahl der Duplikate in A entspricht. Die Eindeutigkeit sowie die Identität (4.10) ergibt sich aus der Konstruktion der optimierten Same-Relation. □

Die eingeführten Begriffe illustrieren wir mit dem folgenden Beispiel.

[2] Zwei Mengen sind gleich, wenn alle Elemente übereinstimmen, für Mengen von Duplikaten vgl. Folgerung 4.7

Beispiel 4.10. A sei eine Tabelle mit Informationen über Personen (die Realwelt-Objekte) und die Äquivalenz von Personen sei durch die Identität der Personen bestimmt. A enthalte 10 Datensätze, die eine fortlaufende Nummer $A.ID = 1, \ldots, 10$ haben. Es sei bekannt, daß sich die Datensätze mit $A.ID = 1, 5, 7, 10$ auf eine Person beziehen ($a_1 \equiv a_1 \equiv a_7 \equiv a_{10}$), sowie daß die Datensätze mit $A.ID = 2, 3$ sich auf eine andere Person beziehen ($a_2 \equiv a_3$, $a_1 \not\equiv a_2, a_3$). Weiter sei bekannt, daß alle anderen Datensätze sich auf verschiedene Personen beziehen, d.h. die Datensätze mit $A.ID = 4, 6, 8, 9$ haben keine Duplikate in A ($a_i \not\equiv a_j$ für alle $i = 1, \ldots, 10, j = 4, 6, 8, 9$). Dann können wir eine Same-Relation $Same_0$ für A bilden,

$$Same_0 := \{(2,3), (3,2), (5,1), (5,7), (7,10), (10,5), (10,7), (10,10)\}\,.$$

$Same_0$ enthält 8 Tupel, $|Same_0| = 8$. Der transitive Abschluß $TC(Same_0)$ läßt sich aus $Same_0$ bilden, indem wir alle geordneten Paare bestimmen, die sich durch die Transitivität der Äquivalenz bzw. Identität von Personen ergeben,

$$TC(Same_0) = \{(1,5), (1,7), (1,10), (2,3), (5,7), (5,10), (7,10)\}\,.$$

Wir können auch die Menge aller Paare von Datensätzen $STC(Same_0) \subset A \times A = \{(a_i, a_j) \mid i, j = 1, \ldots, 10\}$ bilden, die sich auf eine Person beziehen,

$$\begin{aligned} STC(Same_0) \;=\; & \{(1,1), (1,5), (1,7), (1,10), (2,2), (2,3), (3,2), (3,3), (4,4), \\ & (5,1), (5,5), (5,7), (5,10), (6,6), (7,1), (7,5), (7,7), (7,10), \\ & (8,8), (9,9), (10,1), (10,5), (10,7), (10,10)\}. \end{aligned}$$

In A sind Informationen über 6 unterschiedliche Personen enthalten, so daß A lediglich $10 - 6 = 4$ Duplikate enthält, $Same_0$ ist zwar vollständig, aber nicht minimal. Wir können also eine weitere Same-Relation $Same_1 \subset Same_0$ bilden, indem wir 4 Tupel aus $Same_0$ entfernen, die überflüssig sind, um alle Paare von Duplikaten zu bilden, d.h. mit $STC(Same_0) = STC(Same_1)$. Entfernen können wir diejenigen Tupel aus $Same_0$, die dank der Transitivität aus anderen Tupeln ableitbar sind. Es genügt beispielsweise, einen der Tupel $(2,3), (3,2)$ aufzuführen. Wir können jedoch den Tupel $(5,1)$ nicht entfernen, da das Datensatzpaar $(5,1)$ sich nicht transitiv aus anderen Tupeln ableiten läßt. Wir können beispielsweise die Tupel $(2,3), (5,7), (10,7)$ und $(10,10)$ aus $Same_0$ entfernen und erhalten

$$Same_1 := \{(3,2), (5,1), (7,10), (10,5)\}\,,$$

mit $TC(Same_0) = TC(Same_1)$ und $STC(Same_0) = STC(Same_1)$. $Same_1$ ist also vollständig *und* minimal, da $|Same_1| = 4$ gilt. Da die Berechnung des transitiven Abschlusses $TC(Same_1)$ nach (4.2) aufwendiger ist als nach (4.10), bilden wir die optimierte Same-Relation $Same^{opt}$ entsprechend Definition 4.8. Zunächst bilden wir $D(a_i; A)$ für alle i: $D(a_i; A) = \{a_i\}$ für $i = 4, 6, 8, 9$, $D(a_2; A) = D(a_3; A) = \{a_2, a_3\}$ sowie für $i = 1, 5, 7, 10$ ergibt sich $D(a_i; A) = \{a_1, a_5, a_7, a_{10}\}$. Wir können $A'' \subset A$ durch $A'' := \{a_1, a_2, a_4, a_6, a_8, a_9\}$ wählen, so daß $A = \bigcup_{a \in A''} D(a; A)$ eine disjunkte Zerlegung ist. Schließlich erzeugen wir die optimierte Same-Relation aus den

Duplikatmengen $D(a_1; A) = \{a_1, a_5, a_7, a_{10}\}$ und $D(a_2; A) = \{a_2, a_3\}$,

$$Same^{opt} := \{(1,5), (1,7), (1,10), (2,3)\},$$

die minimal, vollständig *und* eindeutig bestimmt ist.

4.2 Semantische Constraints für die Qualität von Daten

Für jede Relation A einer Datenbank mit einer Attributmenge $X = (X_1, \ldots, X_k)$ können wir Metadaten ermitteln, die wir in Form einer endlichen Menge *semantischer Constraints* $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_l\}$ für A angeben. Ein *sematischer Constraint* $\mathcal{C}_j \in \mathcal{C}$ basiert auf einer reellwertigen Funktion $\mathcal{C}_j(X', A')$, die für eine Teilmenge $X' \subset X$ der Attributmenge X von A und eine Teilmenge von A und definiert ist. Ein sematischer Constraint in $\mathcal{C}$ charakterisiert ein Metadatum für die Tabelle A und wird angegeben durch

$$[\mathcal{C}_k(X', A') \diamond c], \tag{4.11}$$

mit einem Operator $\diamond \in \{<, \leq, =, \geq, >\}$ und einem Wert $c \in \mathbb{R}_{\geq 0}$. Wir vereinfachen die Notation wie folgt: Wir schreiben $[\mathcal{C}_k(A') \diamond c]$, falls $X' = X$ gilt, sowie $[\mathcal{C}_k(X') \diamond c]$ für $A' = A$ und $[\mathcal{C}_k \diamond c]$, wenn $X' = X$, $A' = A$.

Bemerkung 4.11. Eine Teilmenge $A' \subset A$ wird durch Bedingungen an Attribute aus X bestimmt, d.h. die Auswahl der Datensätze wird durch eine oder mehrere konjunktiv/disjunktiv verknüpfte Bedingungen der Art

$$X_i(a) \diamond x_i$$

mit $x_i \in \mathrm{dom}(X_i)$ und einem Vergleichsoperator $\diamond \in \{<, \leq, =, >, \geq, \neq\}$ vorgenommen.

Beispiel 4.12. A sei eine Tabelle, die Datensätze aus zwei Datenbanken Q1 und Q2 sowie ein Attribut X_1 mit $\mathrm{dom}(X_1) = \{\text{'Q1'}, \text{'Q2'}\}$ enthalte, in dem die Datenherkunft erfaßt ist. Wenn der Wert von X_1 immer angegeben ist —d.h. falls $\forall a \in A: \; X_1(a) \neq \perp$ gilt— können wir die Teilmengen von Datensätzen aus A auswählen, die aus einer der Quellen stammen,

$$A' := \{a \in A \mid X_1(a) = \text{'Q1'}\}, \tag{4.12a}$$
$$A'' := \{a \in A \mid X_1(a) = \text{'Q2'}\}, \tag{4.12b}$$

und wir erhalten eine disjunkte Zerlegung von A, $A = A' \cup A''$ mit $A' \cap A'' = \emptyset$.

Wir unterscheiden bei den semantischen Constraints zwischen *Komplexitätsmaßen*, die direkt aus A ermittelt werden können und *Charakteristika*, die durch zusätzliches Wissen, beispielsweise durch die Einschätzung von Experten gewonnen werden können.

Wir führen jetzt ein Beispiel ein, an dem wir die semantischen Constraints verdeutlichen werden.

Beispiel 4.13 *(BOA — Berliner Wohnungssuchmaschine*[3]*).* Wir betrachten eine Datenbank mit einer Tabelle A, die Berliner Wohnungsanzeigen aus zwei Online-Datenbanken enthält. Die Anzeigen wurden aus den Online-Editionen der Tageszeitungen *Tagesspiegel* and *Berliner Morgenpost*[4] der Sonnabend-Ausgaben vom 18. und 25. Mai 2002 extrahiert. Aus dem *Tagesspiegel* gewannen wir 1.507 sowie 1.643 Anzeigen (die die Teilmengen $A_1, A_2 \subset A$ bilden), während die *Berliner Morgenpost* jeweils 2.962 und 3.730 Anzeigen lieferte (die die Teilmengen $A_3, A_4 \subset A$ bilden). In beiden Quellen waren die Attribute `AnzeigenText`, `Bezirk`, `Miete`, `Räume` und `Größe` für die einzelnen Wohnungsanzeigen verfügbar. Aus dem Attribut `AnzeigenText` wurden, sofern enthalten, zusätzlich die Attribute `Telefon` (für 96%), `Straße` (für 66%), `Etage` (für 43%) und diverse Eigenschaften und Ausstattungsmerkmale abgeleitet (z.B. Altbau, Gasheizung, Gäste-WC, Terasse, Parkett, Dielen etc.). Zusätzlich zu den extrahierten Attributen ist jeder Datensatz mit einer numerischen `ID` als Primärschlüssel, sowie mit den Attributen `Quelle` für die Datenbank und `Datum` für das Erscheinungsdatum versehen.

4.2.1 Komplexitätsmaße für die Qualität von Daten

Komplexitätsmaße sind berechenbar aus einer gegebenen Tabelle A, ohne daß es zusätzlicher Metadaten über A bedarf. Wir betrachten folgende Komplexitätsmaße ($Y \subset X$ bezeichne eine Attributmenge und A' bezeichne Teilmengen von Datensätzen aus A):

- Größe von A' (z.B. Anzahl der Datensätze oder Größe in Bytes),
- Größe des Wertebereichs $\mathrm{dom}(Y)$ von Y,
- Dichte von Y für A' bezüglich einer Metrik,
- Selektivität von Y für A', sowie
- Relative Häufigkeit von fehlenden Werten von Y für A'.

Definition 4.14 (Größe). Die Größe (*engl.: size*) einer Teilmenge A' von Datensätzen aus A ist durch die Anzahl der Datensätze in A' definiert:

$$\mathrm{size}(A') := |A'| = |\left\{a \in A'\right\}|, \tag{4.13}$$

mit $\mathrm{size}(A') \in \mathbb{N}$. Sei $Y \subset X$ eine Attributmenge mit dem Wertebereich $\mathrm{dom}(Y)$. Die Größe des Wertebereichs ist dann durch die Anzahl verschiedener Werte von Y in A' gegeben,

$$\text{domain-size}(Y, A') := |\left\{y \in \mathrm{dom}(Y) | \exists a \in A' : Y(a) = y\right\}|, \tag{4.14}$$

mit $\text{domain-size}(Y, A') \in \mathbb{N}$.

Folgerung 4.15. *Für alle $Y \subset X$ und $A' \subset A$ gilt*

$$0 \leq \text{domain-size}(Y, A') \leq \mathrm{size}(A') < \infty.$$

[3]Das Akronym *BOA* steht für *Berlin Online Apartment Ads*.
[4]`www.tagesspiegel.de` und `www.mopo-immo.de`.

Beweis. domain-size(Y, A') ist durch size(A') beschränkt, da ein Element $a \in A'$ höchstens einen Wert für Y haben kann. size(A') ist endlich, da jede Datenbank endlich ist. □

Definition 4.16 (Selektivität). Die Selektivität (*engl.: selectivity*) einer Attributmenge $Y \subset X$ auf einer Teilmenge $A' \subset A$ ist definiert durch:

$$\text{selectivity}(Y, A') := \frac{|A'/\text{dom}(Y)|}{|A'|} = \frac{|\,\{\,\{a \in A' \mid Y(a) = y\}\mid y \in \text{dom}(Y)\}\,|}{|A'|}. \tag{4.15}$$

Es gilt selectivity$(Y, A') \in [0, 1]$. $A'/\text{dom}(Y)$ bezeichnet die Zerlegung von A' in Äquivalenzklassen, die induziert ist durch die Äquivalenzrelation $(A', =_Y)$. Eine Äquivalenzklasse $A'/\text{dom}(Y)$ enthält alle Datensätze $a, b \in A'$, für die $Y(a) = Y(b)$ gilt, die also übereinstimmende Werte von Y aufweisen.

Wir erhalten

Folgerung 4.17. *Für alle Y und $A' \subset A$ gilt*

$$\text{selectivity}(Y, A') = \frac{\text{domain-size}(Y, A')}{\text{size}(A')}. \tag{4.16}$$

Beweis. Einsetzen in die Definitionen unter Beachtung der Gleichmächtigkeit der beiden Mengen

$$\left\{\,\left\{a \in A' \mid Y(a) = y\right\}\mid y \in \text{dom}(Y)\right\}$$

und

$$\left\{y \in \text{dom}(Y) \middle|\, \exists a \in A' :\, Y(a) = y\right\}.$$

□

Definition 4.18 (Dichte). Sei $dist : (\text{dom}(Y))^2 \to \mathbb{R}_{\geq 0}$ eine Metrik für eine Attributmenge Y, und $\Delta > 0$. Die Δ-Dichte (*engl.: density*) einer Attributmenge $Y \subset X$ einer Teilmenge $A' \subset A$ ist definiert durch:

$$\Delta\text{-density}(Y, A') := \frac{\sum \{|\,\{a \in A' \mid dist(Y(a) - y) \leq \Delta\}\,| :\; y \in \text{dom}(Y)\}}{\text{domain-size}(Y, A') \cdot |A'|}, \tag{4.17}$$

Es gilt Δ-density$(Y, A') \in (0, 1]$ sowie

$$\Delta\text{-density}(Y, A') \geq 1/(\text{domain-size}(Y, A')).$$

Definition 4.19 (NULL-Werte). Die relative Häufigkeit fehlender Werte von Y für A' (*engl.: missing values*) einer Attributmenge $Y \subset X$ auf einer Teilmenge $A' \subset A$ ist definiert durch:

$$\text{null}(Y, A') := \frac{|\,\{a \in A' \mid \exists X_i \in Y :\, X_i(a) = \perp\}\,|}{|A'|}, \tag{4.18}$$

wobei $X_i(a) = \perp$ das Vorliegen eines fehlenden Wertes (NULL-Wertes) des Attributes X_i für einen Datensatz $a \in A'$ bezeichne. Wir haben null$(Y, A') \in [0, 1]$.

Aus den Definitionen 4.16 und 4.19 erhalten wir direkt

Folgerung 4.20. *Für alle $Y \subset X$ und alle $A' \subset A$ gilt*

$$\text{null}(Y, A') \leq 1 - \text{selectivity}(Y, A'). \tag{4.19}$$

Beweis. Man sieht

$$|\{a \in A' \mid \exists X_i \in Y : X_i(a) = \perp\}| \leq |A'| - |\{\, \{a \in A' \mid Y(a) = y\} \mid y \in \text{dom}(Y)\}|$$

leicht ein, da auf der rechten Seite je vorkommendem Wert y von Y genau eine $\{a \in A' \mid Y(a) = y\}$ existiert, und folglich je Äquivalenzklasse in mindestens einem Datensatz $a \in A'$ kein NULL-Wert vorliegt. □

4.2.2 Charakteristika für die Qualität von Daten

Charakteristika sind semantische Constraints, die mit Hilfe von zusätzlichen Informationen über die Daten (Metadaten) aufgestellt werden können. Einerseits kann Expertenwissen zum Aufstellen der semantischen Constraints verwandt werden, andererseits sind wir aber auch in der Lage, eine Schätzung der Constraints für eine Stichprobe A' aus A anzugeben, falls eine vollständige Same-Relation für A' gegeben ist.

Wir untersuchen folgende Charakteristika ($Y, Y' \subset X$ bezeichnen Attributmengen und A', A'' bezeichnen Teilmengen von Datensätzen aus A), d.h. wir können Constrains angeben für

- Semantische und approximative Schlüssel/Differenzierungs-Schlüssel Y für A',
- Die Anzahl der in A' enthaltenen Duplikate sowie für die Überlappung von A' und A'' (d.h. für wieviele Realwelt-Objekte sind Daten in *beiden* Teilmengen vorhanden),
- Die Korrektheit sowie Identifizierungsgüte von Y für A'.

4.2.2.1 Semantische Schlüssel und Differenzierungs-Schlüssel

Wissen über semantische Schlüssel und Differenzierungs-Schlüssel für $A' \subset A$ kann dazu genutzt werden, die Zahl der zu vergleichenden Datensatz-Paare aus $A \times A$ deutlich zu verringern. Die Zahl *geordneter Paare von Datensätzen*, die sich aus $A \times A$ bilden lassen, ist gegeben durch[5]

$$\sum_{i=1}^{|A|-1} i = \frac{(|A|-1)\,(|A|-2)}{2} \approx \frac{|A|^2}{2}. \tag{4.20}$$

[5]Offensichtlich lassen sich $|A|^2$ verschiedene Paare (a, b) von Datensätzen $a, b \in A$ bilden, jedoch ist es unnötig sowohl das Paar (a, b) als auch das Paar (b, a) zu bilden. Für die Objekt-Identifizierung sind desweiteren die Paare (a, a), $a \in A$ ohne Belang. Zusammenfassend läßt sich feststellen, daß es genügt, —unter Voraussetzung einer (beliebigen aber festen) Ordnung auf A— geordnete Paare (a, b) mit $a < b$ zu bilden.

Definition 4.21 (Semantischer Schlüssel). $Y \subset X$ ist ein *semantischer Schlüssel* auf $A' \subset A$, wenn Y die *Schlüsseleigenschaft* erfüllt, d.h. wenn gilt

$$\forall a, b \in A' : Y(a) = Y(b) \Longleftrightarrow a \equiv b. \tag{4.21}$$

Wenn Y ein *semantischer Schlüssel* auf A' ist, geben wir den Constraint

$$[\text{semantic-key}(Y, A') = 1]$$

an.

Bemerkung 4.22. Ein *semantischer Schlüssel* Y für A' erlaubt eine Objekt-Identifizierung für die Datensätze aus A', da sich Y dann als natürlicher Schlüssel verwenden läßt. Damit verringert sich die Zahl der zu bildenden geordneten Paare in $A \times A$ um

$$\frac{1}{2}(|A'|-1)(|A'|-2) \approx \frac{1}{2}|A'|^2,$$

da sich *alle* Datensätze in A' auf nicht-äquivalente Realwelt-Objekte beziehen.

Definition 4.23 (Semantischer Differenzierungs-Schlüssel). Sei $dist : (\text{dom}(Y))^2 \to \mathbb{R}_{\geq 0}$ eine Metrik für eine Attributmenge Y, und $\Delta > 0$. Y ist ein *semantischer Differenzierungs-Schlüssel* auf $A' \subset A$ (*engl.: differentiating key*), wenn aus einer Δ übersteigenden Abweichung der Y-Werte zweier Datensätze folgt, daß sie sich auf nicht-äquivalente Realwelt-Objekte beziehen, d.h. falls gilt

$$\forall a, b \in A' : \ dist\big(Y(a), Y(b)\big) \geq \Delta \Longrightarrow a \not\equiv b. \tag{4.22}$$

Für einen *semantischen Differenzierungs-Schlüssel* Y auf A' geben wir den Constraint

$$[\Delta\text{-diff-key}(Y, A') = 1]$$

an. Als Metrik *dist* kann für numerische Attribute beispielsweise die Maximumsmetrik oder die gewichtete Euklidische Metrik gewählt werden. Falls als Metrik die Identifikator-Funktion

$$dist_0(x, y) := \mathbb{1}_x(y) \tag{4.23}$$

und $\Delta = \frac{1}{2}$ gewählt werden, vereinfacht sich (4.22) zu

$$\forall a, b \in A' : Y(a) \neq Y(b) \Longrightarrow a \not\equiv b. \tag{4.24}$$

Im letztgenannten Fall geben wir für den *semantischen Differenzierungs-Schlüssel* Y auf A' den Constraint

$$[\text{diff-key}(Y, A') = 1]$$

an.

Ein Differenzierungs-Schlüssel Y induziert eine Zerlegung auf $A' \times A'$, wobei nur Paare (a, b) mit $dist\big(Y(a), Y(b)\big) \leq \Delta$ Duplikate sein können. Für diejenigen Paare (a, b), die $dist\big(Y(a), Y(b)\big) > \Delta$ erfüllen, können wegen der Dreiecksungleichung auch alle Datensätze a', b' aus den $\frac{\Delta}{2}$-Umgebungen von a und b keine Duplikate sein. Diese Beobachtung liefert uns

Satz 4.24. *Sei Y ein Differenzierungs-Schlüssel für $A' \subset A$ mit einer Metrik dist : $(\mathrm{dom}(Y))^2 \to \mathbb{R}_{\geq 0}$ und einem $\Delta > 0$. Die ε-Umgebung eines Elementes $a \in A$ sei für $\varepsilon > 0$ definiert durch*

$$U_\varepsilon(a) := \left\{ a' \in A \mid dist\big(Y(a), Y(a')\big) < \varepsilon \right\}. \tag{4.25}$$

Dann gilt:

$$\begin{aligned} &\forall a, a', b, b' \in A' : \\ &\quad \big(dist\big(Y(a), Y(b)\big) > \Delta \wedge a' \in U_{\frac{\Delta}{2}}(a) \wedge b' \in U_{\frac{\Delta}{2}}(b)\big) \Longrightarrow a' \not\equiv b'. \end{aligned} \tag{4.26}$$

Auf die Problematik von fehlerhaften Daten, bei denen semantische Schlüssel und Differenzierungs-Schlüssel versagen können, kommen wir im Zusammenhang mit *approximativen Schlüsseln* und *approximativen Differenzierungs-Schlüsseln* zu sprechen, vgl. Seite 120f.

Definition 4.25 (Reduktionsrate für Paare). Wenn semantische Schlüssel oder Anti-Schlüssel für Teilmengen $A' \subset A$ angegeben werden, läßt sich die Reduktionsrate für Paare aus A' berechnen,[6] es ergibt sich der Constraint

$$[\text{reduction-rate}(A') = c], \tag{4.27}$$

mit $c \in [0, 1]$. $c = 0$ steht für den Fall, daß keine semantischen (Anti-)Schlüssel angegeben werden können, während $c = 1$ gilt, wenn ein semantischer Schlüssel für ganz A' angegeben wird — im erstgenannten Fall müssen *alle* geordneten Paare und im letztgenannten Fall müssen *keine* Paare aus A' verglichen werden.

Beispiel 4.26 *(BOA — Fortsetzung).* Die Wohnungsanzeigen wurden aus vier verschiedenen Ausgaben gewonnen, die die Teilmengen $\{A_i\}_{i=1,\ldots,4}$ bilden, für die $A = \bigcup_{i=1}^{4} A_i$ und $A_i \cap A_j = \emptyset$ für $i \neq j$ gelten. Aus den $|A| = 9\,842$ Anzeigen lassen sich 48 417 720 geordnete Datensatz-Paare bilden. Diese Anzahl wollen wir durch zusätzliches Wissen reduzieren.

Wir nehmen an, daß jede der Ausgaben Duplikat-frei ist, d.h. daß keine Wohnung mehr als einmal in einer Ausgabe annonciert ist. In der Notation der semantischen Constraints bedeutet dies, daß für jedes A_i ein semantischer Schlüssel existiert $[\text{semKey}(Y, A_i)]$, $i = 1, \ldots, 4$, beispielsweise durch das Attribut $Y = \texttt{ID}$ gegeben.[7] Infolgedessen müssen keine Paare von Datensätzen $(a, b) \in A_i \times A_i$ gebildet werden, also müssen über 13 Millionen Paare nicht gebildet werden, wobei jedoch immer noch 34 604 058 Paare verbleiben. Immerhin haben wir so eine Reduktion von 28.5 % durch diese semantischen Schlüssel erreicht, wir können den Constraint

$$[\text{reduction-rate}(A) = 0.28530]$$

[6] Die Reduktionsrate läßt sich durch eine SQL-Abfrage auf A berechnen, in denen die angegebenen Schlüssel und Anti-Schlüssel als Bedingungen verwandt werden.

[7] Es ist zu beachten, daß $Y = \texttt{ID}$ *kein* semantischer Schlüssel für ganz A ist, falls Duplikate vorhanden sind! Es ist nämlich für alle $a, b \in A$ $\texttt{ID}(a) \neq \texttt{ID}(b)$, also auch für a, b mit $a \equiv b$, was eine Verletzung von (4.21) darstellt.

dafür angeben.

Da wir die Zahl der Paare weiter reduzieren wollen, nehmen wir folgende Differenzierungs-Schlüssel für A an:

$$[\text{diff-key}(\texttt{Bezirk}, A)],\ [\Delta\text{-diff-key}(\texttt{Räume}, A)]$$

mit $\Delta = 0.5$ und $[\Delta\text{-diff-key}(\texttt{Größe}, A)]$ mit $\Delta = 1.0$ und jeweils der Betragsfunktion als Abstandsmaß *dist*. Dadurch können wir die Zahl der zu bildenden geordneten Paare auf 50 958 verringern. Insgesamt haben wir durch semantische Schlüssel und Differenzierungs-Schlüssel eine Reduktionsrate von 99.895% erreicht,

$$[\text{reduction-rate}(A) = 0.99895].$$

4.2.2.2 Anzahl von Duplikaten und Überlappung von Teilmengen

Definition 4.27 (Anzahl von Duplikaten). Die Anzahl der Duplikate in A', läßt sich mit Hilfe einer vollständigen Same-Relation für A' bestimmen mit:[8]

$$\begin{aligned} \text{duplicates}(A') &:= |\left\{o \in O \mid \exists a, b \in A', a \neq b : \ a \hookrightarrow o \ \wedge \ b \hookrightarrow o\right\}| \\ &= |\left\{a \in A' \mid \exists a_0 \in A' : \ a_0 \equiv a \ \wedge \ a_0 < a\right\}|, \end{aligned} \tag{4.28}$$

wobei $a \hookrightarrow o$ den Bezug eines Datensatzes $a \in A$ auf ein Realwelt-Objekt $o \in O$ bezeichnet. Es gilt $\text{duplicates}(A') \in \mathbb{N}, 0 < \text{duplicates}(A') \leq (|A'|-1)$. Wir können semantische Constraints für die Anzahl der Duplikate mit

$$[\text{duplicates}(A') \diamond c], \tag{4.29}$$

mit einem Operator $\diamond \in \{<, \leq, =, \geq, >\}$ und einem Wert $c \in \mathbb{N}$ angeben.

Definition 4.28 (Überlappung von Teilmengen). *Same* sei eine vollständige Same-Relation für $A' \cup A''$. Die Überlappung zweier Teilmengen $A'A'' \subset A$, d.h. die Anzahl der äquivalenten Realwelt-Objekte $o \in O$, auf die sich Datensätze aus A' *und* A'' beziehen, ist wie folgt definiert:

$$\begin{aligned} \text{overlap}(A', A'') &:= |\left\{o \in O \mid \exists a \in A' \exists b \in A'' : \ (a \hookrightarrow o \ \wedge \ b \hookrightarrow o)\right\}| \\ &= |\{a \in A' \mid (\exists b \in A'' : \ a \equiv b) \ \wedge \\ &\qquad \wedge \left(\neg \exists a' \in A' : \ (a' \equiv a \ \wedge \ a' < a)\right)\}|. \end{aligned} \tag{4.30}$$

Aus den Definitionen 4.27 und 4.28 erhalten wir folgendes

Lemma 4.29. *Für alle Teilmengen $A', A'', A''' \subset A$ gelten folgende Identitäten:*

$$\begin{aligned} &\text{overlap}(A', A'') = \\ &\qquad \text{duplicates}(A' \cup A'') - \text{duplicates}(A') - \text{duplicates}(A''), \end{aligned} \tag{4.31a}$$

$$\begin{aligned} &\text{overlap}(A', A'' \cup A''') = \\ &\qquad \text{overlap}(A', A'') + \text{overlap}(A', A''') - \text{overlap}(A'', A'''), \end{aligned} \tag{4.31b}$$

$$\begin{aligned} &\text{duplicates}(A' \cup A'' \cup A'') = \\ &\qquad \text{overlap}(A', A'') + \text{overlap}(A', A''') + \text{overlap}(A'', A''') + \\ &\qquad + \text{duplicates}(A') + \text{duplicates}(A'') + \text{duplicates}(A'''). \end{aligned} \tag{4.31c}$$

[8] Es sei daran erinnert, daß wir die durch das ID-Attribut induzierte Ordnung auf A' verwenden, vgl. Annahme 4.1 auf Seite 103.

Beweis. Für (4.31a) ist nichts zu zeigen. Um (4.31b) einzusehen, haben wir zwei Fälle für einen Datensatz $a' \in A'$, für den es in $A'' \cup A'''$ Duplikate gibt, zu unterscheiden: (1) Es existieren entweder Datensätze in A'' aber nicht in A''', die sich auf ein äquivalentes Realwelt-Objekt beziehen (bzw. umgekehrt), sowie (2) Es existieren sowohl in A'' als auch in A''' Datensätze, die sich auf ein äquivalentes Realwelt-Objekt beziehen. Während im Fall (1) der Datensatz a' auf der rechten Seite von (4.31b) einmalig, entweder in $\text{overlap}(A', A'')$ oder in $\text{overlap}(A', A''')$ berücksichtigt wird, wird er im Fall (2) in *allen drei* Termen der rechten Seite von (4.31b) berücksichtigt — aufgrund des Minuszeichens vor dem dritten Term gilt die Identität (4.31b). Mehrfaches Einsetzen und Umformen von (4.31a) und (4.31b) führt schließlich zur Identität (4.31c). □

Mit dem folgenden Satz zeigen wir auf, wie die Charakteristika $\text{overlap}(\cdot,\cdot)$ und $\text{duplicates}(\cdot)$ dazu genutzt werden können, um ein Intervall für die zu erwartenden Duplikate in A aus Constraints über k Teilmengen $\{A_i\}_{i=1,\ldots,k}$, $A_i \subset A$ zu gewinnen.

Satz 4.30. *Es seien $\{A_i\}_{i=1,\ldots,k}$, $A_i \subset A$ mit $A = \bigcup_{i=1}^{k} A_i$. Weiter sei eine widerspruchsfreie Menge von semantischen Constraints $\mathcal{C}$ gegeben,*[9] *die obere und untere Schranken für die Überlappung und die Zahl der Duplikate für Teilmengen $A_i, i = 1, \ldots, k$ zusichert ($\mathcal{C}$ kann aber auch leer sein).*

Dann existieren Schranken $c_{\min}, c_{\max} \in \mathbb{N}$ mit $0 \leq c_{\min} \leq c_{\max} \leq |A| - 1$, so daß gilt

$$[c_{\min} \leq \text{duplicates}(A) \leq c_{\max}]\,. \tag{4.32}$$

Das durch die Schranken gebildete Intervall $[c_{\min}, c_{\max}]$ läßt sich aus den Constraints $\mathcal{C}$ bestimmen, wobei es umso kleiner wird, umso kleiner die durch die Constraints aus $\mathcal{C}$ zugesicherten Intervalle für den Überlappungsgrad und die Anzahl der Duplikate der Teilmengen $A_i, i = 1, \ldots, k$ ist. Die exakten Werte der Schranken $c_{\min}$ und $c_{\max}$ ergeben sich aus (4.35) *unter Berücksichtigung der Constraints* (4.33).

Beweis. Wir bilden eine Menge semantischen Constraints $\mathcal{C}'$ indem wir Constraints aus $\mathcal{C}$ der Form $[\mathcal{C}_j < c]$, $[\mathcal{C}_{j'} > c]$ und $[\mathcal{C}_{j''} = c]$ als Constraints $[\mathcal{C}_j \leq c - 1]$, $[\mathcal{C}_{j'} \geq c + 1]$ bzw. $[\mathcal{C}_{j''} \leq c]$ und $[\mathcal{C}_{j'''} \geq c]$ formulieren. Wir bilden dann die Menge semantischen Constraints $\mathcal{C}'$,

$$[\text{overlap}(A_i, A_j) \geq c_{ij}^L],\ [\text{duplicates}(A_i) \geq c_i^L]\,, \tag{4.33a}$$

$$[\text{overlap}(A_i, A_j) \leq c_{ij}^U],\ [\text{duplicates}(A_i) \leq c_i^U]\,, \tag{4.33b}$$

mit $1 \leq i < j \leq k$ und $c_i^L, c_i^U, c_{ij}^L, c_{ij}^U \in \mathbb{N}$, $c_i^L \leq c_i^U, c_{ij}^L \leq c_{ij}^U$, indem wir $\mathcal{C}'$ um die in $\mathcal{C}$ fehlenden Constraints mit Default-Constraints ergänzen. Ein Default-Constraint $\mathcal{C}_l$ wird wie folgt definiert: Für untere Schranken setzen wir $c_i^L := c_{ij}^L := 0$, sowie für die oberen Schranken setzen wir $c_i^U := |A_i| - 1$ bzw. $c_{ij}^U := \min\big((|A_i|, |A_j|\big)$.

[9] Widerspruchsfreiheit bedeutet, daß $\mathcal{C}$ keine sich ausschließenden Constraints enthält, z.B. nicht $[\mathcal{C}_i < c]$ *und* $[\mathcal{C}_i > c]$ für ein $c \in \mathbb{N}$.

Sei $\diamond \in \{\geq, \leq\}$ und seien Constraints der Form $[\text{duplicates}(A_i) \diamond c_i^{\cdot}]$ sowie $[\text{overlap}(A_i, A_j) \diamond c_{ij}^{\cdot}]$ gegeben. Die Anwendung von (4.31) liefert

$$\left[\text{duplicates}(A) \diamond \sum_{i=1}^{k} \left(\text{duplicates}(A_i) + \sum_{j=i+1}^{k} \text{overlap}(A_i, A_j) \right) \right]. \quad (4.34)$$

Wenn wir die Formel (4.34) auf die semantischen Constraints (4.33) anwenden, erhalten wir die beiden aggregierten semantischen Constraints

$$\left[\text{duplicates}(A) \quad \leq \quad \sum_{i=1}^{k} \left(c_i^U + \sum_{j=i+1}^{k} c_{ij}^U \right) \right]. \quad (4.35a)$$

$$\left[\text{duplicates}(A) \quad \geq \quad \sum_{i=1}^{k} \left(c_i^L + \sum_{j=i+1}^{k} c_{ij}^L \right) \right]. \quad (4.35b)$$

Damit sind Schranken $c_{\min}$ und $c_{\max}$ bestimmt. □

Wir wollen den Satz für unser Beispiel anwenden.

Beispiel 4.31 *(BOA — Fortsetzung).* Für die Teilmengen $A_1, \ldots, A_4$ machen wir folgende Annahmen:

- zwischen 5% und 30% Anzeigen wiederholen sich in aufeinanderfolgenden Sonnabend-Editionen derselben Zeitungen (d.h. A_1 und A_2 für den Tagesspiegel, bzw. A_3 und A_4 für die Morgenpost),
- die Überlappung zwischen den Ausgaben eines Wochenendes liegt zwischen 10% und 20% (d.h. A_1 und A_3 für den 18. Mai 2001, bzw. A_2 und A_4 für den 25. Mai 2001), sowie
- der Überlappungsgrad zwischen unterschiedlichen Ausgaben und unterschiedlichen Wochenenden betrage maximal 10% (d.h. A_1 und A_4, bzw. A_2 und A_3).

Außerdem verwenden wir wiederum die Duplikat-Freiheit der einzelnen Ausgaben A_i, d.h. $[\text{duplicates}(A_i) = 0]$, $i = 1, \ldots, 4$. Wir erhalten somit folgende semantischen Constraints:

$$\begin{aligned} [76 \leq \text{overlap}(A_1, A_2) \leq 462], \quad & [149 \leq \text{overlap}(A_3, A_4) \leq 888], \\ [151 \leq \text{overlap}(A_1, A_3) \leq 301], \quad & [165 \leq \text{overlap}(A_2, A_4) \leq 328], \\ [0 \leq \text{overlap}(A_1, A_4) \leq 150], \quad & [0 \leq \text{overlap}(A_2, A_3) \leq 164]. \end{aligned}$$

Mit (4.34) erhalten wir

$$[541 \leq \text{duplicates}(A) \leq 2.310].$$

Selbstverständlich können wir daraus nicht ableiten, welche Datensätze aus A wirklich Duplikate sind. Erfreulicherweise haben wir durch eine manuelle Suche 2.187 Duplikate gefunden. Allerdings liegt der Überlappungsgrad der beiden Ausgaben der *Berliner Morgenpost* deutlich oberhalb von 50%. Daraus läßt sich ersehen, das es sich für praktische Anwendungen durchaus schwierig gestalten kann, aufgrund von (plausiblen) Annahmen die Zahl der Duplikate zu bestimmen. Daher sollten die Annahmen stets auf einer Stichprobe geprüft werden.

4.2.2.3 Korrektheit, Identifikationsgüte und approximative Schlüssel

Im Abschnitt 4.2.2.1 haben wir das Konzept von semantischen Schlüsseln und Differenzierungs-Schlüsseln eingeführt. In diesem Abschnitt untersuchen wir, wie gut sich Attributmengen $Y \subset X$ zur Identifizierung eignen, d.h. inwiefern sie die Schlüssel- bzw. die Differenzierungs-Schlüssel-Eigenschaft erfüllen. Durch fehlerhafte und fehlende Attributwerte sind diese für Realwelt-Daten oft verletzt. Eine notwendige Bedingung für die Identifizierung mit einer Attributmenge Y ist ihre *Korrektheit*, d.h. die Forderung nach einer hohen Wahrscheinlichkeit dafür, daß ihre Werte für dublette Datensätze übereinstimmen bzw. nicht zu stark voneinander abweichen.

Notation

Zunächst führen wir die im Folgenden verwendete Notation ein.

$\mathbf{P}\{\, Ereignis \,\}$ bezeichnet die Wahrscheinlichkeit des Eintretens eines Ereignisses in einem endlichen (und mithin diskreten) Wahrscheinlichkeitsraum Ω, sowie $\hat{\mathbf{P}}\{\, Ereignis \,\}$ eine Schätzung dieser Wahrscheinlichkeit auf einer Stichprobe (zumeist von Datensätzen oder Datensatzpaaren).

$A' \subset A$ sei eine Menge von Datensätzen und $Ausdruck_i(a,b)$ seien boolsche Prädikate für $a, b \in A', i \in \mathbb{N}$. Wir betrachten den Wahrscheinlichkeitsraum $\Omega = \{\text{TRUE}, \text{FALSE}\}$ und das Eintreten der beiden Ereignisse

$$Ausdruck_i(a,b) = \text{TRUE},\ Ausdruck_i(a,b) = \text{FALSE}$$

für Elemente aus $(a,b) \in A' \times A'$.

Wir schreiben

$$\hat{\mathbf{P}}\{\, Ausdruck_1(a,b) \mid\ Ausdruck_2(a,b) \,\}$$

für die Schätzung der mit $Ausdruck_2$ bedingten Wahrscheinlichkeit des Erfülltseins von $Ausdruck_1(a,b)$ auf $A' \times A'$,

$$\mathbf{P}\big\{\, (a,b) \in A' \times A' :\ Ausdruck_1(a,b) \mid\ Ausdruck_2(a,b) \,\big\}\,.$$

Die Schätzung einer unbedingten Wahrscheinlichkeit (d.h. in der obigen Formel ist $Ausdruck_2(a,b) = \text{TRUE}$, die Bedingung ist also für alle a, b erfüllt) schreiben wir als

$$\hat{\mathbf{P}}\{\, Ausdruck_1(a,b) \,\}\,.$$

Bemerkung 4.32. Es sei gesagt, daß die in den folgenden Definitionen verwandte Identität $Y(a) = Y(b)$ bzw. die Gültigkeit von $dist\big(Y(a) - Y(b)\big) \leq \Delta$ stets impliziert, daß beide Attributwerte vorhanden sind, also $Y(a) \neq \perp$, $Y(b) \neq \perp$ gelten. Insbesondere heißt daß, das für uns $\perp \neq \perp$ gilt, fehlende Werte sind im Sinne des Wertebereiches $\mathrm{dom}(Y)$ einer Attributmenge nicht vergleichbar (Definitionsgemäß ist $\perp \notin \mathrm{dom}(Y)$ für alle $Y \subset X$). Für den Ausdruck $\hat{\mathbf{P}}\{\, Y(a) = Y(b) \,\}$ bedeutet dieses (mit $Y = (Y_1, \ldots, Y_k)$), daß er äquivalent ist zu dem Ausdruck

$$\hat{\mathbf{P}}\{\, Y(a) = Y(b), \forall i = 1, \ldots, k :\ (Y_i(a), Y_i(b) \neq \perp) \,\}\,.$$

Definition 4.33 (Korrektheit). Die *Korrektheit* (*engl.: accuracy*) einer Attributmenge $Y \subset X$ ist für $A' \subset A$ definiert durch:

$$\begin{aligned} \text{accuracy}(Y, A) &:= \hat{\mathbf{P}}\{\, Y(a) = Y(b) \mid a \equiv b \,\} \\ &= \frac{|\,\{(a,b) \in A' \times A' \mid a \equiv b \wedge Y(a) = Y(b)\}\,|}{|\,\{(a,b) \in A' \times A' \mid a \equiv b\}\,|} \end{aligned} \tag{4.36}$$

Die Δ-*Korrektheit* einer Attributmenge Y für A' ist mit einer Metrik $dist : (\text{dom}(Y))^2 \to \mathbb{R}_{\geq 0}$ und einem $\Delta \geq 0$ definiert durch:

$$\begin{aligned} \Delta\text{-accuracy}(Y) &:= \hat{\mathbf{P}}\big\{\, dist\big(Y(a) - Y(b)\big) \leq \Delta \,\big|\, a \equiv b \,\big\} \\ &= \frac{|\,\big\{(a,b) \in A' \times A' \mid a \equiv b \wedge dist\big(Y(a), Y(b)\big) \leq \Delta\big\}\,|}{|\,\{(a,b) \in A' \times A' \mid a \equiv b\}\,|} \end{aligned} \tag{4.37}$$

Die accuracy ist ein Spezialfall der Δ-accuracy für die Identifikator-Funktion $dist_0(x, y) = \mathbb{1}_x(y)$ mit $\Delta = \frac{1}{2}$.
Wir können für Attributmengen Y die semantischen Constraints

$$[\text{accuracy}(Y, A') = c],\ [\Delta\text{-accuracy}(Y, A') = c]$$

mit $c \in [0, 1]$ angeben.

Wir können die Wahrscheinlichkeit eines Paares Duplikate zu sein betrachten, wenn eine *Übereinstimmung* der Werte einer Attributmenge Y vorliegt. Diese Wahrscheinlichkeit entspricht dem bei den Assoziationsregeln eingeführten Maß der *Confidence*, die definiert ist durch $conf := \mathbf{P}\{\, Conclusion \mid Condition \,\}$ (vgl. Formel (3.20a) auf Seite 85). Wir bezeichnen dieses Maß an dieser Stelle auch als Identifizierungsgüte, da die Größe des Wertes *die Güte der Attributmenge für die Objektidentifizierung* zum Ausdruck bringt

Definition 4.34 (Confidence). Die *Confidence* oder *Identifizierungsgüte* einer Attributmenge $Y \subset X$ ist für $A' \subset A$ definiert durch:

$$\text{confidence}(Y, A') := \hat{\mathbf{P}}\{\, a \equiv b \mid Y(a) = Y(b) \,\} \tag{4.38a}$$

$$= \frac{|\,\{(a,b) \in A' \times A' : a \equiv b \wedge Y(a) = Y(b)\}\,|}{|\,\{(a,b) \in A' \times A' \mid Y(a) = Y(b)\}\,|} \tag{4.38b}$$

Die Δ-*Identifizierungsgüte* einer Attributmenge Y für A' ist mit einer Metrik $dist : (\text{dom}(Y))^2 \to \mathbb{R}_{\geq 0}$ und einem $\Delta \geq 0$ definiert durch:

$$\Delta\text{-confidence}(Y, A') := \hat{\mathbf{P}}\big\{\, a \equiv b \mid dist\big(Y(a), Y(b)\big) \leq \Delta \,\big\} \tag{4.39a}$$

$$= \frac{|\,\big\{(a,b) \in A' \times A' : a \equiv b \wedge dist\big(Y(a), Y(b)\big) \leq \Delta\big\}\,|}{|\,\big\{(a,b) \in A' \times A' : dist\big(Y(a), Y(b)\big) \leq \Delta\big\}\,|} \tag{4.39b}$$

Die confidence ist ein Spezialfall der Δ-confidence mit der Identifikator-Funktion $dist_0(x, y) = \mathbb{1}_x(y)$ als Abstandsmaß und $\Delta = \frac{1}{2}$.
Wir können für Attributmengen Y die semantischen Constraints

$$[\text{confidence}(Y, A') = c],\ [\Delta\text{-confidence}(Y, A') = c]$$

mit $c \in [0, 1]$ angeben.

Satz 4.35. *Sei $Y \subset X$ und $A' \subset A$.*

Dann gilt

$$\text{confidence}(Y, A') = \text{accuracy}(Y, A') \cdot \frac{\hat{\mathbf{P}}\{\, Y(a) = Y(b) \,\}}{\hat{\mathbf{P}}\{\, a \equiv b \,\}}. \tag{4.40}$$

Für die Δ-confidence(Y, A') und Δ-accuracy(Y, A') gilt dieses ebenso.

Beweis. Wenn wir die bedingten Wahrscheinlichkeiten auflösen gemäß der Definition

$$\mathbf{P}\{\, E_1 \mid E_2 \,\} := \frac{\mathbf{P}\{\, E_1,\, E_2 \,\}}{\mathbf{P}\{\, E_2 \,\}}$$

erhalten wir

$$\begin{aligned}
\text{accuracy}(Y, A') &= \hat{\mathbf{P}}\{\, Y(a) = Y(b) \mid a \equiv b \,\} = \frac{\hat{\mathbf{P}}\{\, Y(a) = Y(b), a \equiv b \,\}}{\hat{\mathbf{P}}\{\, a \equiv b \,\}} \\
\text{confidence}(Y, A') &= \hat{\mathbf{P}}\{\, a \equiv b \mid Y(a) = Y(b) \,\} = \frac{\hat{\mathbf{P}}\{\, Y(a) = Y(b), a \equiv b \,\}}{\hat{\mathbf{P}}\{\, Y(a) = Y(b) \,\}}
\end{aligned}$$

und also

$$\text{confidence}(Y, A') \cdot \hat{\mathbf{P}}\{\, a \equiv b \,\} = \text{accuracy}(Y, A') \cdot \hat{\mathbf{P}}\{\, Y(a) = Y(b) \,\}. \tag{4.41}$$

Umformen liefert (4.40). □

Der Faktor auf der rechten Seite von (4.40) ist zumeist größer 1, da für ein Datensatzpaar (a, b) die Wahrscheinlichkeit, daß ein Attribut-Wert übereinstimmt in der Regel größer ist als die Wahrscheinlichkeit, daß es sich um Duplikate handelt. Das liefert uns

Bemerkung 4.36. Die Identifikationsgüte confidence(Y, A') dominiert zumeist die Korrektheit accuracy(Y, A') einer Attributmenge, d.h. es gilt in der Regel

$$\text{accuracy}(Y, A') \leq \text{confidence}(Y, A').$$

Nun definieren wir approximative Schlüssel, die wir als Attribute zur Identifizierung verwenden werden. confidence-Werte deutlich oberhalb von $\frac{1}{2}$, beispielsweise confidence $= 0.8$, sind ein Indikator für eine akzeptable *Güte der Identifizierung* einer Attributmenge Y. Attributmengen mit Werten nahe 1, confidence$(Y, A') \approx 1$, haben eine sehr hohe *Güte der Identifizierung*, d.h. aus der Übereinstimmung (bzw. geringen Abweichung für die Δ-confidence) der Attributwerte von Y für zwei Datensätze folgt mit sehr hoher Wahrscheinlichkeit, daß es sich um Duplikate handelt. Für einen semantischen Schlüssel Y (vgl. Abschnitt 4.2.2.1 auf Seite 111) ergibt sich definitionsgemäß confidence$(Y, A') = 1$. Fehlerhafte Daten führen jedoch dazu, daß (i) aus $a \equiv b$ nicht zwangsläufig $Y(a) = Y(b)$ bzw. $dist(Y(a), Y(b)) \leq \Delta$ folgen muß und daher (ii) der Wert 1 für die confidence(Y, A') oft nicht erreicht wird. Aufgrund dieser Beobachtung erweitern wir das eingeführte Konzept semantischer Schlüssel auf *approximative Schlüssel*.

Definition 4.37 (Approximativer Schlüssel). Es sei $Y \subset X$ und $A' \subset A$ mit $\text{null}(A') = 0$.[10] Weiter gelte für $p, p' > \frac{1}{2}$ sowohl $\text{confidence}(Y, A') = p$ als auch $\text{accuracy}(Y, A') = p'$.

Dann bezeichnen wir Y als *approximativen Schlüssel* auf $A' \subset A$ mit Confidence Faktor $q = \min(p, p')$, bzw. als *q-approximativen Schlüssel*.
Wir geben für einen approximativen Schlüssel Y den semantischen Constraint

$$[\text{approx-key}(Y, A') = q]$$

an. Es sei ein Abstandsmaß $dist : (\text{dom}(Y))^2 \to \mathbb{R}$ und ein $\Delta > 0$ gegeben. Dann ist ein Δ-approximativen Schlüssel gegeben, falls $\Delta\text{-confidence}(Y, A') = p$ und $\Delta\text{-accuracy}(Y, A') = p'$ gilt.

Bemerkung 4.38. Umso größer die Confidence und Korrektheit eines approximativen Schlüssels Y ist, desto fehlerfreier ist die Objektidentifizierung mit Y. Aber im Gegensatz zu semantischen Schlüsseln erfüllt ein approximativer Schlüssel die Schlüsseleigenschaft „$\Longrightarrow$“,

$$Y(a) = Y(b) \Longrightarrow a \equiv b, \tag{4.42}$$

nur mit seiner Identifizierbarkeitsgüte p und die Schlüsseleigenschaft „$\Longleftarrow$“,

$$Y(a) = Y(b) \Longleftarrow a \equiv b, \tag{4.43}$$

nur mit seiner Korrektheit p'.

Wir führen ein Maß für die Güte der Unterscheidbarkeit ein, die Differenzierungsgüte.

Definition 4.39 (Differenzierungsgüte). Die *Differenzierungsgüte* (*engl.: anti-confidence*) einer Attributmenge Y für A' ist mit einer Metrik $dist : (\text{dom}(Y))^2 \to \mathbb{R}_{\geq 0}$ und einem $\Delta \geq 0$ definiert durch:

$$\Delta\text{-anti-confidence}(Y, A') := \hat{\mathbf{P}}\big\{\, a \not\equiv b \mid dist\big(Y(a), Y(b)\big) > \Delta \,\big\} \tag{4.44}$$

Definition 4.40 (Approximativer Differenzierungs-Schlüssel). Es sei $Y \subset X$ und $A' \subset A$. Weiter sei $\Delta\text{-anti-confidence}(Y, A') = p$ für $p > \frac{1}{2}$ mit einer Metrik $dist : (\text{dom}(Y))^2 \to \mathbb{R}_{\geq 0}$ und einem $\Delta \geq 0$. Dann bezeichnen wir Y als *approximativen Differenzierungs-Schlüssel* auf $A' \subset A$ mit Confidence Faktor p, bzw. als *p-approximativen Differenzierungs-Schlüssel*.
Wir geben für einen approximativen Differenzierungs-Schlüssel Y den semantischen Constraint

$$[\text{approx-diff-key}(Y, A') = p]$$

an.

Durch approximative Differenzierungs-Schlüssel kann —analog zu Differenzierungs-Schlüsseln— die Zahl der zu bildenden Paare in $A \times A$ deutlich

[10] Sei $Y = (Y_i, \ldots, Y_k)$. Dann genügt statt der Annahme $\text{null}(A') = 0$ die Annahme, daß $\forall a \in A' \subset A\, \exists i : Y_i(a) \neq \perp$.

reduziert werden. Man muß jedoch in Kauf nehmen, daß die Differenzierungs-Schlüsseleigenschaft

$$\forall a, b \in A' : \ dist(Y(a), Y(b)) > \Delta \Longrightarrow a \not\equiv b \tag{4.45}$$

für einen approximativen Differenzierungs-Schlüssel nur mit seiner Anti-Confidence p gilt. Also kann es dazu kommen, daß $dist(Y(a), Y(b)) > \Delta$ für dublette Datensätze a, b erfüllt ist, und sie deshalb nicht als Paar gebildet werden und daher auch nicht als Duplikate erkannt werden können.

4.2.2.4 Variabilität von Daten, Fehler und Spezialfälle

Fehler haben großen Einfluß auf die Qualität von Daten. Für quantitative Daten kann der Fehlerentstehungsprozeß oft durch *Gaußsches Rauschen* modelliert werden. Ein einfaches Fehlermodell kann dann durch die Angabe oder Schätzung des Erwartungswertes und der Varianz der Normalverteilung aufgestellt werden, dem (unter dieser Annahme) das *Rauschen in den Daten* unterliegt. Ausreißer in den Daten (Daten, die sehr stark vom erwarteten Wert abweichen) können dadurch mit hoher Wahrscheinlichkeit als solche aufgedeckt und korrigiert werden.

Fehler sind in Datenbanken oft schwieriger zu erkennen, da es viele qualitative Attribute (z.B. Text) gibt. Bei einigen Datensätzen fehlen einzelne Attributwerte (*engl.: missing values*), ohne daß bekannt sein muß, ob es überhaupt einen Wert dafür gibt (siehe Abschnitt 2.3.1 auf Seite 56). Falsche Daten müssen nicht unbedingt fehlerhaft sein, da es sich beispielsweise um veraltete, ungültige Daten handeln kann, die zu einem früheren Zeitpunkt korrekt gewesen sein können.

Zusätzliche Probleme erhalten wir beim Zusammenführen unterschiedlicher Datenbestände, da jeder einzelne Datenbestand eine aus der Anwendung begründete Sicht auf die Daten hat, die mit einem Datenmodell und einer Semantik verbunden ist. Daten sind oft nur insofern korrekt, als die Korrektheit für eine Anwendung relevant ist, beispielsweise ist für ein Versandhaus das Geburtsdatum oder der Vorname eines Kunden weniger bedeutsam als die Adresse.

Zusammenfassend läßt sich feststellen, daß Fehler in Datenbanken sich zumeist nicht durch Rauschen (*engl.: noise*) modellieren lassen. Vielmehr können zwei Datensätze a, b, deren Attributwerte verschieden sind, durch aus sich auf dasselbe Realwelt-Objekt beziehen, z.B. durch Verwendung von Abkürzungen und alternativen Bezeichnungen, wie *Dr. H. Mueller* und *Hans Müller, Dr.*, durch optionale Elemente wie *Hans Müller* und *Hans W. Müller* oder durch Veränderung der Werte in der Realwelt, beispielsweise könnte *Herta Schmidt Hans* heiraten und ihren Namen ändern auf *Herta Müller*.

Maschinell lassen sich jedoch einige bei der Datenerfassung und -verarbeitung entstandene Fehler erkennen, beispielsweise Tippfehler oder Konvertierungsfehler. Ebenfalls maschinell erkennen lassen sich Abkürzungen, indem man eine entsprechende Vergleichsfunktion verwendet.[11] Es sei bemerkt, daß eine Standardisierung von Sonderzeichen zu einem höheren Wert

[11] Eine Abkürzung einer Zeichenkette ist immer eine Teilfolge (*engl.: subsequence*) der Zeichenkette, d.h. die Minimum-Edit-Distance ist gleich der Längendifferenz.

der accuracy führt. Spezialfälle können zudem in einem Wörterbuch (Thesaurus) erfaßt und bei der Identifizierung berücksichtigt werden.

Weiterhin können wir mit der *Same*-Relation einer Tabelle A prüfen, inwieweit Attributwert-Variationen sich als Rauschen modellieren lassen, d.h. ob durch geeignete Wahl von Abstandsmaßen Δ-confidence und Δ-accuracy Werte nahe 1 haben (z.B. Minimum-Edit-Distance, oder Grad der Nicht-Übereinstimmung von NGrams zweier Zeichenketten). Solche Maße bieten sich dann als Vergleichsfunktionen bei der Identifizierung an.

Kapitel 5

Vorauswahl von Datensatz-Paaren (Preselection)

Ihr zahmen Täubchen, ihr Turteltäubchen,
all ihr Vöglein unter dem Himmel,
kommt und helft mir lesen,

die guten ins Töpfchen,
die schlechten ins Kröpfchen.

Aus „Aschenputtel“

Inhalt dieses Kapitels. Um die Performanz der Identifikation für große Datenmengen zu gewährleisten, wird eine Vorauswahl von Datensatz-Paaren mittels eines Selektors umgesetzt. Ein Selektor ist eine Kombination aus zulässigen Selektoren, wobei bei der Wahl eine Optimalität bezüglich der Fehler- und Selektionsrate angestrebt wird. Im letzten Abschnitt wird auf die Verwendung von Indexstrukturen zur effizienten Umsetzung von Selektoren eingegangen.

5.1 Effizienzgewinn durch eine Vorauswahl

Um die Effizienz eines Objektidentifizierungs-Verfahrens für große Datenmengen zu gewährleisten, muß die Anzahl der miteinander zu vergleichenden Datensätze reduziert werden. Dieses läßt sich durch eine Vorauswahl von Datensatzpaaren umsetzen, bei der nur diejenigen Datensätze zu Paaren zusammengefaßt werden, die als Duplikate in Frage kommen. Mit anderen Worten, wir suchen nach einer Lösung, die

1. die Zahl der zu bildenden Datensatz-Paare deutlich verringert, aber auch

2. dabei sicherstellt, daß Duplikate (bezüglich der Entscheidungsfunktion δ) als Datensatz-Paare gebildet werden.

Beispiel 5.1. Wir betrachten zwei Kunden-Datenbanken A_1 und A_2 mit jeweils 100 000 Datensätzen. Wollte man jeden Datensatz der einen Datenbank mit jedem Datensatz der anderen vergleichen, so müßte man $100\,000 \cdot 100\,000 = 10^{10}$ Vergleiche durchführen und dafür —aufgrund von Arbeitsspeicher-Restriktionen— (zehn)tausende Datensätze *mehrmals* laden. Sinn und Zweck des Optimierungsansatzes ist deshalb, die Zahl der Vergleiche zu reduzieren.

Definition 5.2 (Selektor). M sei eine Menge. $\mathcal{P}(M)$ bezeichne die *Potenzmenge* von M.[1]

Dann bezeichnen wir eine Abbildung

$$\sigma : \mathcal{P}(M) \to \mathcal{P}(M) \tag{5.1}$$

für die

$$\sigma(M') \subset M' \tag{5.2}$$

für alle $M' \in \mathcal{P}(M)$ erfüllt ist, als Selektor (Filter).

Lemma 5.3. *M sei eine Menge. $\sigma_1, \sigma_2 : \mathcal{P}(M) \to \mathcal{P}(M)$ seien Selektoren.*

Dann sind die Abbildungen

- $\sigma = \sigma_1 \cup \sigma_2$ *und*
- $\sigma = \sigma_1 \cap \sigma_2$

ebenfalls Selektoren.

Beweis. Nachrechnen unter Ausnutzen der entsprechenden Eigenschaften von Mengen. □

Beispiel 5.4 *(Blocking).* Für die Durchführung der Objektidentifizierung wird oft in praktischen Anwendungen eine Gruppierung bezüglich unterscheidender Attribute oder Attributmengen Y vorgenommen, falls die Werte zweier Datensätze für diese Attribute nicht übereinstimmen, werden sie nicht als Paar gebildet und als nicht-dublett klassifiziert (Dies wird im Zusammenhang mit Record Linkage als *Blocking* bezeichnet und geht auf die Arbeit von Newcombe [New67] zurück). Da hier nur Datensatz-Paare gebildet werden müssen, für die die Attributwerte *exakt* übereinstimmen, kann ein effizienter Zugriff auf die Daten erfolgen, beispielsweise durch einfaches Sortieren oder Verwendung eines (inversen) Indexes.

Es sei abschließend bemerkt, daß diese unterscheidenden Attribute oder Attributmengen Y Differenzierungs-Schlüssel sind, so daß der relationale Selektor

$$\sigma := \{(a, b) \in A \times A \mid Y(a) = Y(b)\}$$

dafür angegeben werden kann.

[1] Die *Potenzmenge* einer Menge M ist die Menge aller ihrer Teilmengen, $\mathcal{P}(M) := \{M' \subset M\}$.

Beispiel 5.5 *(Band-Join).* Sei A eine Tabelle und Y eine Attributmenge mit sämtlichst kardinal und ordinal skalierten Attributen. Hernandez und Stolfo [HS95] benutzten den sogenannten *Band-Join*, bei dem alle Datensätze $a, b \in A$ miteinander verglichen werden, bei denen der Wert $Y_i(b)$ jedes Attributes Y_i einer Attributmenge Y in einem Toleranzintervall $[Y_i(a) - \Delta_{i,0}, Y_i(a) + \Delta_{i,1}]$ liegt, für $\Delta_{i,0}, \Delta_{i,1} \in \mathbb{R}_{\geq 0}$. Der Band-Join definiert somit einen Selektor σ,

$$\sigma = \bigcap_i \sigma_i = \bigcap_i \{(a,b) \in A \times A : Y_i(a) - \Delta_{i,0} \leq Y(b) \leq Y_i(a) + \Delta_{i,1}\}\,. \tag{5.3}$$

Nominal skalierte Attribute lassen sich in den Band-Join mit aufnehmen, wenn man für sie die Wert-Gleichheit fordert (d.h. $\Delta_{i,0} = \Delta_{i,1} = 0$).
Für das Beispiel 4.13 auf Seite 109 ist für die Wohnungsanzeigen ein Band-Join für die Attribute *Größe*, *Räume* und *Bezirk* realisiert. Es sei bemerkt, daß diese drei Attribute dabei als approximative Differenzierungs-Schlüssel für A wirken.

Aus einer Entscheidungsregel $\delta' : A \times A \to \delta'(A \times A)$ gemäß Definition 2.17 auf Seite 50 läßt sich eine Entscheidungsregel $\delta : A \times A \to \delta'(A \times A)$ bilden durch

$$\delta(a,b) = \begin{cases} \delta'(a,b) & (a,b) \in \sigma, \\ \delta'_0 & \text{sonst.} \end{cases} \tag{5.4}$$

wobei $\delta'_0 \in \delta'(A \times A)$ der Entscheidung *kein Duplikat* entspricht.[2] Wir schreiben dafür auch $\delta = \delta' \circ \sigma$.
Das heißt, alle Datensatzpaare $(a,b) \in \sigma(A \times A)$ werden *a priori* als Duplikate verworfen, sie müssen also nicht gebildet werden.

Für die Entscheidungsregel

$$\delta'(a,b) = \begin{cases} 0 & \text{als Duplikat angenommen,} \\ \frac{1}{2} & \text{keine Entscheidung} \\ 1 & \text{als Duplikat verworfen.} \end{cases} \tag{5.5}$$

ist δ'_0 durch $\delta'_0 = 1$ gegeben.

Aus den in den Beispielen 5.4 und 5.5 dargestellten Selektoren läßt sich mittels Durchschnitt und Vereinigung ein allgemeiner relationaler Selektor bilden.

Bemerkung 5.6 *(relationaler Selektor).* Ein auf Bedingungen an Attributwerte basierender *relationaler Selektor* σ kann durch eine Formel ρ der relationalen Algebra angegeben werden:

$$\sigma = \{(a,b) \in A \times A \mid \rho(a,b)\}\,, \tag{5.6}$$

wobei $\rho(a,b) = \rho($`a.X1,...,a.Xn,b.X1,...,b.Xn`$)$ eine Formel mit bis zu $2n$ freien Variablen ist, die sich auf die Attributmenge $X = (X_1, \ldots, X_n)$ von A bezieht. ρ läßt sich in disjunktiver Normalform angeben,

$$\rho(a,b) = \bigvee_i \rho_i(a,b) = \bigvee_i \Big(\bigwedge_j \rho_{ij}(a,b) \Big), \tag{5.7}$$

[2] Die Entscheidung für die Datensatzpaare (a,a), $a \in A$ können wir an dieser Stelle vernachlässigen, da wir lediglich nach Duplikaten (a,b) mit $a \neq b$ in A suchen.

mit atomaren Termen $\rho_{ij}(a,b)$ wie

$$(\texttt{a.Xk} \diamond \texttt{b.Xk}),\ (\texttt{a.Xk} \diamond \texttt{x}) \text{ oder } (\texttt{b.Xk} \diamond \texttt{x})$$

unter Verwendung von Vergleichsoperatoren $\diamond \in \{<, \leq, =, \geq, >, \neq\}$, $k \in \{1, \ldots, n\}$ und konstanten Werten $x \in \mathrm{dom}(X_k)$ oder, für kardinal oder ordinal skalierte Attribute $(a.Xk \diamond b.Xk + x)$. Ein in disjunktiver Normalform formulierter Selektor σ läßt sich aus der Vereinigung (`UNION`) der den Formeln ρ_i entsprechenden SQL-Statements konstruieren,[3]

$$\sigma = \bigcup_i \sigma_i = \bigcup_i \{(a,b) \in A \times A \mid \rho_i(a,b)\}\,. \tag{5.8}$$

Beispiel 5.7. Sei $A =$ `Customer` eine Tabelle mit Kundendaten. `ID` sei ein Primärschlüssel für A, weitere Attribute seien `BirthDate` (Geburtsdatum) und `Initial` des Nachnamens. Der Selektor σ für A, der nur geordnete Datensatzpaare (a,b) aus $A \times A$ filtert (d.h. `a.ID<b.ID`), bei denen entweder `Initial` oder `BirthDate` übereinstimmen, ist gegeben durch

$$\sigma = \big\{(a,b) \in A \times A \mid \underbrace{\big(\rho_{11}(a,b) \wedge \rho_{12}(a,b)\big)}_{\rho_1(a,b)} \vee \underbrace{\big(\rho_{21}(a,b) \wedge \rho_{22}(a,b)\big)}_{\rho_2(a,b)}\big\}, \tag{5.9}$$

mit $\rho_{11}(a,b) = \rho_{21}(a,b) =$ `(a.ID<b.ID)`, $\rho_{12}(a,b) =$ `(a.Initial=b.Initial)`, $\rho_{22}(a,b) =$ `(a.BirthDate=b.BirthDate)`. Das zugehörige SQL-Statement ist

```
  SELECT A.*,B.*
    FROM Customer AS A, Customer AS B
    WHERE A.Initial = B.Initial
      AND A.ID < B.ID;
UNION
  SELECT A.*,B.*
    FROM Customer AS A, Customer AS B
    WHERE A.BirthDate = B.BirthDate
      AND A.ID < B.ID;
```

oder, zusammengefaßt in ein SQL-`SELECT`-Statement:

```
SELECT A.*,B.*
  FROM Customer AS A, Customer AS B
  WHERE A.ID < B.ID
    AND (A.Initial = B.Initial
    OR A.BirthDate = B.BirthDate);
```

Bemerkung 5.8 *(metrische Selektoren).* Wir können metrische Selektoren angeben, beispielsweise Δ-Umgebungs-Selektoren durch

$$\sigma = \{(a,b) \in A \times A : dist(Y(a), Y(b)) \leq \Delta\}\,, \tag{5.10}$$

[3]Man beachte jedoch, daß in dem Resultat einer `UNION`-Abfrage Datensätze mehrfach auftreten können.

sowie k-Nächste-Nachbarn-Selektoren mit

$$\sigma = \{(a,b) \in A \times A : b \in \text{k-NN}(Y(a))\}, \tag{5.11}$$

wobei k-NN$(Y(a))$ die Menge der k-Nächsten Nachbarn eines Datensatzes a bezüglich der für eine Attributmenge Y definierten Metrik *dist* bezeichnet, vgl. Abschnitt 3.3.4 auf Seite 91ff.

Wir untersuchen den Zusammenhang mit den im Kapitel 4 auf Seite 103 eingeführten semantischen Constraints.

Satz 5.9. *$\sigma : \mathcal{P}(A \times A) \to \mathcal{P}(A \times A)$ sei ein Selektor, der für A angegeben werden kann (z.B. über (approximative) Schlüssel oder Differenzierungs-Schlüssel). Dann gilt*

$$\text{reduction-rate}(A) = 1 - \frac{|\sigma(A \times A)|}{|A|^2}. \tag{5.12}$$

Wir erhalten daraus

Korollar 5.10. *Sei Y ein Attribut von A, das eine lineare Ordnung $(A, <_Y)$ induziert, z.B. ein* `ID`*-Attribut. Wir können dafür einen Selektor $\sigma_<(A \times A)$ angeben:*

$$\sigma_<(A \times A) := \{(a,b) \in A \times A \mid Y(a) < Y(b)\}. \tag{5.13}$$

Die Anzahl der zu bildenden Datensatzpaare aus $A \times A$ beträgt damit $\frac{1}{2}(|A|-1)(|A|-2)$. Dann ist

$$\text{reduction-rate}(A) = 1 - \frac{\frac{1}{2}(|A|-1)(|A|-2)}{|A|^2} = 1 - \frac{1}{2}\frac{(|A|-1)(|A|-2)}{|A|^2} \approx \frac{1}{2}.$$

Korollar 5.11. *Sei Y ein semantischer Differenzierungs-Schlüssel für A. Die Zahl der zu bildenden geordneten Datensatzpaare reduziert sich dann auf diejenigen, bei denen der Attributwert von Y übereinstimmt. Wir können einen Selektor $\sigma(A \times A)$ angeben:*

$$\sigma(A \times A) = \sigma' \circ \sigma_<(A \times A) = \{(a,b) \in \sigma_<(A \times A) \mid Y(a) = Y(b)\} \tag{5.14}$$

mit $\sigma'(A \times A) = \{(a,b) \in A \times A \mid Y(a) = Y(b)\}$. Bei Gleichverteilung der Attributwerte $y \in \text{dom}(Y)$ ergibt sich

$$\begin{aligned} \text{reduction-rate}(A) &= 1 - \frac{\frac{1}{2}(|A|-1)(|A|-2)}{|A|^2} \cdot \frac{\text{domain-size}(A,Y) \cdot |A|)}{|A|^2} \\ &\approx 1 - \frac{1}{2}\,\text{selectivity}(A,Y). \end{aligned}$$

Für einen semantischen Differenzierungs-Schlüssel mit einer Metrik $dist : (\text{dom}(Y))^2 \to \mathbb{R}, \Delta > 0$ ergibt sich entsprechend

$$\sigma(A \times A) = \sigma' \circ \sigma_<(A \times A) = \{(a,b) \in \sigma_<(A \times A) \mid dist(Y(a), Y(b)) \leq \Delta\} \tag{5.15}$$

mit $\sigma'(A \times A) = \{(a,b) \in A \times A \mid dist(Y(a), Y(b)) \leq \Delta\}$. *Bei einer gleichmäßigen Verteilung der Attributwerte* $y \in \mathrm{dom}(Y)$ *bezüglich des Abstandsmaßes*[4] *erhalten wir*

$$\begin{aligned} \text{reduction-rate}(A) &= 1 - \frac{1}{2}\frac{(|A'|-1)(|A'|-2)}{|A|^2} \cdot \frac{\Delta\text{-density}(A,Y)|A| \cdot |A|)}{|A|^2} \\ &\approx 1 - \frac{1}{2}\,\Delta\text{-density}(A,Y). \end{aligned}$$

Für einen k-NN Selektor mit einer Metrik $dist : (\mathrm{dom}(Y))^2 \to \mathbb{R}$,

$$\sigma(A \times A) = \{(a,b) \in \sigma_<(A \times A) \mid b \in \text{k-NN(Y,dist)}(a)\} \tag{5.16}$$

ergibt sich

$$\begin{aligned} \text{reduction-rate}(A) &= 1 - \frac{1}{2}\frac{(|A'|-1)(|A'|-2)}{|A|^2} \cdot \frac{k \cdot |A|}{|A|^2} \\ &\approx 1 - \frac{k}{|A|}. \end{aligned}$$

5.1.1 Bestimmen einer optimalen Vorauswahl von Datensatzpaaren

Das der Wahl einer optimalen Vorauswahl von Datensatzpaaren zugrundeliegende Prinzip wurde von Neiling und Müller [NM01] entworfen und wird hier weiter verfolgt. Die dort definierten *Ablehnungsregeln, engl. rejection rules,* entsprechen hier den Selektoren σ_i. Im Folgenden wird auf das Verfahren eingegangen.

Wir betrachten eine Tabelle A, für die k Selektoren $\sigma_1, \ldots, \sigma_k$ angegeben werden können. Eine Vorauswahl σ ist dann eine Kombination aus diesen Selektoren, beispielsweise

$$\sigma = \sigma_{i_1} \cap \sigma_{i_2} \text{ oder } \sigma = \sigma_{i_1} \cup \sigma_{i_2} \tag{5.17}$$

Aus k Selektoren $\sigma_1, \ldots, \sigma_k$ wird ein Selektor σ für eine Tabelle A in zwei Schritten erzeugt:

1. Berechnung eines oder mehrerer Selektoren, die jeweils dem Durchschnitt beliebig vieler σ_i, $i = 1, \ldots, k$ entsprechen, und anschließend
2. Vereinigung/Merging der unter 1. erzeugten Selektoren (UNION für ausschließlich relationale Selektoren, simultane Abfrage für allgemeine Selektoren).

Hierbei ist zu beachten, daß der Durchschnitt in 1. nur für gleichartige Selektoren gebildet werden kann. Dies sei für zwei Klassen von Selektoren erläutert:

[4] D.h. es gibt ein $c \in \mathbb{N}$ so daß $\forall y \in \mathrm{dom}(Y) \mid \{a \in A \mid dist(Y(a), y) \leq \Delta\} \mid \approx c$ erfüllt ist.

- Relationale Selektoren: Durch Konjunktion der Bedingungen, z.B. $(\rho_i \wedge \rho_j)$ für $\sigma_i \cap \sigma_j$,
- Metrische Selektoren: Durch Definition einer Maximumsmetrik z.B. $dist = \max(dist_i, dist_j)$ für $\sigma_i \cap \sigma_j$.

Im zweiten Schritt ist die Vereinigung einer Menge mit einer ihrer Teilmengen nicht sinnvoll und daher nicht durchzuführen, z.B. für die Selektoren $(\sigma_1 \cap \sigma_2 \cap \sigma_3) \cup (\sigma_1 \cap \sigma_2)$, da $(\sigma_1 \cap \sigma_2 \cap \sigma_3) \subset (\sigma_1 \cap \sigma_2)$.

Um eine optimale Vorauswahl treffen zu können, ermitteln wir

1. Die *Selektionsrate* $\text{sel}(\sigma, A)$ einer Vorauswahl, d.h. wie hoch ist der Anteil der selektierten Datensatzpaare, sowie
2. Die *Fehlerrate* $\text{err}(\sigma, A)$ einer Vorauswahl, d.h. den Anteil der nicht selektierten dubletten Datensatzpaare.

Die Fehlerrate wird bezüglich einer gewählten binären Relation *Dupl* gemessen. Mögliche Setzungen für *Dupl* sind ($\delta(a, b) = 0$ bezeichne die Klassifikation eines Paares als Duplikat, *Same* eine Same-Relation für A):

1. $Dupl = \{(a, b) \in A \times A \mid \delta(a, b) = 0\}$,
2. $Dupl = Same$, sowie
3. $Dupl = \{(a, b) \in Same \mid \delta(a, b) = 0\}$.

Definition 5.12 (Selektions- und Fehlerrate eines Selektors). A sei eine Tabelle, zu der eine binäre Relation *Dupl* angegeben sei, in der Duplikate aus A erfaßt sind. σ sei ein Selektor für A. Dann definieren wir die Selektions- und Fehlerrate des Selektors σ:[5]

$$\text{sel}(\sigma, A) := \frac{|\sigma(A \times A)|}{|A|^2} = 1 - \text{reduction-rate}(A), \tag{5.18a}$$

$$\begin{aligned} \text{err}(\sigma, A) &= \text{err}(\sigma, A, Dupl) \\ &:= \frac{|TC(Dupl) \backslash \{\sigma(A \times A) \cap TC(Dupl)\}|}{|TC(Dupl)|} \\ &= 1 - \frac{|\sigma(A \times A) \cap TC(Dupl)|}{|TC(Dupl)|} \end{aligned} \tag{5.18b}$$

Eine optimale Vorauswahl sollte eine niedrige Selektionsrate mit einer geringen Fehlerrate vereinen. Das führt zu den folgenden drei Optimierungsstrategien,

1. Minimiere das Produkt $\text{sel}(\sigma, A) \cdot \text{err}(\sigma, A)$,
2. Minimiere $\text{sel}(\sigma, A)$ unter der Nebenbedingung $\text{err}(\sigma, A) \leq \text{err}_0$, sowie

[5] Die Fehlerrate err (5.18b) messen wir für geordnete Paare von Duplikaten (das sind genau die mit $TC(Dupl)$ bestimmten), da es generell genügt, geordnete Paare in einer Vorauswahl zu erzeugen, vgl. Korollar 5.10.

3. Minimiere $\mathrm{err}(\sigma, A)$ unter der Nebenbedingung $\mathrm{sel}(\sigma, A) \leq \mathrm{sel}_0$.

Als zusätzliche Nebenbedingung können wir den Aufwand für Erzeugung und einmalige Verarbeitung der Vorauswahl beschränken (d.h. Laden aller Paare in σ), beispielsweise indem wir die Nebenbedingung

$$\mathrm{runtime}(\sigma, A) \leq t_0$$

mit berücksichtigen. Alternativ dazu kann man den Aufwand in der Zielfunktion mit berücksichtigen, z.B. in dem Optimierungsproblem

4. Maximiere den Quotienten[6] $(1 - \mathrm{sel}(\sigma, A))/\mathrm{runtime}(\sigma, A)$ unter der Nebenbedingung $\mathrm{err}(\sigma, A) \leq \mathrm{err}_0$.

Im Übrigen ist der Gesamtaufwand eines Objektidentifizierungs-Verfahrens umso größer, je mehr Datensatzpaare verarbeitet (geladen, konvertiert, verglichen und gespeichert) werden müssen. Deshalb hat in der Regel eine Verringerung der Selektionsrate eine Verringerung des Gesamtaufwands zur Folge.

Lemma 5.13. *σ_1, σ_2 seien Selektoren für A, weiter sei für $i = 1, 2$ $\eta_i = \mathrm{sel}(\sigma_i, A)$, $\kappa_i = \mathrm{err}(\sigma_i, A)$. Dann gilt*

$$\max(\eta_1, \eta_2) \leq \mathrm{sel}(\sigma_1 \cup \sigma_2, A) \leq \eta_1 + \eta_2 \tag{5.19a}$$
$$0 \leq \mathrm{err}(\sigma_1 \cup \sigma_2, A) \leq \min(\kappa_1, \kappa_2) \tag{5.19b}$$

$$0 \leq \mathrm{sel}(\sigma_1 \cap \sigma_2, A) \leq \min(\eta_1, \eta_2) \tag{5.19c}$$
$$\max(\kappa_1, \kappa_2) \leq \mathrm{err}(\sigma_1 \cap \sigma_2, A) \leq \kappa_1 + \kappa_2 \tag{5.19d}$$

Beweis. Wir kürzen im Beweis $\sigma_i(A \times A)$ und $\sigma_1 \diamond \sigma_2(A \times A)$ durch σ_i bzw. $\sigma_1 \diamond \sigma_2$ ab (für $i = 1, 2$ und $\diamond \in \{\cup, \cap\}$) sowie $TC(Dupl)$ verkürzen wir zu TC.

Aus $\sigma_i \subset (\sigma_1 \cup \sigma_2)$ für $i = 1, 2$ und der Ungleichung

$$|M_1 \cup M_2| \leq |M_1| + |M_2| \tag{5.20}$$

für zwei Mengen M_1, M_2 erhalten wir (5.19a) mit

$$\begin{aligned} |A|^2 \eta_i = |\sigma_i| &\leq |\sigma_1 \cup \sigma_2| \\ &\leq |\sigma_1| + |\sigma_2| = |A|^2(\eta_1 + \eta_2). \end{aligned}$$

Wegen $\big((\sigma_1 \cup \sigma_2) \cap TC\big) \supset (\sigma_i \cap TC)$ gilt

$$|(\sigma_1 \cup \sigma_2) \cap TC| \geq |\sigma_i \cap TC|$$

[6] Beachte die Identität

$$\frac{\text{reduction-rate}(A)}{\mathrm{runtime}(\sigma, A)} = \frac{(1 - \mathrm{sel}(\sigma, A))}{\mathrm{runtime}(\sigma, A)}.$$

für $i = 1, 2$ und also

$$\mathrm{err}(\sigma_1 \cup \sigma_2) = 1 - \frac{|(\sigma_1 \cup \sigma_2) \cap TC|}{|TC|} \leq 1 - \frac{|\sigma_i \cap TC|}{|TC|} = \kappa_i,$$

womit (5.19b) gezeigt ist.

Umgekehrt gilt $\sigma_i \supset (\sigma_1 \cap \sigma_2)$ für $i = 1, 2$ und daher auch (5.19c):

$$|\sigma_1 \cap \sigma_2| \leq |\sigma_i| = |A|^2 \eta_i.$$

Da für Mengen $M_1, M_2 \subset M$ stets

$$\big(M\backslash\{M_1\} \cup M\backslash\{M_2\}\big) \supset M\backslash\{M_1 \cap M_2\}$$

erfüllt ist, erhalten wir die Ungleichung

$$|M\backslash\{M_1\} \cup M\backslash\{M_2\}| \geq |M\backslash\{M_1 \cap M_2\}|.$$

Zusammen mit (5.20) erhalten wir daraus (5.19d):

$$\begin{aligned}
(\kappa_1 + \kappa_2)\,|TC| &= |TC\backslash\{TC \cap \sigma_1\}| + |(TC\backslash\{TC \cap \sigma_2\}| \\
&\geq |TC\backslash\{TC \cap \sigma_1\} \cup TC\backslash\{TC \cap \sigma_2\}| \\
&\geq |TC\backslash\{TC \cap \sigma_1 \cap \sigma_2\}| = \mathrm{err}(\sigma_1 \cap \sigma_2)\,|TC| \\
&\geq |TC\backslash\{TC \cap \sigma_i\}| = \kappa_i\,|TC|, \quad i = 1, 2.
\end{aligned}$$

□

Da wir bei der Optimierung nicht für jede Kombination σ der Selektoren $\sigma_1, \ldots, \sigma_k$ Werte für $\mathrm{err}(\sigma)$ und $\mathrm{sel}(\sigma)$ bestimmen können, müssen wir eine Annahme treffen. Für den Optimierungsansatz setzen wir $\mathrm{err}(\sigma)$ und $\mathrm{sel}(\sigma)$ durch Werte innerhalb der Grenzen (5.19) (Es bezeichne $\eta_i = \mathrm{sel}(\sigma_i)$ sowie $\kappa_i = \mathrm{err}(\sigma_i)$),

$$\mathrm{sel}^*(\sigma_1 \cup \sigma_2 \cup \ldots \cup \sigma_k, A) := \frac{1}{2}\big(\max_i(\eta_i) + \sum_i \eta_i\big), \tag{5.21a}$$

$$\mathrm{err}^*(\sigma_1 \cup \sigma_2 \cup \ldots \cup \sigma_k, A) := \sqrt{\min_i(\kappa_i)} \cdot \prod_i \sqrt{\kappa_i}, \tag{5.21b}$$

$$\mathrm{sel}^*(\sigma_1 \cap \sigma_2 \cap \ldots \cap \sigma_k, A) := \sqrt{\min_i(\eta_i)} \cdot \prod_i \sqrt{\eta_i}, \tag{5.21c}$$

$$\mathrm{err}^*(\sigma_1 \cap \sigma_2 \cap \ldots \cap \sigma_k, A) := \frac{1}{2}\big(\max_i(\kappa_i) + \sum_i \kappa_i\big). \tag{5.21d}$$

Wir erläutern die Setzungen in (5.21). Für (5.21a) und (5.21d) setzen wir den Wert durch den Wert des Maximums (der unteren Schranke gemäß (5.19)) plus der Hälfte der Summe der verbleibenden Werte (die Summe aller Werte bildet die obere Schranke). Für (5.21b) und (5.21c) erhalten wir bei Unabhängigkeit der einzelnen Dimensionen als Wert das Produkt, d.h. für (5.21b) gilt dann $\mathrm{err}^*(\sigma_1 \cup \sigma_2 \cup \ldots \cup \sigma_k, A) \approx \prod_i \kappa_i$. Da wir die Unabhängigkeit nicht garantieren können, setzen wir einen höheren Wert in (5.21b) für die Fehlerrate bzw. für die Selektionsrate in (5.21c). Dazu verwenden wir eine in (0,1) konvexe Transformation, $x \mapsto \sqrt{x}$, für alle κ_i bis auf den minimalen Wert (der die obere Schranke bildet), der nicht transformiert in das Produkt eingeht.

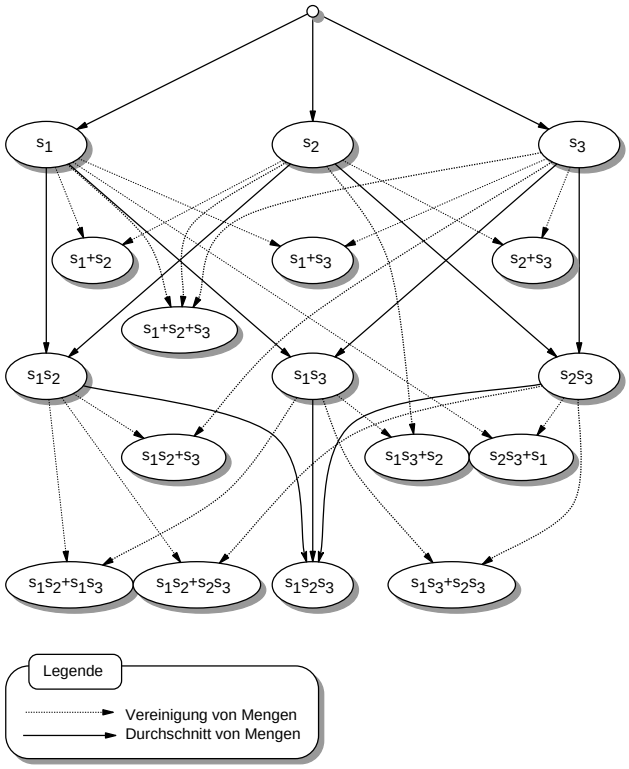


Abbildung 5.1: Der vollständige Suchgraph für drei Selektoren s_1, s_2, s_3 (s_i steht für σ_i, $s_i s_j$ bedeutet $s_i \cap s_j$, $s_i + s_j$ bedeutet $s_i \cup s_j$), vgl. Beispiel 5.14

Beispiel 5.14. Wir betrachten die Tabelle $A = \texttt{Customer}$ aus dem Beispiel 5.7 auf Seite 126 und nehmen an, daß sie 100 000 Datensätze enthalte, von denen 40% Duplikate seien. Es seien drei Selektoren angegeben:

$$\begin{aligned}
\sigma_1(A \times A) &= \{(a,b) \mid \texttt{a.BirthDate} = \texttt{b.BirthDate}\}, \\
\sigma_2(A \times A) &= \{(a,b) \mid \texttt{a.Initial} = \texttt{b.Initial}\}, \\
\sigma_3(A \times A) &= \{(a,b) \mid \texttt{a.ZIP} = \texttt{b.ZIP}\},
\end{aligned}$$

die jeweils approximative Differenzierungs-Schlüssel für das Geburtsdatum, den Initialen des Nachnamens und für die Postleitzahl (ZIP-Code) ausdrücken.

Selektor σ_i	η_i	κ_i
σ_1 (Geburtsdatum)	0.003	0.025
σ_2 (Initial)	0.050	0.005
σ_3 (PLZ)	0.005	0.010

Aus den Selektoren σ_i lassen sich durch Mengen-Durchschnitt und Vereinigung mehrere Selektoren σ bilden, z.B.

$$\begin{aligned}
\sigma &= \sigma_1 \cup (\sigma_2 \cap \sigma_3), \\
&= \{(a,b) \mid \texttt{a.BirthDate} = \texttt{b.BirthDate} \quad \texttt{OR} \\
&\qquad (\texttt{a.Initial} = \texttt{b.Initial} \quad \texttt{AND} \quad \texttt{a.ZIP} = \texttt{b.ZIP})\}
\end{aligned}$$

mit

$$\begin{aligned} \mathrm{err}^*(\sigma, A) &= \mathrm{err}^*(\sigma_1 \cup (\sigma_2 \cap \sigma_3), A) \\ &= 0.00198, \\ \mathrm{sel}^*(\sigma, A) &= \mathrm{sel}^*(\sigma_1 \cup (\sigma_2 \cap \sigma_3), A) \\ &= 0.00356 \end{aligned}$$

Aus der Menge der bildbaren Selektoren soll der bezüglich einer Zielfunktion optimale Selektor ausgewählt werden. Der vollständige Suchgraph für das Optimierungsproblem ist in der Abbildung 5.1 dargestellt. Die Werte der Selektions- und Fehlerraten sind in der Tabelle 5.1 zu finden.

Für die Lösung des Optimierungsproblems verwenden wir die Setzungen der Selektions- und Fehlerraten gemäß (5.21). Weiter sei

$$\mathrm{runtime}(\sigma_i, A) = 100, \mathrm{runtime}(\sigma_i \cap \sigma_j, A) = 150, \mathrm{runtime}(\sigma_1 \cap \sigma_2 \cap \sigma_3, A) = 200$$

und

$$\mathrm{runtime}(\sigma \cup \sigma', A) = \mathrm{runtime}(\sigma, A) + \mathrm{runtime}(\sigma', A).$$

Wir lösen exemplarisch zwei Optimierungsprobleme für das Beispiel, eines ohne und ein zweites mit Berücksichtigung des Aufwandes (Σ bezeichnet die Menge aller bildbaren Selektoren):

(1) $\min_{\sigma \in \Sigma} \stackrel{!}{=} ZF^{(1)}(\sigma) := \mathrm{err}^*(\sigma, A)$ unter der NB: $\mathrm{sel}^*(\sigma, A) \leq \eta_0 = 0,010$.

(2) $\max_{\sigma \in \Sigma} \stackrel{!}{=} ZF^{(2)}(\sigma) := (1 - \mathrm{sel}^*(\sigma, A))/\mathrm{runtime}(\sigma, A)$
unter der NB: $\mathrm{err}^*(\sigma, A) \leq \kappa_0 = 0.025$.

Wir erhalten je Optimierungsproblem eine optimale Lösung:

(1) $\sigma^{(1)} = \sigma_1 \cap \sigma_3$,

(2) $\sigma^{(2)} = (\sigma_1 \cap \sigma_2) \cup (\sigma_1 \cap \sigma_3)$.

Die optimale Lösung wird durch eine Suche entlang des Suchgraphen (vgl. Abb. 5.1) ermittelt, wobei bei einer Verletzung der Nebenbedingung abgebrochen wird. Bei Optimierungsproblemen mit einer größeren Zahl von Selektoren, z.B. mit mehr als 50 alternativen Selektoren σ_i, wird der Einsatz von heuristischen Suchverfahren notwendig. Typischerweise gibt es jedoch maximal 10–25 alternative Selektoren bei Objektidentifizierungs-Problemen, so daß selbst die vollständige Enumeration aller Kombinationen der Selektoren praktikabel ist.

5.2 Die Verwendung von Indexstrukturen

Im vorangegangenen Abschnitt haben wir uns mit Selektoren befaßt. Für die effiziente Umsetzung eines Selektors ist der Einsatz von Indexstrukturen erforderlich, da große Datenbestände nach Datensätzen (respektive Paaren) gefiltert werden, die Bedingungen an Attributwerte erfüllen sollen. Hierbei ist es für die Wahl einer Indexstruktur von Bedeutung,

Tabelle 5.1: Die Werte der Fehler- und Selektionsraten sowie die Laufzeit für die Selektoren des Beispiels 5.14

Selektor $s_i = \sigma_i$	η_i	κ_i	runtime
s_1	.00300	.02500	100
s_2	.05000	.00500	100
s_3	.00500	.01000	100
$s_1 s_2$	.00067	.02750	150
$s_1 s_3$	.00021	.03000	150
$s_2 s_3$	.00112	.01250	150
$s_1 s_2 s_3$	.00005	.03250	200
$s_1 + s_2$	.05150	.00079	200
$s_1 + s_3$	*.00650*	*.00158*	*200*
$s_2 + s_3$	.05250	.00050	200
$s_1 + s_2 + s_3$	.05400	.00008	300
$s_1 + s_2 s_3$	.00356	.00198	250
$s_2 + s_1 s_3$	.05011	.00087	250
$s_3 + s_1 s_2$	.00534	.00166	250
$s_1 s_2 + s_1 s_3$	*.00078*	*.00476*	*300*
$s_1 s_2 + s_2 s_3$	.00145	.00207	300
$s_1 s_3 + s_2 s_3$	.00122	.00217	300

(1) ob auf Wert-Gleichheit geprüft werden soll,

(2) ob die alternative Übereinstimmung von Wert-Kombinationen erfüllt sein soll (z.B. mindestens $k \geq 2$ Bigrams[7]), oder

(3) ob eine tolerierbare Abweichung bezüglich eines Toleranzintervalles für ordinal und kardinal skalierte Attribute oder bezüglich einer Metrik für metrisch skalierte Attribute eingehalten werden soll, oder ob die k nächsten Nachbarn bezüglich einer Metrik gefunden werden sollen.

Zu den Punkten (1)–(3) können wir feststellen:

(1) Die Prüfung auf Wert-Gleichheit ist durch den Einsatz der Indexstrukturen, die in marktüblichen Datenbank-Management-Systemen verfügbar sind (z.B. Bitmaps, Hashing, B-Trees), effizient lösbar.

(2) Die Prüfung auf alternative Übereinstimmung von Wert-Kombinationen ist zurückführbar auf die Prüfung auf Wertgleichheit. Eine Umsetzung solch einer Indexstruktur ist mit dem Ansatz von Schallehn et. al [SSS01] möglich, bei dem eine Erweiterung der Anfragesprache (z.B. SQL) um benutzerspezifische Funktionen vorgeschlagen wird. Die hier verwandte Umsetzung basiert auf dem Prinzip *relationaler Indexstrukturen* (vgl. Kriegel et. al [KPPS03]), bei dem benutzerspezifische Indexstrukturen auf Relationen abgebildet werden, die für

[7] Bigrams sind aufeinanderfolgende Buchstabenpaare einer Zeichenkette, z.B. *HA AN NS* für *Hans*, vgl. Definition 5.15 auf Seite 136.

einen effizienten Zugriff mit den in Datenbank-Management-Systemen verfügbaren Indexstrukturen versehen werden können. Wir erzeugen einen Index für die Prüfung auf alternative Übereinstimmung von Wert-Kombinationen entweder mit einer Index-Tabelle, in der

(a) Wert-Kodierungen der betreffenden Attribute der Datensätze, oder

(b) Kombinationen von Wert–Kodierungen erzeugt und erfaßt werden.

Während für den Fall (a) eine `COUNT`-Operation notwendig ist, läßt sich diese durch Einsatz der Lösung (b) vermeiden, wobei alle Kombinationen von k Werten gespeichert werden müssen, dies sind bei n Werten allerdings $\binom{n}{k}$ Einträge. Die Realisierung eines solchen Indexes für N-Grams wird im Abschnitt 5.2.1 auf Seite 135 beschrieben.

(3a) Kardinale Skala: Die effektive Filterung von Datensätzen, für die kardinal skalierte Attribute ein mehrdimensionales Toleranzintervall einhalten, ist durch mehrdimensionale Indexstrukturen realisierbar, für einen Überblick siehe dazu z.B. [GG98] oder [BBKM00]. Geeignete mehrdimensionale Indexstrukturen sind (Range-encoded) Bitmap-Indizes, Baum-basierte Indexstrukturen (z.B. kdb-tree [Rob81], R*-tree oder X-tree [BKK96]) oder Grid-files (z.B. ε-grid [BBKK01]). In den verschiedenen Datenbank-Management-Systemen sind einige dieser mehrdimensionalen Indexstrukturen verfügbar. Zur Gewährleistung eines effizienten Zugriffs kann also für die Attributmengen, an die in einem (relationalen) Selektor σ Bedingungen gestellt werden, ein Index in der Datenbank erzeugt werden.

(3b) Metrische Skala: Für die Filterung von Datensätzen, bei denen metrisch skalierte Attribute eine tolerierbare Abweichung nicht überschreiten sowie das Filtern der k–nächsten Nachbarn bezüglich einer Metrik, sind Indexstrukturen für metrische Skalen einsetzbar. Diese befinden sich derzeit in der Forschung (z.B. *M–Tree* [CPZ97] oder MVP-Tree [BO99]) und müssen als externe Indexstruktur erzeugt und verwaltet werden.

5.2.1 Index-Erstellung für Wertkombinationen

In Datenbanken sind viele Text-Informationen in Form von Zeichenketten gespeichert, wie z.B. Namen- und Adreßdaten. Dem effizienten Zugriff auf Datensätze und Datensatzpaare, die Bedingungen an diese Attribute erfüllen, kommt deshalb im Rahmen der Objektidentifizierung eine besondere Bedeutung zu. Für die Indexierung von Zeichenketten, insbesondere für die Überprüfung auf Übereinstimmung von Teilen der Zeichenketten (*engl.: substrings*), stehen Indexstrukturen zur Verfügung. Will man jedoch andere Bedingungen, wie beispielsweise die Übereinstimmung von $k \geq 2$ NGrams, an Zeichenketten formulieren und effizient abarbeiten, ist eine spezielle Codierung und Umformulierung der Bedingungen notwendig, so daß auf Wert-Gleichheit prüfende Indexstrukturen eingesetzt werden können. Durch die-

ses Vorgehen läßt sich die Effizienz des Zugriffs durch die verfügbaren Indexstrukturen zur Prüfung auf Wert-Gleichheit garantieren.

Falls man einen Selektor realisieren möchte, der nicht auf einer Metrik basiert, läßt sich eine Distanz-basierte Indexstruktur nicht einsetzen, da die Dreiecksungleichung nicht erfüllt ist. An zwei Beispielen stellen wir dar, wie solche Selektoren durch eine geeignete Berechnung von Codes und deren Speicherung in Tabellen, die versehen mit effizienten (numerischen) Indizes für uns als Index-Tabellen fungieren, umgesetzt werden können. Die zentrale Idee hierbei ist, die in relationalen Datenbank-Management-Systemen verfügbaren Indexstrukturen durch eine geeignete Kodierung des Selektors nutzbar zu machen.

Definition 5.15 (NGram). w sei eine Zeichenkette (ein Wort) über einem Alphabet $\mathcal{A}$, d.h. w ist eine Folge von Zeichen aus $\mathcal{A}$, $w = w_1 w_2 \cdots w_m$, $w_i \in \mathcal{A}$. Weiter sei $n \in \mathbb{N}_+$, $n \leq n$. Dann sind $w_j w_{j+1} \cdots w_{j+n-1}$, $j = 1, \ldots, (m - n + 1)$ die *NGrams* der Länge n von w.
Für $n = 2$ bezeichnet man sie auch als *BiGrams*.

Beispiel 5.16 *(NGram-Index).* Es ist eine Datenbank A mit Personendaten (Name, Vorname, Adresse und Geburtsdatum) gegeben, in der Duplikate gefunden werden sollen. Es sollen alle Datensätze miteinander verglichen werden, bei denen mindestens k NGrams der Länge n des Namens übereinstimmen. D.h. wir bilden den Selektor ($k \geq 2$)

$$\sigma(A \times A) = \{(a, b) \mid \#MatchingNGrams(\texttt{a.Name}, \texttt{b.Name}, n) \geq k\} \quad (5.22)$$

Dazu erzeugen wir eine Index-Tabelle `NGramIndex`, um daraus mit σ eine Vorauswahl zu erzeugen:

1. Für alle Datensätze $a \in A$: Bestimmen der NGrams des Namens eines des Datensatzes (z.B. mit einer Funktion `NGramList(Name)`), numerische Codierung jedes NGrams und Speicherung des Codes in einer Tabelle `NGramIndex(ID,CODE)` zusammen mit der `ID` des Datensatzes a. Um effizient auf Datensätze zugreifen zu können, wird für die Spalten `ID` und `CODE` der Tabelle `NGramIndex` ein eindimensionaler Index in der Datenbank erzeugt, z.B. ein B-Baum [] für `ID` und ein komprimierter Bitmap-Index für `CODE`.

2. Für alle Datensätze $a \in A$: Filtern der Datensätze b (d.h. jeweils die `ID`) aus der Indextabelle `NGramIndex(ID,CODE)`, bei denen bei mindestens k Codes übereinstimmen. Das liefert geordnete Tupel (`a.ID, b.ID`).

Die Auswahl der Datensätze im 2. Schritt erfolgt über das unten angegebene SQL-Statement `Statement_1`, bei dem die Variablen `A_ID` und `B_ID` gebunden werden und das Statement ausgeführt wird. Die Anzahl der gleichzeitig im Arbeitspeicher zu haltenden Datensätze sei `WindowSize`≥ 2, `N`:= $\frac{1}{2}$`WindowSize` und die `ID` eine fortlaufende Nummer in A. Der Selektor σ (5.22) ist dann gegeben durch die Vereinigung (`UNION`) der Resultate. Der Index kann mit dem folgenden SQL-Statement in Blöcken von bis zu `WindowSize` Datensätzen abgearbeitet werden, wenn die `A_ID` aufsteigend

gebunden wird, und der Wert A_ID und B_ID für einen darauffolgenden Block jeweils mit dem maximalen Wert des letzten Blockes gesetzt wird.

```
Statement_1(WindowSize, A_ID, B_ID, k)=
{ SELECT TOP WindowSize idxA.ID, idxB.ID
   FROM NGramIndex AS idxA, NGramIndex AS idxB
   WHERE idxA.CODE = idxB.CODE
     AND idxA.ID < idxB.ID
     AND (idxA.ID > A_ID OR (idxA.ID = A_ID AND idxB.ID > B_ID))
   GROUP BY idxA.ID, idxB.ID
   HAVING COUNT(*) >= k; }
```

Dieses Vorgehen birgt die Nachteile, daß (1) alle Paare von Datensätzen gebildet werden müssen, bei denen mindestens ein NGram übereinstimmt, und (2), daß über alle gefilterten Datensätze ein COUNT berechnet werden muß. Durch Vorberechnung aller Kombinationen von k NGrams und deren Speicherung[8] lassen sich beide Nachteile vermeiden, allerdings bei Speicherung von bis zu $\binom{n}{k}$ statt n Einträgen je Datensatz aus A in der Tabelle NGramIndex(ID,CODE), $n = \#NGrams$. Folglich ist es nur praktikabel bei kurzen Zeichenketten (d.h. $n \approx$10-15) und kleinen k (z.B. $k = 2, 3$). Das obige SQL-Statement reduziert dann zu:

```
Statement_2(WindowSize, A_ID, B_ID, k)=
{ SELECT TOP WindowSize idxA.ID, idxB.ID
    FROM NGramIndex AS idxA, NGramIndex AS idxB
    WHERE idxA.CODE = idxB.CODE
      AND idxA.ID < idxB.ID
      AND (idxA.ID > A_ID
          OR (idxA.ID = A_ID AND idxB.ID > B_ID));}
```

Im Gegensatz zum Statement_1 werden im Statement_2 nur die Datensatzpaare gebildet, bei denen *mindestens* k NGrams übereinstimmen. Allerdings werden hier alle geordneten Tupel (a.ID, b.ID) mehrfach selektiert, bei denen mehr als k NGrams übereinstimmen. Das ließe sich zwar durch Verwendung von SELECT DISTINCT vermeiden, ist aber aufgrund des dafür notwendigen Rechenaufwandes bei der Selektion nicht sinnvoll. Bei der weiteren Verarbeitung kann alternativ dazu kontrolliert werden, daß für einen Tupel (a.ID, b.ID) nur einmalig die Datensätze aus A gelesen werden.

Bemerkung 5.17. Das oben angegebene SQL-Statement Statement_2 ist generell als Selektion von Tupeln basierend auf der Gleichheit von *mindestens* einem in einer Index-Tabelle hinterlegten Codes (beispielsweise von Attributwerten) geeignet, etwa für die phonetische Kodierung von Namen.

Bemerkung 5.18. Der im Beispiel 5.5 auf Seite 125 beschriebene Band-Join für kardinal skalierte Attribute $X_1, \ldots, X_k$ läßt sich ebenfalls mit einer analogen Kodierung umsetzen (alternativ zur Verwendung mehrdimensionaler Indexstrukturen, vgl. Punkt (3a) auf Seite 135), das Vorgehen ist folgendermaßen:

[8] D.h. anstatt des Codes eines NGrams wird ein Code für die Kombination von k NGrams berechnet, z.B. durch Aneinanderfügen der Codes der einzelnen Bigrams unter vorheriger lexikographischer Sortierung.

- Für jedes Attribut X_i, das ein Toleranzintervall $\pm\Delta_i$ nicht überschreiten soll, wird eine Indextabelle `IDX_i(ID,CODE)` angelegt.

- Für die Wertebereiche $\mathrm{dom}(X_i) \subset \mathbb{R}$ wird eine hinreichend feine Diskretisierung $\mathrm{dom}(X_i) = \bigcup I_{ij}$ von Intervallen $I_{ij} =\subset \mathbb{R}$ mit $I_{ij} \cap I_{ij'} = \emptyset$ für $j \neq j'$ gewählt[9] und jedem Intervall ein Code-Wert zugeordnet.

- Für einen Vektor $(x_1, \ldots, x_k)$ eines Datensatzes, $x_i \in \mathrm{dom}(X_i)$, werden alle Intervalle I_{ij} ermittelt, für die $I_{ij} \cap [x_i - \Delta_i, x_i + \Delta_i] \neq \emptyset$ gilt und die assoziierten Code-Werte jeweils als Paar mit der `ID` des Datensatzes in den Tabellen `IDX_i(ID,CODE)` gespeichert.

Das Abarbeiten dieses Band-Joins kann dann analog zum Vorgehen für N-Grams erfolgen, wobei ein Paar nur dann gebildet wird, wenn in jeder der i Index-Tabellen jeweils mindestens ein Code übereinstimmt (durch Bildung des INNER JOIN).

Bemerkung 5.19 *(Caching).* Beim Abarbeiten eines CODE-Indexes müssen die Datensätze, die mehrere Codes haben, zumeist mehrfach geladen werden.

Durch die im Folgenden kurz skizzierte Caching-Strategie läßt sich das mehrfache Laden von Datensätzen begrenzen. Zunächst nehmen wir an, daß jeder Block von Datensätzen mit einem identischen `Code` im Arbeitsspeicher zu halten ist, d.h. daß die `WindowSize` genügend groß ist (andernfalls müssen Datensätze mit identischem Code, die nicht gleichzeitig in den Arbeitsspeicher passen, mehrfach geladen werden). Je einzelnem Code werden alle Datensätze geladen und vollständig abgearbeitet, d.h. es werden alle Paare gebildet und prozessiert. Da die meisten Code-Blöcke nicht den gesamten Arbeitsspeicher benötigen, kann ein dynamischer Cache gefüllt werden, der die schon geladenen Datensätze solange vorhält, bis sie für einen großen Code-Block überschrieben werden müssen, oder bis sie vollständig abgearbeitet sind.

Um zu erreichen, daß Datensätze nicht mehrfach geladen werden müssen, gibt es mehrere Strategien des Caching. Falls es beispielsweise durch Clustern der Datensätze möglich ist, daß ein Großteil der geladenen Datensätze ebenfalls im nächsten Block enthalten sind, kann dies eine hohe Effizienz garantieren. Jedoch ist dies nur schwer im Allgemeinen zu erfüllen, da sie von der Häufigkeit der einzelnen Codes *und* der Zahl von Codes je Datensatz abhängt. Der Effizienzgewinn ist daher nicht vorhersagbar und hängt zudem von der verwandten Cluster-Methode ab.

Wir verfolgen eine alternative Caching-Strategie. Sie besteht darin, die Code-Blöcke (das sind Blöcke aller Datensätze mit jeweils übereinstimmendem Code) in eine Reihenfolge zu bringen, daß für einzelne Datensätze sichergestellt werden kann, daß alle ihre Codes geladen wurden (da sie dann aus dem Cache entfernt werden können). Das folgende Sortierschema ist dahingehend optimal.[10]

[9] Die Disjunktivität der Intervalle $I_{ij} \cap I_{ij'} = \emptyset$ fordern wir nur aus Effizienzgründen, sie kann auch fallengelassen werden.

[10] Alternativ zum Sortieren der Index-Tabelle können auch die Datensätze selbst physisch in dieser Reihenfolge gespeichert werden.

1. Anordnung der Index-Einträge gruppiert in Blöcken je gleichem Code, wobei jeder Code nur einmal verarbeitet wird und

2. der erste Code-Block beliebig ist,

3. die Reihenfolge der Einträge innerhalb des ersten Code-Blockes ebenfalls beliebig ist, jedoch aber die Abarbeitung der weiteren Codes bestimmt,

4. die $(k-1)$ darauffolgenden Blöcke sind durch die $(k-1)$ verbleibenden Codes des ersten Datensatzes aus dem ersten Code-Block bestimmt — wodurch dieser Datensatz nach Verarbeitung dieser Code-Blöcke vollständig abgearbeitet ist, und

5. anschließend werden die Code-Blöcke mit den (noch nicht aufgeführten) Codes des zweiten bis letzten Datensatzes im ersten Block abgearbeitet, danach die Code-Blöcke mit den Codes des ersten noch nicht vollständig abgearbeiteten Datensatzes im zweiten Code-Block usw. usf.

Kapitel 6

Benchmarking

Spieglein, Spieglein an der Wand,
Wer ist die Schönste im ganzen Land?

Frau Königin, Ihr seid die Schönste hier,
aber Sneewittchen hinter den sieben Bergen
ist tausendmal schöner als ihr!

Aus „Sneewittchen“

A software benchmark is a prescription for a set of measurements
to evaluate some category of software capability,
usually performance.
P. O'Neill (in [O'N94, p. 602])

Inhalt dieses Kapitels. In diesem Kapitel wird ein Vergleich von Objektidentifizierungs-Verfahren durchgeführt. Wir geben Qualitätskriterien für Objektidentifizierungs-Verfahren an. Es wird eine Spezifikation für eine *Benchmark für Objektidentifizierung* angegeben. Für drei Datenbanken werden drei Objektidentifizierungs-Verfahren evaluiert. Das Kapitel schließt mit einer kritischen Diskussion der Resultate.

Wir orientieren uns bei der Definition einer Benchmark für Objektidentifizierung an den Prinzipien für Benchmarks aus dem Datenbank-Bereich, wie die TPC-Benchmark des Transaction Processing Performance Council [TPC]. Diese Prinzipien sind nach J. Gray [Gra93]:

(1) Relevanz für die Anwendungsdomäne (*engl.: Relevancy*),

(2) Übertragbarkeit auf andere Hard- und Software-Plattformen (*engl.: Portability*),

(3) Skalierbarkeit bezüglich anwendungsrelevanter Parameter (*engl.: Scalability*), sowie

(4) Einfachheit, Verständlichkeit und Transparenz (*engl.: Simplicity*).

Diese vier Kriterien stellen den Ausgangspunkt für den Entwurf einer Benchmark für Objektidentifizierung dar: (1) Relevanz: Fehler in den Duplikaten der verwandten Daten sollen den für Realwelt-Daten typischen Variationen entsprechen (möglicherweise aber unterschiedlich je nach Anwendungsfall). Erfülltsein typischer Charakteristika von Realwelt-Daten, z.B. die Existenz semantischer Schlüssel, (2) Übertragbarkeit: Verwendung einer System-unabhängigen Implementierung, Vermeiden spezifischer Funktionalität von Datenbank Management Systemen (z.B. *work flows*, *trigger* oder *stored procedures*) in der allgemeinen Spezifikation,[1] (3) Skalierbarkeit: Adaptierbarkeit auf verschiedenartige, kleine und große Datenbanken (z.B. Speichervolumen/Datensatzzahl, Schemakomplexität, Anzahl und Art der Tabellen/Attribute), Adaptierbarkeit eines Verfahrens bezüglich Qualitätskriterien (z.B. Skalierbarkeit bzgl. der Fehlerraten oder der Performanz), (4) Einfachheit: Spezifikation des Testablaufes für Daten mit einfachen Schemata (z.B. für eine universale Relation), modulares Design einer Implementierung der Testumgebung (z.B. voneinander unabhängige Routinen für Stichprobenziehung und Qualitätsmessung).

Unterschiedliche Objektidentifizierungs-Verfahren, die auf eine Datenbank angewandt werden können, liefern unterschiedliche Ausgaben. Ein Verfahren liefert als Ergebnis der Objektidentifizierung eine Liste der ID's dubletter Paare von Elementen, ein anderes Dublettengruppen. Ein drittes Verfahren liefert zu den Paaren eine Bewertung, die einem Grad der Übereinstimmung entspricht. Wir wollen die Qualität von solchen Verfahren evaluieren. Dazu fassen wir ein Objektidentifizierungs-Verfahren als Realisierung einer Entscheidungsfunktion δ wie folgt auf:

Definition 6.1 (Objektidentifizierungs-Verfahren). A sei eine Datenbank. Ein Objektidentifizierungs-Verfahren für Instanzen a, b von Klassen C aus A (z.B. Datensätze einer universalen Relation A) schreiben wir als Entscheidungsfunktion δ, $(a, b) \mapsto \delta(a, b) \in [0, 1]$, die folgende Werte annimmt:

$$\delta(a,b) = 0 \iff a \text{ und } b \text{ als Duplikate klassifiziert sind,} \tag{6.1a}$$

$$\delta(a,b) = 1 \iff a \text{ und } b \text{ als Nicht-Duplikate klassifiziert sind,} \tag{6.1b}$$

$$0 < \delta(a,b) < 1 \iff \text{die Entscheidung für } (a,b) \text{ unbestimmt bleibt.} \tag{6.1c}$$

In Lim et al. [LSPR96] werden die in (6.1a) genannten drei möglichen Entscheidungen als *identical/non matching/undetermined* bezeichnet. Es sei bemerkt, daß durch Erhöhung der Zahl der unbestimmten Fälle der Fehler der Entscheidungsfunktion verringert werden kann. Deshalb erweist es sich für den Vergleich von Verfahren als sinnvoll, eine Schranke $\varepsilon \in [0, 1)$ anzugeben, so daß

$$|\{(a,b) \in A \times A : \ a < b \wedge 0 < \delta(a,b) < 1\}| \leq \varepsilon \cdot |A| . \tag{6.2}$$

ε wird 0 gesetzt, falls keine Datensatzpaare unbestimmt bleiben sollen und andernfalls größer Null.

[1] Nichtsdestotrotz kann diese Funktionalität natürlich für spezifische Anwendungen genutzt werden, aufgrund variierender Funktionalität der Systeme ist ein Vergleich dadurch jedoch erschwert.

Wir geben an, was wir unter einer *Benchmark für die Objektidentifizierung* verstehen und diskutieren im Anschluß, welche Anforderungen wir an solch eine Benchmark stellen.

Definition 6.2 (Objektidentifizierungs-Benchmark). Eine Benchmark für die Objektidentifizierung besteht aus:

- Einer Datenbank A, zu der eine vollständige *Same*-Relation angegeben ist,
- Einer Menge von (semantischen) Constraints $\mathcal{C}$ über die Datenbank A, sowie
- Einer Menge von Qualitätskriterien $\mathcal{Q}$, nebst einer Spezifikation ihrer Bestimmung/Berechnung für Objektidentifizierungs-Verfahren auf A.

Bemerkung 6.3 *(Realwelt-Benchmark).* Wir fordern von einer Benchmark *Relevanz*, d.h. unter anderem, daß die Fehler in den Daten der Benchmark dem Auftreten von Variationen in realen Datenbeständen entsprechen. Um diese Forderung zu erfüllen, plädieren wir für die *ausschließliche Verwendung von Realwelt-Datenbeständen* als Benchmark-Daten, da diese Variationen kaum adäquat mit einem fehlererzeugenden Zufalls-Prozeß nachbildbar sind. Insbesondere läßt sich für eine mit Fehlern aus einem Zufalls-Prozeß erzeugte Benchmark-Datenbank —sofern der Zufalls-Prozeß bekannt oder erkennbar ist— immer ein Objektidentifizierungs-Verfahren finden, das zwar optimal ist bezüglich der mit diesem Prozeß erzeugten Duplikate, aber für Realwelt-Daten eine deutlich schlechtere Anpassung aufweisen wird. Zum Erzeugen einer Benchmark-Datenbank kann man aber Daten verwenden, die einen Identifizierer aufweisen (z.B. die Social Security Number SSN für Personendaten in den USA oder die ISBN für Bücher) und den Identifizierer zwar für das Erstellen der Same-Relation ausnutzen, diesen dann aber nicht für das Lernen einer Klassifikation verwenden.

6.1 Qualitätskriterien für Objektidentifizierungs-Verfahren

In diesem Abschnitt betrachten wir Qualitätskriterien für Objektidentifizierungs-Verfahren. Als *Objektidentifizierungs-Verfahren* fassen wir ein Computerprogramm auf, das eine Datenbank als Input erhält und als Output eine Liste (oder Tabelle) liefert, die die als Duplikate klassifizierten/erkannten Datensätze enthält bzw. auf sie verweist (die Duplikate müssen in Gruppen zu mindestens zwei Datensätzen ausgegeben werden).

Für einen Vergleich kommen folgende Qualitätskriterien in Frage:

- **Quantitative Kriterien:** Korrektheit, Skalierbarkeit, Performanz sowie Aufwendungen und Kosten.
- **Qualitative Kriterien:** Verständlichkeit, Transparenz, Benutzbarkeit, Integrierbarkeit, Verläßlichkeit, Vollständigkeit, Robustheit, Erweiterbarkeit, Adaptierbarkeit und Flexibilität.

Die quantitativen Kriterien lassen sich für einen Anwendungsfall schätzen bzw. ermitteln. In den folgenden Abschnitten werden wir auf die Schätzung der ersten drei quantitativen Qualitätskriterien näher eingehen. Für die qualitativen Kriterien kann in der Regel eine Bewertung nur durch Experten in Form einer Beschreibung der Eigenschaften des Verfahrens bezüglich eines Kriteriums vorgenommen werden. Je nach Anwendbarkeit für ein Kriterium kann aber auch eine Punkteskala oder Rangfolge für die Bewertung verwandt werden.

- **Quantitative Kriterien:**
  - *Korrektheit*: Ermittlung der Fehlerraten sowie der Anzahl unbestimmter Datensatzpaare durch Testläufe,
  - *Skalierbarkeit* (1) bezüglich wachsender Datenbank-Größe $|A|$ oder (2) bezüglich Parallelisierbarkeit der Berechnungen,
  - *Kontrollierbarkeit* bezüglich der Anzahl unbestimmter Datensatzpaare und Fehlerschranken,
  - *Performanz*: Berechnungsaufwand, z.B. Komplexität der Algorithmen, Rechenzeit von Testläufen,
  - *Kosten*: Ausgaben für Inbetriebnahme und Betrieb, z.B. Hardware, Software-Lizenzen, Personalkosten (Administration, Pflege und Betreuung), sonstige Aufwendungen (z.B. für Installation, Datenbereinigung und Lernen von Klassifikationen).
- **Qualitative Kriterien:**
  - *Verständlichkeit und Transparenz* des Verfahrens (Beschreibung von Algorithmen, Heuristiken, etc.), Nachvollziehbarkeit der Ergebnisse,
  - *Benutzbarkeit* der Software, z.B. wie Benutzer-freundlich ist die Software oder wird Expertenwissen gebraucht (und wenn ja, welches),
  - *Integrierbarkeit* in vorhandene Software-Architekturen, z.B. inwieweit sind (standardisierte) Schnittstellen (*engl.: interfaces*) für den Daten-/Objekt-Austausch mit anderen Software-Komponenten vorhanden oder realisierbar, ist eine automatische Steuerung des Programms möglich (z.B. durch Trigger),
  - *Vollständigkeit* der Software, d.h., wird der gesamte Objektidentifizierungsprozeß durch die Software abgedeckt, oder sind ergänzende Module notwendig,
  - *Verläßlichkeit und Robustheit* der Verfahren, insbesondere den Einsatz im Produktivbetrieb betreffend (z.B. ist das Verfahren methodisch ausgereift und hat es sich in der Praxis bewährt),
  - *Erweiterbarkeit, Adaptierbarkeit und Flexibilität*: z.B. (1) besteht die Möglichkeit von automatischen und inkrementellen Updates (wie z.B. eine Modifikation der Entscheidungsregel durch neue Daten), (2) inwieweit ist eine Anpaßbarkeit an eine veränderte Datenmodellierung gegeben (z.B. zusätzliche Attribute), sowie

(3) können weitere Datenbanken nachträglich in den Identifikationsprozeß mit einbezogen werden.

Im Folgenden geben wir Maße für die quantitativen Qualitätskriterien Korrektheit, Kontrollierbarkeit und Performanz sowie deren Berechnung für Stichproben von Datensätzen an.

6.1.1 Korrektheit der Identifizierung

Die *Korrektheit* eines Objektidentifizierungs-Verfahrens ist bestimmt durch die Wahrscheinlichkeit, eine Fehlentscheidung zu fällen. Sie entspricht (für eine Tabelle A) dem Anteil der nicht erkannten Duplikate und der fälschlich als Duplikate klassifizierten Datensatzpaare aus A. Der Fehler setzt sich zusammen aus:

- Fehler erster Art, α*–Fehler*: Die Wahrscheinlichkeit, Duplikate nicht zu erkennen, sowie
- Fehler zweiter Art, β*–Fehler*: Die Wahrscheinlichkeit, Datensätze fälschlich als Duplikate aufzufassen.

Die α– und β*–Fehler* sind definiert durch

$$\alpha := P\big(\delta(a,b) = 1 \mid a \equiv b\big), \tag{6.3a}$$
$$\beta := P\big(\delta(a,b) = 0 \mid a \not\equiv b\big). \tag{6.3b}$$

Für eine Datensatz-Stichprobe mit einer vollständigen Same-Relation können wir die Fehler schätzen:

$$\hat{\alpha} := \hat{\mathbf{P}}\{\, \delta(a,b) = 1 \mid a \equiv b \,\} \tag{6.4a}$$
$$= \frac{|\,\{(a,b) \mid \delta(a,b) = 1 \wedge a \equiv b\}\,|}{|\,\{(a,b) \mid a \equiv b\}\,|},$$

$$\hat{\beta} := \hat{\mathbf{P}}\{\, \delta(a,b) = 0 \mid a \not\equiv b \,\} \tag{6.4b}$$
$$= \frac{|\,\{(a,b) \mid \delta(a,b) = 0 \wedge a \not\equiv b\}\,|}{|\,\{(a,b) \mid a \not\equiv b\}\,|}.$$

Falls für alle Datensatzpaare aus $A \times A$ eine Entscheidung gefällt wird, erhalten wir vier Fälle (vgl. Abb. 6.1):

1. **True Positives** TP: korrekt als Duplikate klassifizierte Paare,
2. **False Positives** FP: falsch als Duplikate klassifizierte Paare,
3. **True Negatives** TN: korrekt als Nicht-Duplikate klassifizierte Paare, sowie
4. **False Negatives** FN: falsch als Nicht-Duplikate klassifizierte Paare.

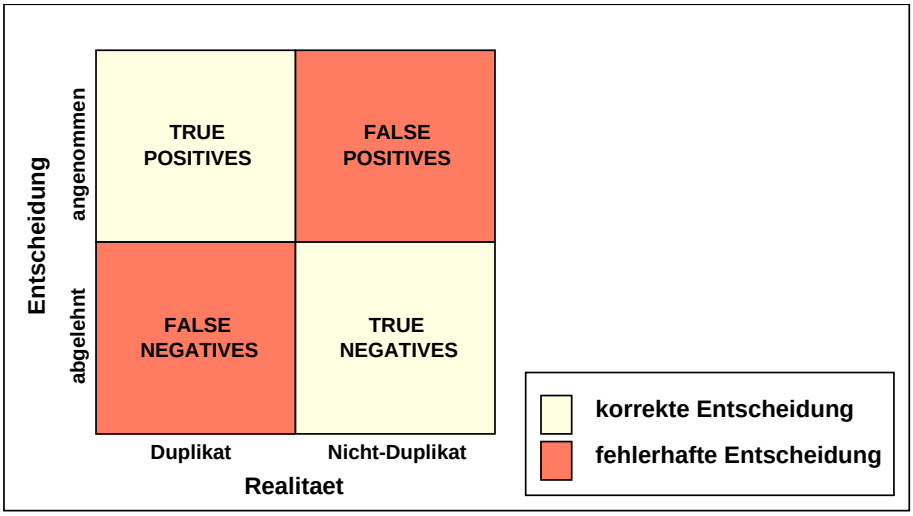


Abbildung 6.1: Entscheidung vs. Realität

α– und β–*Fehler* lassen sich darstellen als

$$\hat{\alpha} = \frac{FN}{TP + FN}, \quad \hat{\beta} = \frac{FP}{TN + FP}. \tag{6.5}$$

Es sind aber auch andere Maße für die Korrektheit ableitbar, wie z. B. *Precision* und *Recall*, die definiert sind durch

$$\begin{aligned} \mathit{precision} &:= \frac{TP}{TP + FP} \\ &= \hat{\mathbf{P}}\{\, a \equiv b \mid \delta(a,b) = 0 \,\} \\ &= \frac{|\,\{(a,b) \mid \delta(a,b) = 0 \wedge a \equiv b\}\,|}{|\,\{(a,b) \mid \delta(a,b) = 0\}\,|}, \end{aligned} \tag{6.6}$$

sowie

$$\begin{aligned} \mathit{recall} &:= \frac{TP}{TP + FN} \\ &= \hat{\mathbf{P}}\{\, \delta(a,b) = 0 \mid a \equiv b \,\} = (1 - \hat{\alpha}). \end{aligned} \tag{6.7}$$

Precision und *recall* können zur $[-\infty, 1]$–wertigen *match-Accuracy* zusammengefaßt werden, die von Melnik et al. [MGMR02] eingeführt wurde,

$$\text{match\_accuracy} := \mathit{recall}\left(2 - \frac{1}{\mathit{precision}}\right) = \frac{TP - FP}{TP + FN}\,. \tag{6.8}$$

Die *match_accuracy* mißt den Aufwand eines Benutzers, das Ergebnis eines Objektidentifizierungs-Verfahrens durch Hinzufügen fehlender und Löschen falscher Duplikate in das korrekte Ergebnis zu überführen, das durch eine vollständige Same-Relation gegeben ist. Ein geringer Korrekturaufwand entspricht einer hohen Korrektheit des Ergebnisses und ist bei Werten nahe 1 gegeben, während negative Werte darauf hindeuten, daß der Anteil der fälschlicherweise als richtig klassifizierten Paare überwiegt ($\mathit{precision} < 0.5$).

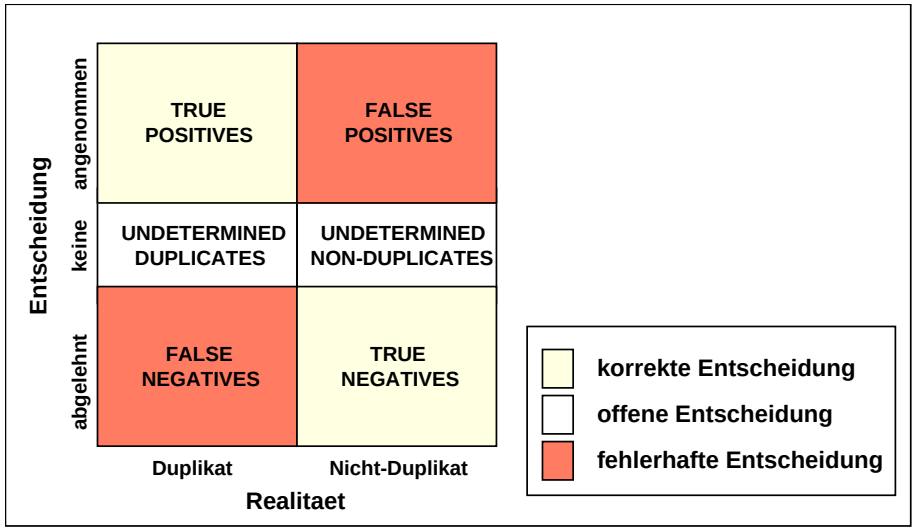


Abbildung 6.2: Entscheidung vs. Realität (mit unbestimmten Datensatzpaaren)

Bemerkung 6.4 *(Berücksichtigung der unbestimmten Paare).* Falls für $n \leq \varepsilon \cdot |A|$ Datensatzpaare aus $A \times A$ keine Entscheidung gefällt wurde, d.h. es gilt (6.2) mit $\varepsilon > 0$, erhalten wir zwei weitere Fälle (vgl. Abb. 6.2):

5. **Undetermined Duplicates** UD: Duplikate, für die keine Entscheidung gefällt wurde, und

6. **Undetermined Non-Duplicates** UN: Nicht-Duplikate, für die keine Entscheidung gefällt wurde.

Bei der Messung der hier eingeführten Maße für die Korrektheit ist zu beachten, daß Paare, für die die Entscheidung Duplikat/Nicht-Duplikat nicht gefällt wurde, in die Berechnung nicht eingehen. Der Vergleich von verschiedenen Verfahren macht also nur dann Sinn, wenn der Anteil der unbestimmten Paare jeweils durch dieselbe Schranke $\varepsilon > 0$ gemäß (6.2) begrenzt ist.

Alternativ können die unbestimmten Paare auch in allen Maßen als falsch klassifizierte Paare berücksichtigt werden (mit dem Superskript '*' markiert) oder die bei zufälliger Entscheidung zu erwartende Zahl der falsch klassifizierten Paare (d.h. falls jeweils eine Hälfte der unbestimmten Paare als Duplikate bzw. als Nicht-Duplikate klassifiziert werden und wird mit dem Superskript '+' markiert; der Faktor c_{UD} entspricht dem Anteil dubletter Paare unter den unbestimmten Paaren, $c_{UD} = \frac{UD}{UN+UD}$):

$$\hat{\alpha}^* := \tfrac{FN+UN}{TP+FN}, \quad \hat{\alpha}^+ := \tfrac{FN+(1-c_{UD})UN}{TP+FN}, \tag{6.9a}$$

$$\hat{\beta}^* := \tfrac{FP+UD}{TN+FP}, \quad \hat{\beta}^+ := \tfrac{FP+c_{UD}UD}{TN+FP}, \tag{6.9b}$$

$$precision^* := \tfrac{TP}{TP+FP+UD}, \quad precision^+ := \tfrac{TP}{TP+FP+c_{UD}UD}, \tag{6.9c}$$

$$recall^* := \tfrac{TP}{TP+FN+UD}, \quad recall^+ := \tfrac{TP}{TP+FN+c_{UD}UD}. \tag{6.9d}$$

Diese Werte liefern Schranken für α, β, *precision* und *recall*:

$$\hat{\alpha}^* \geq \hat{\alpha}^+ \geq \hat{\alpha},$$
$$\hat{\beta}^* \geq \hat{\beta}^+ \geq \hat{\beta},$$
$$precision^* \leq precision^+ \leq precision,$$
$$recall^* \leq recall^+ \leq recall.$$

Dieses Vorgehen führt aber zu deutlich schlechteren Werten der Korrektheitsmaße (pessimistischer Ansatz: alle unbestimmten Paare können falsch klassifiziert werden bzw. im Mittel die Hälfte der Paare), so daß in dieser Arbeit der optimistische Ansatz (alle unbestimmten Paare können noch richtig klassifiziert werden) verfolgt wird, wobei wir —wie eingangs in (6.2) dargestellt— die Zahl der unbestimmten Paare beschränken.

Bemerkung 6.5. Die Untersuchung der Korrektheit (Bestimmung der Fehlerraten) bei der Anwendung von Record Linkage auf Personendaten wurde in einigen Arbeiten untersucht, vgl. z.B. Burgess [Bur88], Adman et al. [ABB92], Bartlett et al. [BKWZ93], Belin et al. [Bel93, BR95], Scheuren et al. [SW93], oder Armstrong et al. [AM93].

6.1.2 Performanz und Skalierbarkeit

Wir betrachten Objektidentifizierungs-Verfahren, die die Form einer Entscheidungsfunktion und eines Selektors haben, d.h.

$$(\delta \circ \sigma). \tag{6.10}$$

Dann können wir den Aufwand zur Durchführung der Objektidentifizierung bestimmen durch

(a) Analytische Komplexitätsbestimmung (in groß-O-Termen, $O(\cdot)$), mit $n = |A'|$, für $A' \subset A$, sowie durch

(b) Laufzeitmessung für implementierte Objektidentifizierungs-Verfahren, wobei eine Vergleichbarkeit der Ergebnisse dadurch erreicht werden kann, daß die Verfahren sämtlichst für eine einzige Hard- und Software-Konfiguration implementiert und auf einem Rechner lauffähig sind.

Für die Aufwandsermittlung können wir davon ausgehen, daß der Aufwand der Berechnungen je geladenem Datensatzpaar näherungsweise konstant ist, wobei dieser konstante Wert c_δ jedoch von der gewählten Entscheidungsfunktion abhängt. Infolgedessen ist die Performanz hauptsächlich von dem gewählten Selektor σ und dessen algorithmischer Umsetzung abhängig. Deshalb genügt es bei der analytischen Komplexitätsbestimmung sowie bei der Laufzeitmessung, die Komplexität der Selektoren zu bestimmen und den Wert c_δ abzuschätzen.

Wir setzen uns mit der analytischen Komplexitätsbestimmung auseinander.

Die analytische Komplexitätsbestimmung

Wir geben den Aufwand in Vielfachen des Aufwandes eines Datenzugriffes aus A an, d.h. wir setzen als Einheit den Aufwand für das Lesen eines Datensatzes $a \in A$ mit vorgegebenem Primärschlüssel-Wert (z.B. des ID-Attributes).

Der konstante Wert c_δ entspricht den je gebildetem Datensatz-Paar notwendigen Berechnungen (für Konversion, Vergleich und Klassifikation) sowie Lese- und Schreiboperationen (z.B. Datenzugriffe auf Festspeicher-Medien oder Datenbank), d.h dem Aufwand der Klassifikation *eines* Paares. Die Performanz ergibt sich dann aus der Zahl der zu bildenden Paare $|\sigma(A \times A)|$ multipliziert mit c_δ zuzüglich der Anzahl der zu ladenden Datensätze aus A, $\#load(A, \sigma, n_{\max})$, wobei wir eine Beschränkung des Arbeitspeichers annehmen (d.h. es können maximal $n_{\max}$ Datensätze gleichzeitig im Arbeitsspeicher geladen sein). D.h. die Performanz ist gegeben durch

$$\text{performance} = c_\delta \cdot |\sigma(A \times A)| + \#load(A, \sigma, n_{\max}). \tag{6.11}$$

Es sei $A_\sigma = \{a \in A | \exists b \in A : ((a, b) \in \sigma(A \times A) \vee (b, a) \in \sigma(A \times A))\}$ die Teilmenge aller Datensätze aus A, die in mindestens einem Paar enthalten sind. Ohne Beschränkung des Arbeitspeichers, d.h. mit $n_{\max} \geq |A_\sigma|$, erhalten wir eine untere Schranke für (6.11),

$$\text{performance} \geq c_\delta \cdot |\sigma(A \times A)| + |A_\sigma| = \left(\frac{|A_\sigma|}{|\sigma(A \times A)|} + c_\delta \right) \cdot |\sigma(A \times A)|, \tag{6.12}$$

da die gesamte Tabelle A_σ in den Arbeitspeicher geladen werden kann. Für $n_{\max} = 2$, d.h. bei minimalem Arbeitspeicher erhalten wir eine obere Schranke für (6.11),

$$\text{performance} \leq c_\delta \cdot |\sigma(A \times A)| + \frac{1}{2} \cdot 2 \cdot |\sigma(A \times A)| = (1 + c_\delta) \cdot |\sigma(A \times A)|, \tag{6.13}$$

da $|\sigma(A \times A)|$ Paare zu bilden sind, die jeweils aus zwei Datensätzen bestehen und da für jedes Paar *höchstens einer* der beiden Datensätze mehrfach geladen werden muß. Mit der Selektionsrate $\text{sel}(\sigma) = \frac{|\sigma(A \times A)|}{|A|^2}$ eines Selektors (vgl. Definition 5.12 auf Seite 129) erhalten wir

$$\begin{aligned} \text{performance} &= (\theta + c_\delta) \cdot |\sigma(A \times A)| && (6.14) \\ &= (\theta + c_\delta) \cdot \text{sel}(\sigma) \cdot |A|^2, && (6.15) \end{aligned}$$

mit $\theta = \theta(n_{\max}) \in [0, 2]$, typischerweise[2] mit $\theta \in [0, 1]$. (6.12) liefert uns die untere Schranke für θ,

$$\theta \geq \frac{|A_\sigma|}{|\sigma(A \times A)|}, \tag{6.16}$$

mit $\theta = 0$ falls $\sigma(A \times A) = \emptyset$. Zusammengefaßt erhalten wir folgenden

Satz 6.6. *($\delta \circ \sigma$) sei ein Objektidentifizierungs-Verfahren, $n = |A|$.*

[2]Der Fall $\theta > 1$ tritt nur auf, wenn $|A_\sigma| > |\sigma(A \times A)|$ gilt, d.h. wenn es mehr Datensätze als daraus gebildete Paare in σ gibt — eine eher pathologische Situation für Objektidentifizierungs-Probleme ohne globale Identifizierer.

(i) Falls die Selektionsrate von σ für große A näherungsweise konstant ist, d.h. falls

$$\mathrm{sel}(\sigma, A) \approx \mathrm{sel}(\sigma, A')$$

für $A' \subset A$, $|A'| \geq \frac{1}{2}|A|$ gilt, dann hat $(\delta \circ \sigma)$ quadratischen Aufwand $O(n^2)$.

(ii) Falls

$$|\sigma(A \times A)| \approx |\sigma(A' \times A')|$$

für $A' \subset A$, $|A'| \geq \frac{1}{2}|A|$ näherungsweise konstant ist, dann hat $(\delta \circ \sigma)$ linearen Aufwand $O(n)$.

Daraus folgen unmittelbar

Korollar 6.7. *Ein Objektidentifizierungs-Verfahren mit einem auf der relationalen Algebra gemäß Formel* (5.6) *auf Seite 125 basierenden Selektor σ hat quadratischen Aufwand.*

Korollar 6.8. *Ein Objektidentifizierungs-Verfahren, das auf einem Selektor basiert, bei dem die Zahl der Paarbildungen je Datensatz $a \in A$ begrenzt ist, d.h. es gibt ein $k \in \mathbb{N}$, $k \ll n$, so daß gilt*

$$\forall a \in A : |\{b \in A : ((a,b) \in \sigma \vee (b,a) \in \sigma\}| \leq k,$$

hat linearen Aufwand $O(k \cdot n)$.

Die Voraussetzung des Korollars 6.8 ist z.B. für einen k-NN-Selektor erfüllt, wie er beispielsweise für die k-Nächsten Nachbarn (vgl. Abschnitt 3.3.4 auf Seite 91) verwandt wird.

Bemerkung 6.9 *(Skalierbarkeit).* Aus den beiden Korollaren läßt sich ersehen, daß Objektidentifizierungs-Verfahren nur dann linear skalierbar für große Datenbestände sind, wenn sie die Zahl der Paarbildungen je Datensatz auf maximal k beschränken, für ein $k \ll n$. Durch diese Beschränkung der Zahl der Paarbildungen wird jedoch in der Regel der Anteil nicht entdeckter Duplikate (der α-Fehler) mit der Datenbankgröße steigen.

Aufgrunddessen ist auch für große Datenbestände (z.B. mehrere Millionen Datensätze) der Einsatz anderer, z.B. quadratisch skalierender Selektoren erforderlich, um niedrige Fehlerraten zu erreichen.

6.1.3 Kontrollierbarkeit der Fehlerraten

Falls ein Objektidentifizierungs-Verfahren für Datensatzpaare einen *Decision*-Wert δ aus dem Intervall [0,1] liefert, kann man eine Kontrollierbarkeit der Fehlerraten erreichen. Durch Verwendung von Grenzen $0 < \mathit{Lowerbound} \leq \mathit{Upperbound} < 1$ erhalten wir eine Entscheidungsfunktion der Gestalt

$$\delta'(a, b; \mathit{Lowerbound}, \mathit{Upperbound}) := \begin{cases} 0 & \mathit{Decision}(a,b) < \mathit{Lowerbound}, \\ 1 & \mathit{Decision}(a,b) > \mathit{Upperbound}, \\ - & \text{otherwise.} \end{cases} \tag{6.17}$$

wobei für Werte von $Decision(a, b)$ mit $Lowerbound \leq Decision(a, b) \leq Upperbound$ keine Entscheidung gefällt wird (in (6.17) mit '-' gekennzeichnet). Dadurch kann mit der Variation der Grenzen im Intervall [0,1] eine Reskalierung der Fehlerraten erreicht werden, d.h. unter Inkaufnahme eines höheren β-Fehlers kann der α-Fehler verringert werden.

Bemerkung 6.10 *(Berücksichtigung von Kosten für Fehler).* Durch die Kontrollierbarkeit der Fehlerraten wird es möglich, den α-Fehler (bei Grenzen nahe 1) bzw. den β-Fehler (bei Grenzen nahe 0) sehr gering zu halten, oder, alternativ dazu, beide Fehler zu beschränken (bei Grenzen nahe 0.5). Durch diese Möglichkeit der Skalierung lassen sich Klassifikationsmodelle optimieren, die separate Kosten für die beiden Fehler berücksichtigen, beispielsweise das Kostenmodell[3]

$$\text{costs} = \mu \cdot \alpha + \nu \cdot \beta, \tag{6.18}$$

mit $\mu, \nu \in \mathbb{R}_{\geq 0}$.

6.2 Die Spezifikation des Tests

6.2.1 Die Testumgebung

Für die Testumgebung ist eine Tabelle A zusammen mit einer für A vollständigen Same-Relation *Same* gegeben (oder zumindest für eine Teilmenge $A' \subset A$). Für den Test verwenden wir drei Datenbestände:

- Wohnungsanzeigen aus den Online-Editionen der Tageszeitungen *Tagesspiegel* und *Berliner Morgenpost*,
- Kundendatensätze eines deutschen Unternehmens, sowie
- Datensätze aus einem Bibliothekskatalog.

Der Vergleich von Objektidentifizierungs-Methoden wird in einer in *Visual Basic for Applications* (VBA) realisierten Testumgebung durchgeführt. Diese Testumgebung ermöglicht die Metadaten-gesteuerte Ausführung von Transaktionen, z.B.

- Bestimmung der semantischen Constraints (vgl. Kapitel 4)
- Sampling (geschichtete Stichprobenziehung von Datensätzen und Datensatzpaaren, vgl. Abschnitt 6.2.2 auf Seite 152),
- Konvertierung von Attributen mittels benutzerdefinierter Funktionen (vgl. Abschnitt 2.3 auf Seite 52),
- Vergleich von Attributen mittels benutzerdefinierter Funktionen einschließlich Speicherung der Vergleichswerte in einer Datensatzpaar-Tabelle (vgl. Abschnitt 2.4 auf Seite 59),

[3] Zu Kostenmodellen siehe auch die Arbeit von Verykios et al. [VME02].

- Klassifikation von Datensatzpaaren (z.B. basierend auf Entscheidungsbäumen, Assoziationsregeln und Record Linkage)
- Messung von Qualitätskriterien für Verfahren (Korrektheit und Performanz, vgl. Abschnitt 6.1 auf Seite 143)
- Erzeugung von Index-Tabellen (z.B. für NGrams, vgl. Abschnitt 5.2.1, Seite 136ff)
- Vorauswahl (*engl.: Preselection*) von Datensatzpaaren basierend auf Bedingungen an Attributwerte und Einträge in Index-Tabellen.

Die Metadaten werden in Tabellen gespeichert und bei der Ausführung der in der Datenbank gespeicherten Routinen (stored procedures) eingelesen und verarbeitet. Die gespeicherten Routinen verwenden eingebettetes SQL (embedded SQL) für die Daten-Selektion und Modifikation (Ausführung von Lösch-/Einfüge- und Aktualisierungs-Transaktionen). Die dazu verwandten SQL-Statements werden dynamisch unter Verwendung der Metadaten generiert.

6.2.2 Die Stichprobenziehung

Es sei eine Tabelle $A =$ `A(ID,X1,...,Xk)` mit $n = |A|$ Datensätzen $a =$ `(a.ID, a.X1, ...,a.Xk)` gegeben, die Informationen über Realwelt-Objekte aus einer Datenbank enthalte.

Bevor wir auf die Stichprobenziehung eingehen können, müssen wir kurz auf die in Abschnitt 4.1 auf Seite 104 eingeführte *Same*-Relation eingehen. Eine *Same*-Relation enthält Paare von Datensätzen, die als dublett zu betrachten sind, d.h. $Same \subset A \times A$. Wir können aus jeder *Same*-Relation eine *optimierte Same-Relation* entsprechend Definition 4.8 auf Seite 106 bilden. Zur Vereinfachung der Notation und des effizienten Zugriffs auf die Daten betrachten wir im Folgenden *nur* optimierte *Same*-Relationen. Die *Same*-Relation benötigen wir sowohl für die Stichprobenziehung als auch für die Bewertung der Korrektheit einer gelernten Klassifikation.

Für das Lernen einer Klassifikation gewinnen wir Stichproben mit *Paaren von Datensätzen* aus A. Um einen bestimmten Mindest-Anteil dubletter Paare in der Stichprobe sicherzustellen, müssen wir eine geschichtete Stichprobenziehung durchführen. $\sigma(A \times A) \subset A \times A$ bezeichne den Selektor, der für die Vorauswahl von Datensatzpaaren bestimmt wurde, $\sigma(\{a\} \times A)$ bezeichne dessen Restriktion für einen Datensatz $a \in A$,

$$\sigma(\{a\} \times A) := \{b \in A \backslash \{a\} \mid (a,b) \in \sigma(A \times A) \vee (b,a) \in \sigma(A \times A)\} \quad (6.19)$$

$N \leq |\sigma(A \times A)|$ bezeichne die Stichprobengröße, $N_0 < N$ die Anzahl zu ziehender dubletter Paare in der Stichprobe ($N_0 \leq |TC(Same)|$). Die Ziehung einer Stichprobe von Datensatz-Paaren aus A gestaltet sich dann wie folgt:

Stichprobenziehung 1

1. *Zufällig gebildete Paare*: Wiederhole solange, bis $(N - N_0)$ verschiedene Paare gezogen wurden:
   (a) Zufällige Ziehung eines Datensatzes $a \in A$,
   (b) Zufällige Ziehung eines Datensatzes $b \in \sigma(\{a\} \times A)$ (fahre fort mit (1a) falls $\sigma(\{a\} \times A) \neq \emptyset$),
   (c) Einfügen des Paares (a, b) in P, falls $a < b$ und von (b, a) andernfalls (falls das Paar noch nicht in P enthalten ist).
2. *Zufällig gezogene dublette Paare*: Führe durch:
   (a) Bildung des transitiven Abschlusses der Same-Relation $TC(Same)$,
   (b) Zufällige Ziehung von N_0 verschiedenen Datensatz-Paaren $(a, b) \in TC(Same) \backslash \{P\}$,
   (c) Einfügen der gezogenen Paare in P.

Außerdem wird eine Markierung der dubletten Paare in $(a, b) \in P$ durch Setzen eines Attributes `Same` mit dem Wert 0 (d.h. $(a, b) \in TC(Same)$), und der nicht-dubletten Paare in P mit dem Wert 1 vorgenommen.

Es sei bemerkt, daß im Allgemeinen beim zufälligen Bilden von Paaren in Schritt 1 innerhalb einer Vorauswahl σ nur sehr wenige dublette Paare gebildet werden, weshalb ein zusätzliches Ziehen nicht-dubletter Paare und eine Verringerung des Anteiles N_0 um die Zahl bereits gezogener Duplikate in der Regel nicht erforderlich ist. Lediglich falls die Vorauswahl σ bereits einen hohen Anteil (z.B. 30%) dubletter Paare enthält, ist folgendes Vorgehen zu empfehlen. Die obige Stichprobenziehung wird wie folgt modifiziert:[4]

[4] Aus Effizienzgründen kann die Überprüfung in Punkt (1c), ob zufällig ein dublettes Paar gebildet wurde, auch unterlassen werden, wenn dafür die in (2) zu ziehenden dubletten Paare um die bereits gezogenen reduziert werden und ein zusätzliches Ziehen nicht-dubletter Paare umgesetzt wird.

Stichprobenziehung 1a

1. *Zufällig gebildete Paare*: Wiederhole solange, bis $(N - N_0)$ verschiedene *nicht-dublette* Paare in P enthalten sind:
   (a) Zufällige Ziehung eines Datensatzes $a \in A$,
   (b) Zufällige Ziehung eines Datensatzes $b \in \sigma(\{a\} \times A)$ (fahre fort mit (1a) falls $\sigma(\{a\} \times A) \neq \emptyset$),
   (c) Einfügen des Paares (a, b) in P, falls $a < b$ und $(a, b) \notin TC(Same)$ bzw. von (b, a) falls $a > b$ und $(b, a) \notin TC(Same)$ (nur wenn noch nicht in P enthalten).
2. *Zufällig gezogene dublette Paare*: Führe durch:
   (a) Bildung des transitiven Abschlusses der Same-Relation $TC(Same)$,
   (b) Zufällige Ziehung von N_0 verschiedenen Datensatz-Paaren $(a, b) \in TC(Same) \backslash \{P\}$,
   (c) Einfügen der gezogenen Paare in P.

Bemerkung 6.11. Aus einer für eine Tabelle A *vollständigen Same- Relation* lassen sich —mit der Bildung des transitiven Abschlusses $TC(Same)$— alle Paare von Duplikaten gewinnen. Im Folgenden gehen wir davon aus, daß alle Duplikate in A bekannt sind, d.h. das eine *vollständige Same-Relation* gegeben ist. Falls keine vollständige *Same*-Relation für A verfügbar ist, muß entweder eine Datensatz-Stichprobe $A' \subset A$ zur Messung der Korrektheit gezogen werden, (für die dann manuell eine vollständige *Same*-Relation erstellt werden muß) oder jede gezogene Stichprobe von Paaren manuell auf dublette Paare geprüft werden (d.h. die vollständige *Same*-Relation wird stückweise gebildet).

Bemerkung 6.12 *(Notwendigkeit der Vorauswahl).* Bei der Ziehung einer Stichprobe P von Datensatzpaaren aus einer Datensatz-Stichprobe $S \subset A$ wird der Stichprobenumfang N und der Anteil dubletter Datensatzpaare gewählt N_0/N (z.B. 20%), der gezogen werden soll. Der Anteil der nicht-dubletten Datensatzpaare setzt sich aus einer geschichteten Stichprobe zusammen. Ein frei wählbarer Anteil (z.B. 5%) besteht aus Paaren zufällig gezogener Datensätze $a, b \in S$, die nur mit geringer Wahrscheinlichkeit ähnliche oder übereinstimmende Werte in den Attributen aufweisen. Der verbleibende Anteil nicht-dubletter Datensatzpaare wird unter Berücksichtigung von geeigneten Selektoren σ_i aus $S \times S$ gezogen.

Die Vorauswahl von Datensatzpaaren ist notwendig, da sich aus einer Stichprobe mit N Datensätzen $\frac{1}{2}(N-1)(N-2) \approx \frac{1}{2}N^2$ geordnete Datensatzpaare bilden lassen, wobei der Anteil von dubletten Datensatzpaaren typischerweise deutlich kleiner als N ausfällt.[5] Wenn man sich auf die

[5] Der Anteil dubletter Datensatzpaare hängt hauptsächlich von der Anzahl *mehrfacher* Duplikate in A ab, falls z.B. $\frac{N}{2}$ Datensätze sich auf ein und dieselbe Äquivalenzklasse von Realwelt-Objekten beziehen, beträgt der Anteil mit ca. $\frac{1}{32}N^2$ geordneten Datensatzpaaren ein Sechzehntel der insgesamt bildbaren Datensatzpaare. Falls jedoch *keine* mehrfachen Duplikate in A enthalten sind, erhalten wir maximal $\frac{N}{2}$ geordnete Datensatzpaare, das ist lediglich ein winziger Bruchteil aller $\frac{1}{2}N^2$ Paare.

rein zufällige Ziehung von Datensatzpaaren beschränken würde, müßte entweder der Stichprobenumfang sehr groß gewählt werden oder auf eine repräsentative Auswahl nicht-dubletter Paare verzichtet werden. Eine gute Repräsentativität von Paaren läßt sich *ausschließlich* durch große Stichproben erreichen, da wir Attribute mit identifizierenden Informationen betrachten, die über eine hohe Trennschärfe verfügen. Jedes Paar von Attributwerten wird dann nur für einige wenige Datensatzpaare auftreten, so daß bei kleineren Stichproben einige (möglicherweise sogar fast alle!) Attributwert-Kombinationen in *keinem* der gezogenen Paare auftreten—was die grundlegende Anforderung für die Repräsentativität einer Stichprobe verletzt.

Bei einem großen Stichprobenumfang —etwa $N \approx \frac{1}{40}(|A|)^2$ für eine fünfprozentige Auswahl— sind nur sehr wenige dublette Paare in der Stichprobe enthalten (z.B. sind für $|A| = 1.000$ bei einer fünf-prozentigen Stichprobe und 200 nicht mehrfachen Duplikaten in A weniger als 1% dublette Paare in P enthalten). Bei kleineren Stichprobenumfängen (z.B. $N \approx |A|$) läßt sich eine repräsentative Auswahl von nicht-dubletten Paaren nicht sicherstellen. Beide dieser Alternativen erweisen sich als problematisch für das Lernen einer Klassifikation:

- Große Stichproben (z.B $N \geq 10^6$) sind in der Regel problematisch, da nur wenige Klassifikationsverfahren große Stichproben verarbeiten können. Andererseits sind die Ergebnisse eines Klassifikationsverfahrens bei einem geringen Anteil *positiver Beispiele*—der dubletten Paare— und einem sehr hohen Anteil *negativer Beispiele*—der nicht-dubletten Paare— zumeist nicht zufriedenstellend, d.h. sie tendieren zu hohen Fehlerraten der Klassifikation der unterrepräsentierten positiven Beispiele.

- Nicht repräsentative Stichproben können zu Verzerrungen in der gelernten Klassifikation führen, so daß ebenfalls mit hohen Fehlerraten bei der Klassifikation unbekannter Beispiele zu rechnen ist.

Um die oben geschilderten Probleme zu vermeiden, nutzen wir Zusatzwissen über die Daten aus. Das Ziel besteht darin, die Anzahl der zu bildenden Paare drastisch zu verringern. Grundlage für dieses Vorgehen ist die Beobachtung, daß für das Erkennen von Duplikaten nur diejenigen Paare relevant sind, die überhaupt als Duplikate in Frage kommen. Bei der Stichprobenziehung wird eine Repräsentativität der Paare bezüglich dieser potentiellen Duplikate angestrebt. In der Regel kann für einen Großteil der Paare a priori ausgeschlossen werden, daß sie Duplikate sein können. Dieses sei an drei typischen Fällen verdeutlicht:

- **Semantischer Schlüssel:** Falls zugesichert werden kann, daß für Duplikate *immer* die Übereinstimmung des Wertes eines Attributes gilt, so müssen nur solche Paare gebildet werden.

- **Semantischer Differenzierungs-Schlüssel:** Wenn bekannt ist, daß ein (mehrdimensionales) Toleranzintervall für metrisch skalierte Attri-

butmengen[6] von Duplikaten eingehalten wird, müssen nur Paare gebildet werden, die dieses Intervall einhalten.

- **Approximative Schlüssel/Differenzierungs-Schlüssel:** Wenn eine Attributmenge die Eigenschaft eines semantischen Schlüssels/Differenzierungs-Schlüssels nur mit der confidence und accuracy bzw. anticonfidence der Attributmenge erfüllt.

Die Vorauswahl realisieren wir mittels der in Abschnitt 5 auf Seite 123 eingeführten Selektoren.

6.2.3 Ablauf eines Tests

Wir beschreiben den Ablauf eines Tests von Verfahren bezüglich der Korrektheit (Fehlerraten), bei denen die Lern- und Test-Stichproben Datensatzpaar-Stichproben sind.

- Stichprobenziehung
  1. Geschichtete Ziehung von Datensatzpaar-Stichproben P_i, $i = 1, \ldots, n$ mit vorgegebener Stichprobengröße N_i, die einen Mindestanteil dubletter Paare enthalten,
  2. Zerlegung von P_i in jeweils eine Lern- und Teststichprobe L_i, T_i (mit $P_i = L_i \cup T_i$ und $L_i \cap T_i = \emptyset$),
  3. Berechnung und Speicherung der Vergleichswerte der in den Datensatzpaar-Stichproben L_i, T_i enthaltenen Paare.
- Lernen und Testen von Verfahren (Ansteuerung externer Programme)
  1. Einlesen des Klassifikationsmodells (d.h. der zu setzenden Parameter) aus der Metadaten-Tabelle `ClassificationModels`,
  2. Export der Daten der Lernstichproben L_i und Erzeugung evtl. zusätzlicher Dateien einer Datensatzpaar-Stichprobe im erforderlichen Format (z.B. Generierung einer Kontingenztabelle und einer Modell-Datei für Record Linkage, Kodierung von Attribut-Wert-Kombinationen als Items für AssoClass),
  3. Aufruf/Ansteuerung des externen Programmes (z.B. apriori.exe) mit den exportierten Dateien (beispielsweise durch Ausführung einer Stapel-Datei (*.bat) automatisiert),
  4. Einlesen der Ausgabe-Daten, Transformation sowie Speicherung des Ergebnisses,
  5. Ableiten der Entscheidungsregel der Klassifikation sowie deren Anwendung auf die komplementäre Teststichprobe T_i,
  6. Berechnung und Speicherung der Korrektheitsmaße der Klassifikation auf T_i.

[6] Eine Attributmenge X ist metrisch skaliert, wenn ein Abstandsmaß $d : \mathrm{dom}(X) \times \mathrm{dom}(X) \to \mathbb{R}_{\geq 0}$ gegeben ist.

Die verwandten Vergleichsfunktionen, die Bedingungen für (relationale) Selektoren sowie die untersuchten Klassifikationsmodelle (d.h. ihre Parametrisierung) sind als Metadaten modelliert, so daß eine Anpassung (z.B. Hinzunahme weiterer Vergleiche oder eines neuen Klassifikationsmodells) einfach und transparent ist. In der Testumgebung sind diese Metadaten in Form der Tabellen

`Comparisonfunctions`, `Conditions` und `ClassificationModels`

in der Datenbank gespeichert. Beim Test werden diese Daten aus den Tabellen ausgelesen und mit ihnen die entsprechenden Parametrisierungen vorgenommen.

6.2.4 Die Klassifikationsmodelle für die einzelnen Verfahren

6.2.4.1 Das Entscheidungsbaumverfahren

Für das Entscheidungsbaumverfahren verwenden wir die folgenden drei Maße zur Berechnung eines Entscheidungsbaumes:

- *infgain*: Informationsgewinn basierend auf der Entropie (*engl.: Information Gain*),
- *infgr*: Informationsgewinnverhältnis basierend auf der Entropie (*engl.: Information Gain Ratio*), sowie
- *gini*: Minimierung des Gini-Indexes.

Das verwandte Maß wird im *model*-Abschnitt des Klassifikationsmodells angegeben, z.B. `measure={infgain}`. Jeder erzeugte Entscheidungsbaum kann zusätzlich beschnitten werden (*engl.: pruned*), um die Zahl der Entscheidungsregeln und damit die Zahl der Partitionen zu verringern, Auswahl der Option im *pruned*-Abschnitt, z.B. `pruned={yes}`.

Als Software setzen wir die von Chr. Borgelt entwickelte und frei erhältliche Software ein [Bor03b], bei der einige Optionen gegeben sind, beispielsweise das zur Baumerzeugung verwandte Maß betreffend.[7] Die Skalen der Attribute bzw. ihre Wertebereiche (*engl.: domains*) wurden für die Vergleichs-Attribute `Rent`, `Size`, `Phone` sowie `FullText` ordinal ($\mathbb{Z}$) und für alle anderen Attribute nominal gewählt (Dazu wurde die automatisch erzeugte Datei mit den Wertebereichen einer Lernstichprobe (Datei-Extension *.dom) entsprechend modifiziert).

6.2.4.2 Record Linkage

Für die Gewinnung der Entscheidungsregel δ berechnen wir den Likelihood-Quotienten $\lambda(r)$ eines Vergleichwert-Vektors r durch

$$\hat{\lambda}(r) = \frac{\hat{\mathbf{P}}\{\, r = f(a,b) \mid a \equiv b \,\}}{\hat{\mathbf{P}}\{\, r = f(a,b) \mid a \not\equiv b \,\}}. \tag{6.20}$$

[7] Es sei bemerkt, daß bei dem im Softwarepaket IBM Intelligent-Miner 8 verfügbaren Entscheidungsbaumverfahren ein Baum nur mit einem einzigen Maß —dem *Informationsgewinn*— berechnet werden kann.

Hierfür sind die beiden bedingten Wahrscheinlichkeiten

$$\hat{\mathbf{P}}\{\, r = f(a,b) \mid a \diamond b \,\}\,,\ \diamond \in \{\equiv, \not\equiv\}$$

auf der Lernstichprobe zu schätzen. Da wir nominale Skalen annehmen, ist eine mehr-dimensionale Multinomialverteilung (z.B. 4–10 Dimensionen) zu schätzen. Durch Modellrestriktionen liefert die Schätzung für (relativ) kleine Stichprobengrößen brauchbare Ergebnisse. Wir teilen die Lernstichprobe L_i auf in zwei Stichproben L_i^+, L_i^-, mit

$$L_i^+ = \{(a,b) \in L_i \mid a \equiv b\}\,, \quad L_i^- = \{(a,b) \in L_i \mid a \not\equiv b\}\,.$$

Für die Schätzung der Multinomialverteilung auf den Stichproben L_i^+, L_i^- verwenden wir die frei erhältliche Software LEM [Ver97]. Dazu spezifizieren wir hierarchische log-lineare Modelle, in denen die maximalen Interaktionen zwischen Attributen angeben werden (bzw. die Haupteffekte im Falle von 1-Faktor-Interaktionen). Bei einer hohen Dimensionalität (bis zu 13 Attribute für die Wohnungsdaten) sind für ein log-lineares Modell eine große Zahl an Parametern zu schätzen. Infolgedessen müssen wir in den Modellen jeweils eine Auswahl von Attributen treffen (Es sei bemerkt, daß eine Attribut-Selektion für das Entscheidungsbaumverfahren nicht erforderlich ist.). Außerdem können wir nicht beliebig hohe k–Faktor-Interaktionen zwischen Attributen aufgrund der daraus resultierenden hohen Zahl an Parametern betrachten.

Die Werte der Variablen *model*, *attributes*, *alpha*, *beta* und *undetermined* bestimmen die Parametrisierung von Record Linkage wie folgt:

- *model*: Das verwandte log-lineare Modell. Es wird ein Modell spezifiziert, das alle 1–, 2– oder 3–Faktor-Interaktionen der beteiligten Attribute (z.B. `model={1-way}`) enthält.
- *attributes*: Die in einem Modell verwandten Attribute, respektive ihre Nummern[8] in der Meta-Daten-Tabelle `ComparisonFunctions` (siehe z.B. die Tabelle 6.1 auf Seite 162 für die Wohnungsdaten),
- *alpha, beta, undetermined*: Angabe von Niveaus für den α–Fehler sowie für den β–Fehler oder den maximalen Anteil unbestimmter Paare *undetermined*, bei allen Klassifikationsmodellen für Record Linkage wurde die Parametrisierung `alpha={0.01}`, `undetermined={0}` verwandt.

6.2.4.3 AssoClass—Klassifikation mit Assoziationsregeln

Für die Klassifikation mit Assoziationsregeln extrahieren wir zunächst aus der Lernstichprobe Assoziationsregeln mittels des Apriori-Algorithmus (vgl. Agrawal [AIS93]) mit einem Item in der *Regel-Folgerung* (*engl.: head*) und beliebig vielen konjunktiv verknüpften Items in der *Regel-Bedingung* (*engl.:*

[8] Die Verwendung einer Nummer anstatt der Attributnamen im Modell vereinfacht die automatische Verarbeitung; die Software LEM erlaubt zudem nur Variablenbezeichner mit maximal drei Zeichen.

body). Als Software verwenden wir dazu die Implementierung von C. Borgelt [Bor03a], die für verschiedene Plattformen frei verfügbar ist. Wir erzeugen folgende Regelsätze (durch entsprechende Parametrisierung):

1. Generierung aller Regeln mit bis zu 5 Items (Defaultwert bzw. Option -n5) mit dem klassifikatorischen Attribut *Same* als *head* (Restriktion ist in der Datei `Same.itm` angegeben), die mindestens 0.5% Support und eine Confidence größer als 90% haben (Option -c90s0.5).

2. Extraktion aller Regeln mit ein bis drei Items (Option -n3) in denen das Attribut *Same* nicht —also weder in der Bedingung noch in der Folgerung einer Assoziationsregel— enthalten ist (Restriktion ist in der Datei `Other.itm` angegeben), die mindestens 1% Support und eine Confidence größer als 85% haben (Option -c85s0.5).

In der Datei `items.dat` sind zeilenweise die Items je Transaktion (in unserem Fall die Attribut-Wert-Kombinationen der Datensätze aus L_i) enthalten und in den Dateien `Same.itm` sowie `Other.itm` sind die Restriktionen bezüglich der Attribute in den zu extrahierenden Regeln gesetzt. Als Kommandozeilen geben wir also an (durch Verwendung der Option -a erhält man den Support der Regeln in absoluten Werten):

```
apriori.exe -ac90s0.5 items.dat aprules1.txt Same.itm
apriori.exe -ac85s1n3 items.dat aprules2.txt Other.itm
```

Die Dateien `aprules1.txt` sowie `aprules2.txt` enthalten dann die extrahierten Assoziationsregeln.

Die erzeugten Regeln werden eingelesen und in einer temporären Tabelle gespeichert. Aufgrund der unterschiedlichen Zahl an Datensätzen in L_i) mit *Same*= 0 (die Anzahl bezeichnen wir mit c^0)und *Same*= 1 (bezeichnet mit c^1) ist in der erzeugten Regelmenge ein Ungleichgewicht der Anzahl von Regeln mit *Same*= 0 und *Same*= 1 vorhanden, es ist zumeist $c^0 < c^1$. Sei also $c^0 < c^1$. Um eine Ausgewogenheit der Regeln zu erreichen, bestimmen wir einen Skalierungsfaktor für den minimalen Support der Regeln mit *Same*= 1 durch c^1/c^0 und entfernen dann alle Regeln mit *Same*= 1 als Folgerung aus der Regelmenge,[9] die den (reskalierten) minimalen Support

$$\text{min\_supp}^* = \frac{c^1}{c^0} \cdot \text{min\_supp}$$

nicht einhalten. Für die gewählte Parametrisierung von min_supp = 0.5 für die Regelerzeugung erhalten wir somit $\text{min\_supp}^* = \frac{c^1}{c^0} \cdot 0.5$. Eine weitere Verkleinerung der Regelmenge erhalten wir durch das Entfernen redundanter Regeln, d.h. von Regeln, für die es ein Redukt in der Regelmenge gibt. Eine Regel `head1 <- body1` ist Redukt einer Regel `head2 <- body2`, wenn gilt (vgl. Definition 3.6 auf Seite 86):

[9] Falls $c^1 = c^0$ gilt, ist keine Reskalierung notwendig; für $c^0 > c^1$ ist entsprechend die Menge der Regeln mit *Same*= 0 als Folgerung zu reduzieren.

(1) `head2` = `head1`,

(2) `body2` $\supset$ `body1` (d.h. jedes Item aus `body1` ist ebenfalls in `body2` enthalten) sowie

(3) conf(`head2 <- body2`) $\leq$ conf(`head1 <- body1`).

Im Folgenden geben wir eine kurze Beschreibung der Durchführung des Tests von AssoClass (die implizit das Ableiten einer Klassifikation aus den gelernten Assoziationsregeln enthält).[10]

Zunächst wird die Teststichprobe T_i um ein Attribut *Decision* ergänzt. Solange Datensätze mit Vergleichswertvektoren $r = f(a, b)$ von Datensätzen $a, b \in A$ in der Teststichprobe T_i existieren, für die das Attribut *Decision* keinen Wert hat, wird durchgeführt:

1. Ermittlung der transitiv abhängigen Attribut-Wert-Kombinationen des Vergleichswertvektors r (d.h. Attribut-Werte, die eine Folgerung in erzeugten passenden Assoziationsregeln sind, bei denen das Attribut *Same* nicht enthalten ist) und Erzeugung der abhängigkeitsfreien Regelmenge passender Regeln für r mit den verbleibenden Attributen,

2. Die passenden Regeln werden je Vergleichswertvektor r absteigend nach ihrer Confidence aufgezählt und es wird gemäß der gewählten Aggregationsmethode ein Wert $\delta(r)$ für die Klassifikation aus dem Intervall $[0, 1]$ bestimmt.

3. Es wird ein Update für das Attribut *Decision* mit dem berechneten Wert $\delta(r)$ für alle Datensätze mit dem Vergleichswert r durchgeführt (Optional können alle Updates als temporäre Abfragen gespeichert werden, wodurch sich eine gelernte Klassifikation explizit machen läßt.).

Die Werte der Variablen *model*, *MinConfidence*, *TopN*, *Lowerbound* und *Upperbound* eines Klassifikationsmodells bestimmen die Parametrisierung von AssoClass wie folgt:

- *model*: Die verwandte Aggregationsmethode für die für einen Vergleichswertvektor passenden Regeln, vgl. Abschnitt 3.3.3.2.

- *MinConfidence*: Die Confidence, die die zu aggregierenden Regeln mindestens einzuhalten haben,

- *TopN*: Beschränkung der Anzahl der maximal zu verarbeitenden passenden Regeln je Vergleichswertvektor,[11] keine Beschränkung mit *TopN*={INFINITY}

[10] Bei dem hier aufgezeigten Algorithmus für AssoClass wird auf effiziente Datenstrukturen für den Zugriff auf die Assoziationsregeln nicht näher eingegangen (z.B. Präfixbäume vgl. C. Borgelt, [BK02]), wir beschreiben das methodische Vorgehen .

[11] Bei gleicher Confidence der letzten Regel in der Aufzählung mit weiteren Regeln der Regelmenge wird die Beschränkung jedoch überschritten: die Regeln mit dieser Confidence werden alle mit berücksichtigt.

- *Lowerbound, Upperbound*: Grenzen für die Klassifikation mit $\delta(r)$ ($0 < Lowerbound \leq Upperbound < 1$), Werte aus dem Intervall [*Lowerbound, Upperbound*] bleiben unbestimmt (*undetermined*).

6.2.5 Präsentation der Ergebnisse

Die Ergebnisse stellen wir für jede Stichprobengröße (z.B. 2 500, 5 000 und 10 000 Datensatz-Paare) separat wie folgt dar:

1. Ergebnisse jedes Klassifikationsmodells für alle 12 Stichproben: $\hat{\alpha}$– und $\hat{\beta}$–Fehler für alle Daten, sowie exemplarisch *precision* und *recall* für die Bibliotheksdaten,[12]

2. Ergebnisse ($\hat{\alpha}$– und $\hat{\beta}$–Fehler) bei Variation der Grenzen *Lowerbound, Upperbound* der Entscheidungsfunktion, exemplarisch für je ein Modell von AssoClass für die Wohnungs- und Adreß-Daten[13] für ausgewählte Klassifikationsmodelle.

Der Übersichtlichkeit halber werden in den ersten beiden genannten Diagrammen (anstatt aller Einzelergebnisse) angegeben:

- *Median bzw. 50%–Quantil*: Punkt innerhalb der Box,

- *25%– und 75%–Quantil*: untere und obere (horizontale) Kante der Box,

- *Minimum bzw. 0%–Quantil*: untere horizontale Markierung unterhalb der Box,

- *Maximum bzw. 100%–Quantil*: obere horizontale Markierung oberhalb der Box.

Die Höhe der Box spiegelt die Stabilität der Werte wider, während die Differenz zwischen Maximum und Minimum die Größe der Streuung zeigt.

Die Diagramme sind bei den Ergebnissen der einzelnen Tests zu finden, für die Wohnungsdaten beispielsweise ab Seite 167. **DT** bezeichnet die Klassifikationsmodelle des Entscheidungsbaumverfahrens, **RL** diejenigen des Record Linkage und schließlich **AC** die Modelle für AssoClass. Die fortlaufenden Nummern auf der Abszisse bezeichnen eine Modell-ID, aus der sich die Parametrisierung der Verfahren ersehen läßt.

[12] Wir verzichten auf die Abbildung von *precision* und *recall* für alle Datenbestände, da kein grundsätzlicher Unterschied zwischen dem Verhalten von $(1-\hat{\alpha})$ und *recall* sowie von $(1-\hat{\beta})$ und *precision* besteht, lediglich durch das Verhältnis der dubletten zu den nichtdubletten Paaren in der Stichprobe erhält die *precision* eine Reskalierung, im Gegensatz ist der β–Fehler unabhängig von diesem Verhältnis.

[13] Da lediglich AssoClass die Möglichkeit einer anwendbaren Skalierbarkeit der Fehlerraten bietet, vgl. die Diskussion bei der Auswertung der Tests, Abschnitt 6.6.1.2 auf Seite 196.

Tabelle 6.1: Wohnungsdaten: Attribute und Vergleichsfunktionen

Nr.	Attributname	Vergleichsfunktion	#Werte	Skala
1	Rent	fctdistanceOrdinal	6	ordinal
2	District	fctYesNo	2	nominal
3	Rooms	fctdistanceOrdinal	3	ordinal
4	Street	fctYesNoNull	3	nominal
5	Size	fctdistanceOrdinal	3	ordinal
6	Phone	fctDiscreteMinEditDistance	4	ordinal
7	Datasource	fctYesNo	2	nominal
8	Rent_Brutto	fctYesNo	2	nominal
9	AddCosts	fctYesNo	2	nominal
10	FreeDate	fctYesNoNull	3	nominal
11	ExtraCommission	fctYesNoNull	3	nominal
12	Floor	fctYesNoNull	3	nominal
13	Fulltext	fctPercentageSameWordsDiscrete	4	ordinal

6.3 Evaluation mit Wohnungsdaten

Zur Beurteilung der Qualität von drei ausgewählten Methoden (Decision Tree, Record Linkage und AssoClass) verwenden wir eine Tabelle mit 9 842 Berliner Wohnungsanzeigen, die knapp 2 200 Duplikate enthält. Für die Identifizierung der annoncierten Wohnungen ziehen wir bis zu 13 Attribute heran, für die jeweils eine diskrete Vergleichsfunktion festgelegt wurde (siehe Tabelle 6.1). Als einen die Vorauswahl von Datensatzpaaren bestimmenden Selektor σ haben wir die Differenzierungs-Schlüssel `District`, `Size` ($\Delta = 1$), `Rooms` ($\Delta = 0.5$) sowie die semantischen Schlüssel für die vier Teilmengen $A_1, \ldots, A_4$ (d.h. `Date` oder `Datasource` ist verschieden) verwandt. Der Selektor ist darstellbar mit dem SQL-Statement

```
SELECT A.*, B.*
  FROM A AS A, A AS B
  WHERE A.District = B.District
    AND A.Size >= B.Size - 1 AND A.Size <= B.Size + 1
    AND A.Rooms >= B.Rooms - 0.5 AND A.Rooms <= B.Rooms + 0.5
    AND (A.Date <> B.Date OR A.Datasource <> B.Datasource)
    AND A.ID < B.ID;
```

Für den Test betrachten wir die in der Tabelle 6.1 angegebenen 13 Attribute sowie ihnen zugeordnete diskrete Vergleichsfunktionen (nominale/ordinale Skalen). Jede Vergleichsfunktion hat bis zu 6 mögliche Vergleichswerte, insgesamt gibt es mehr als 10^6 unterschiedliche Kombinationen (Vergleichsvektoren),[14] wovon allerdings ein Großteil bei keinem Vergleich von Datensätzen auftritt.

[14] Es sind $6 \cdot 2 \cdot 3 \cdot 3 \cdot 3 \cdot 4 \cdot 2 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot 3 \cdot 4 = 1.119.744$ Kombinationen

Tabelle 6.2: Wohnungsdaten: Charakteristika der Daten

Attribut	accuracy	confidence	anti-confidence	nulls	selectivity
Datasource	0.8453	0.6348	0.9845	0.0000	0.0002
District	1.0000	0.5000	1.0000	0.0000	0.0026
ExtraCommision	0.7009	0.6689	0.9976	0.3546	0.0002
Floor	0.4908	0.8833	0.9829	0.5651	0.0013
FreeDate	0.9248	0.6775	0.9931	0.0002	0.0040
Fulltext	0.4120	0.9896	0.9701	0.0000	0.8614
NK	0.8780	0.7435	0.9960	0.0875	0.0305
Phone	0.9575	0.9580	0.9986	0.0389	0.2706
Rent	0.9402	0.9719	0.9982	0.0302	0.1301
Rent_Brutto	0.8198	0.6954	0.9949	0.1799	0.0024
Rooms	0.9957	0.5335	0.9986	0.0011	0.0016
Size	0.9721	0.7137	0.9999	0.0311	0.0252
Street	0.8721	0.9653	0.9999	0.3324	0.1580

Attribut	delta-accuracy	delta-confidence
Fulltext	0.6671	0.9864
Phone	0.9654	0.9524
Rent	0.9528	0.9052
Rooms	0.9992	0.4933
Size	0.9736	0.4933

Tabelle 6.3: Die Klassifikationsmodelle DT1–DT6 für das Entscheidungsbaumverfahren (für alle drei Datenbestände)
(Die Skalen der Vergleichs-Attribute bzw. ihre Wertebereiche (*engl.: domains*) wurden entsprechend Tabelle 6.1 gesetzt (z.B. für die Wohnungsdaten: ordinal ($\mathbb{Z}$) für die Vergleichs-Attribute `Rent`, `Size`, `Phone` sowie `FullText` und nominal für die verbleibenden Attribute)

Modell	measure	pruning	attributes
DT1	infgain	no	all
DT2	infgain	yes	all
DT3	infgr	no	all
DT4	infgr	yes	all
DT5	gini	no	all
DT6	gini	yes	all

Tabelle 6.4: Wohnungsdaten: Die Klassifikationsmodelle RL1–RL10 für Record Linkage

model	interaction terms	attributes
RL1	{1-way}	{3 4 5 6 12},
RL2	{2-way}	{3 4 5 6 12}
RL3	{3-way}	{3 4 5 6 12}
RL4	{1-way}	{1 3 5 6 12}
RL5	{1-way}	{3 4 5 6 12 13}
RL6	{2-way}	{3 4 5 6 12 13}
RL7	{3-way}	{3 4 5 6 12 13}
RL8	{1-way}	{1 3 4 5 6 12 13}
RL9	{1-way}	{1 3 4 5 6 7 12 13}
RL10	{1-way}	{1 3 4 5 6 7 10 11 12 13}

Tabelle 6.5: Wohnungsdaten: Die Klassifikationsmodelle AC1–AC9 für AssoClass (bei allen Modellen war `UpperBound={0.5+ϵ}`, `LowerBound={0.5+ϵ}`, `minConfidence={90},topN={100}`, $\epsilon = 10^{-12}$ gesetzt)

model	rule aggregation method	attributes
AC1	NonConflicts	{3 4 5 6 12}
AC2	Confidence	{3 4 5 6 12}
AC3	Rank	{3 4 5 6 12}
AC4	NonConflicts	{3 5 12 13}
AC5	Confidence	{3 5 12 13}
AC6	Rank	{3 5 12 13}
AC7	NonConflicts	{1 3 4 5 6 12 13}
AC8	Confidence	{1 3 4 5 6 12 13}
AC9	Rank	{1 3 4 5 6 12 13}

6.3.1 Die Ergebnisse des Tests

Wir verwandten für den Test folgende 36 Datensatz-Lernstichproben: `L2500a`,..., `L2500l`, `L5000a`,..., `L5000l`, `L10000a`,..., `L10000l`, die jeweils die Hälfte der Datensätze der Stichproben `P2500a`,..., `P10000l`, d.h. 1 250, 2 500 bzw. 5 000 Datensatzpaare enthielten. Die Zahl der in den Stichproben enthaltenen Duplikate betrug jeweils ca. 50%, 25% und 20% des Stichprobenumfangs, d.h. für die Lernstichproben `L2500a`,..., `L2500l`, `L5000a`,..., `L5000l` jeweils ca. 650 dublette Paare sowie ca. 1 000 dublette Paare für `L10000a`,..., `L10000l`.

Es sei bemerkt, daß wir in diesem Test keine unbestimmten Paare zugelassen haben. Desweiteren liefern alle Verfahren eine Bewertung (*engl.: score*) mit einem Wert aus dem Intervall [0,1] die Grenzen *LowerBound* und *UpperBound* (die in diesem Test gleichgesetzt wurden, s.u.). Alle Paare (a, b) mit $\delta(a, b) < \mathit{LowerBound}$ werden als Duplikat klassifiziert, alle Paare mit $\delta(a, b) > \mathit{UpperBound}$ als Nicht-Duplikat. Durch Verschiebung der Grenze *UpperBound* nach oben wird der α-Fehler verringert, während sich durch Verschiebung der Grenze *LowerBound* nach unten der β-Fehler verkleinert. Falls eine bestimmte Zahl von Datensatz-Paaren unbestimmt bleiben darf, kann ein Unsicherheitsintervall [*LowerBound*,*UpperBound*] spezifiziert werden, in dem keine Entscheidung gefällt werden soll und damit erhalten wir deutlich niedrigere als die dargestellten Fehlerraten für alle Klassifikationsmodelle.

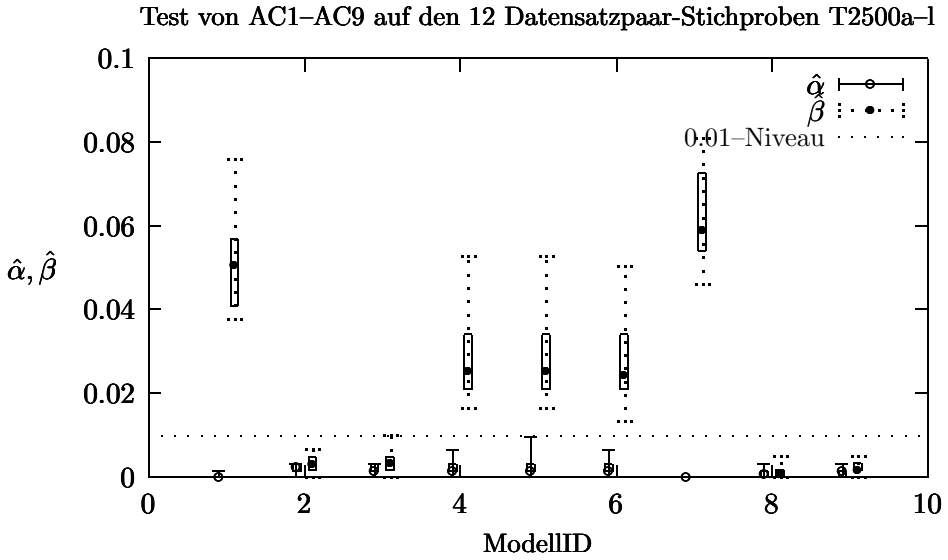
- *Record Linkage* liefert bei Beschränkung auf eine kleine Zahl von Attributen gute Ergebnisse, insbesondere die Modelle RL1–RL3 mit den fünf Attributen *Rooms, Street, Size, Phone* und *Floor* zeigen für alle Stichprobengrößen überdurchschnittlich niedrige Raten des α-Fehlers, wobei der β-Fehler etwas höher ausfällt als bei den anderen Attributen. Das Klassifikationsmodell RL4 hat eine andere Attributauswahl. Statt *Street* wurde *Rent* verwandt, was jedoch zu höheren Fehlerraten führt, da diese Attributkombination für die Identifizierung ungeeignet ist. Die Modelle RL5–RL10 mit sechs bis zehn Einflußgrößen haben deutlich höhere und zudem instabile α-Fehlerraten. Es sei bemerkt, daß die Fehlerraten von der Zahl verwandter Einflußgrößen und kaum vom spezifizierten log-linearen Modell abhängen (d.h. der Berücksichtigung von 1-, 2- oder 3-Faktor-Interaktionen).

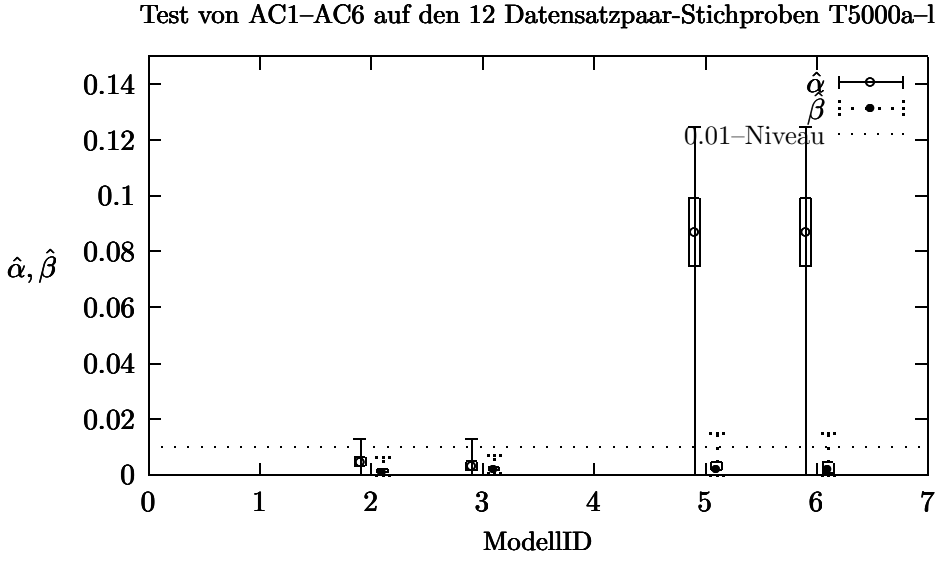
- das *Entscheidungsbaum-Verfahren* liefert bei den Modellen DT1–DT6 sehr gute Fehlerraten. Interessanterweise zeigt das Modell DT3 (Information Gain Ratio ohne Pruning) bessere Ergebnisse als die anderen 5 Modelle, die alle ähnlich niedrige Fehlerraten haben.

- Für die Ergebnisse von *AssoClass* läßt sich feststellen, daß die Aggregationskritererien *Confidence* und *Rank* zu denselben Fehlerraten führen, und das auch schon für die kleinen Stichproben `T2500a-1` (in denen allerdings 50% dublette Paare enthalten waren). Die Modelle AC4–AC6 schneiden in diesem Test nicht gut ab, was mit der gewählten Attributmenge *Rooms, Size, Floor* und *FullText* zu erklären ist, die für eine hinreichende Identifizierung ungeeignet

ist. Akzeptable Werte liefern sowohl die Modelle AC2 und AC3 mit den Attributen *Rooms, Street, Size, Phone* und *Floor* (bessere Werte als die Modelle RL2 und RL3 mit denselben Attributen) als auch die Modelle AC8 und AC9, bei denen zusätzlich die Attribute *Rent* und *FullText* verwandt werden. Allerdings ist durch die Hinzunahme dieser beiden Attribute keine Veränderung der Fehlerrate zu ersehen. Zu den Modellen AC1 und AC7 ist zu bemerken, daß der α-Fehler minimiert werden kann (bis auf Null!) durch die Verwendung der Aggregationsmethode *NonConflicts*, bei dem ein Paar nur dann als Nicht-Duplikat klassifiziert wurde, wenn *keine* widersprüchlichen Regeln für das Paar existierten. Allerdings muß man dabei einen erhöhten β-Fehler in Kauf nehmen. Durch Variation der Grenzen für die Klassifikation läßt sich die in Abbildung 6.6 gezeigte Skalierbarkeit der Fehlerraten erreichen.

Zusammengefaßt läßt sich sagen, daß alle drei Methoden für die Wohnungsdaten zufriedenstellende Ergebnisse der Korrektheit mit Fehlerraten um 1% zeigen.





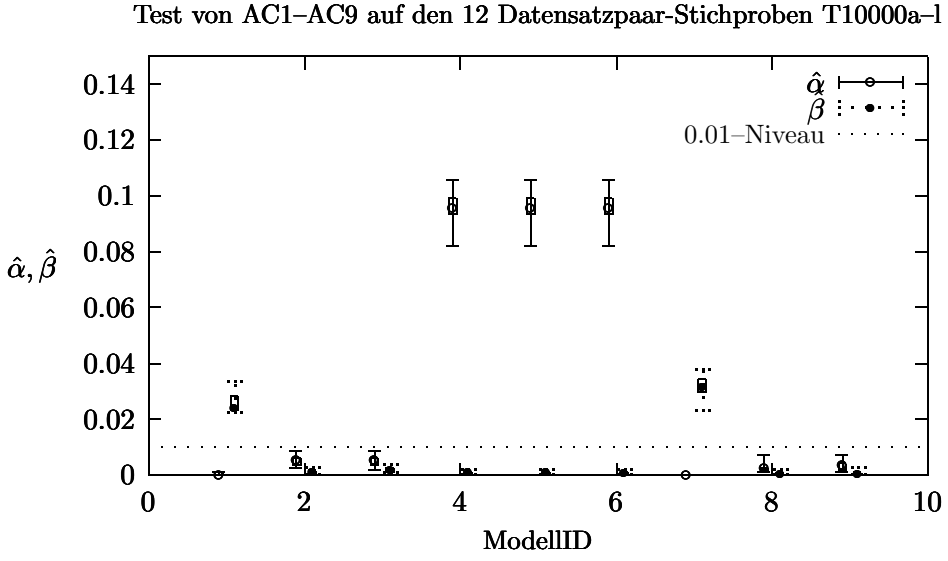


Abbildung 6.3: Die Ergebnisse von AssoClass für die Wohnungsdaten

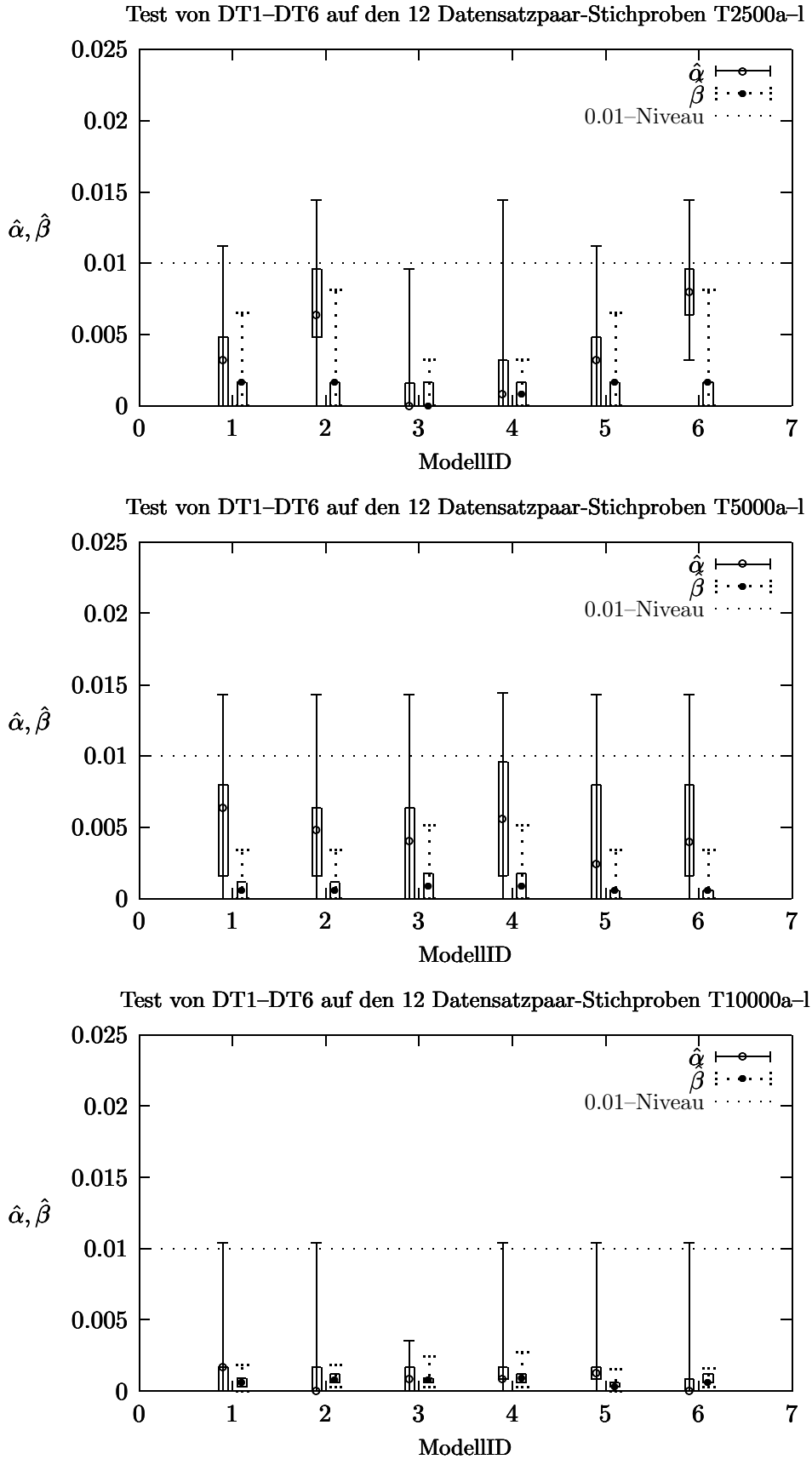


Abbildung 6.4: Die Ergebnisse des Entscheidungsbaumverfahrens für die Wohnungsdaten

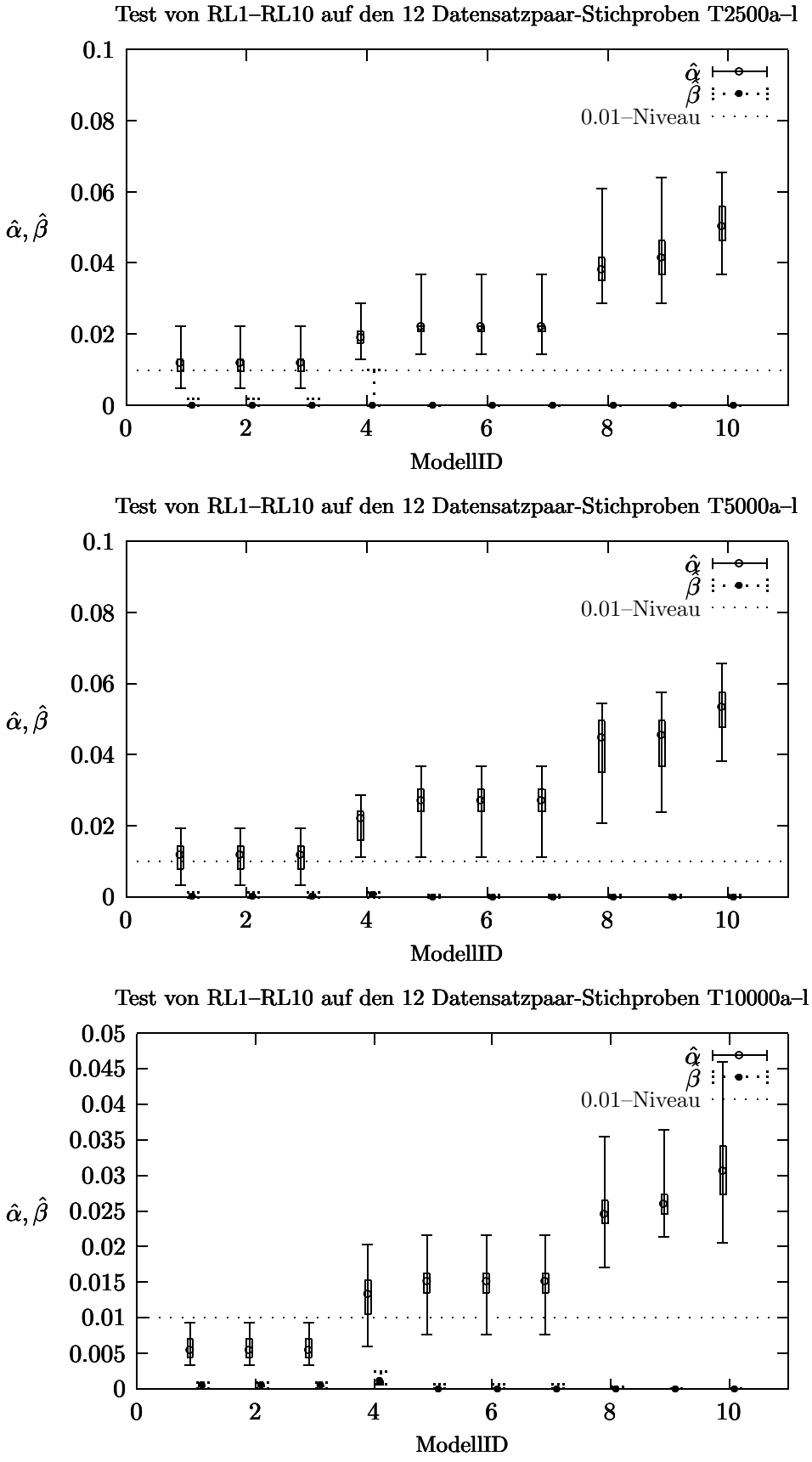


Abbildung 6.5: Die Ergebnisse von Record Linkage für die Wohnungsdaten

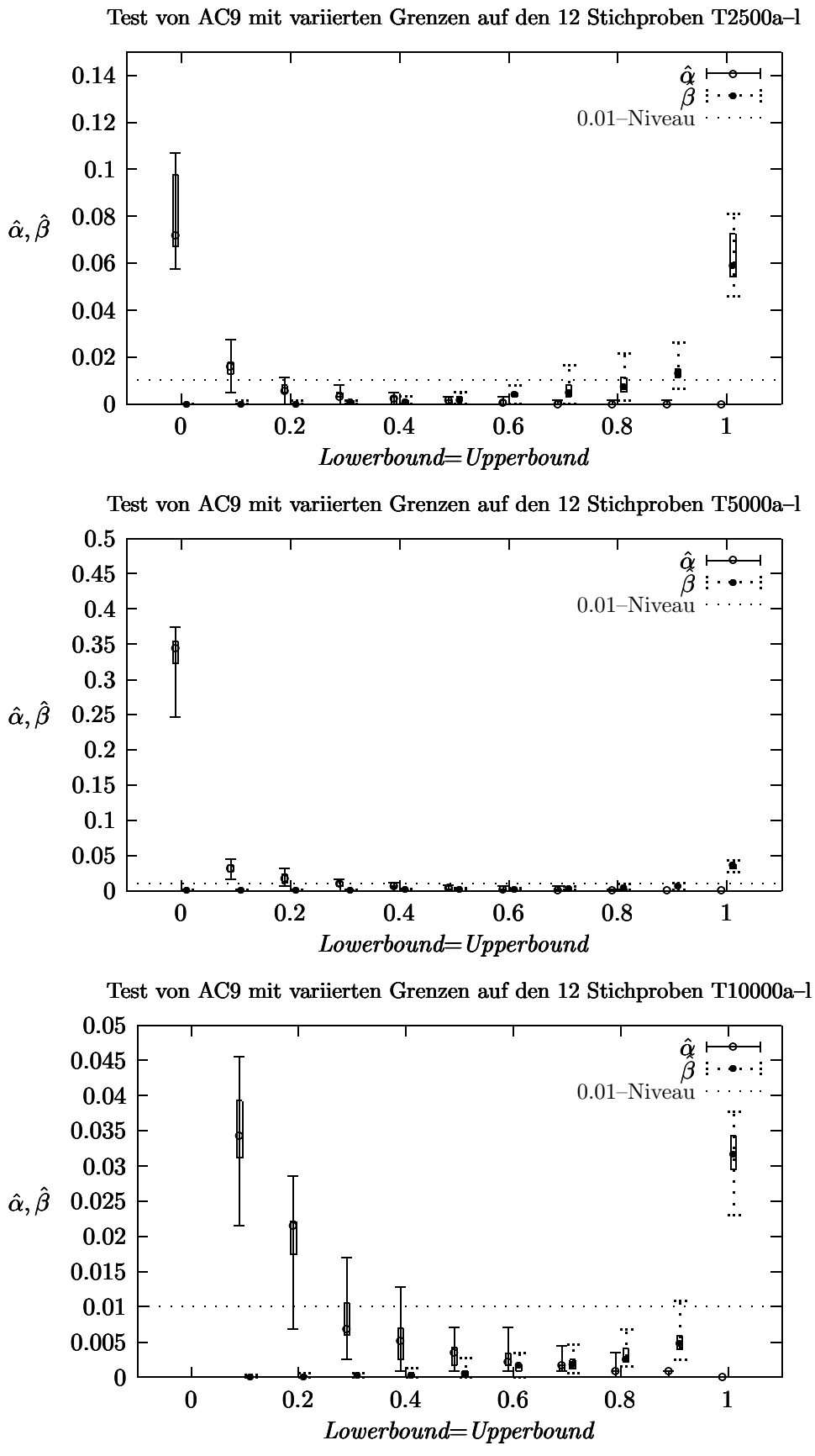


Abbildung 6.6: Die Ergebnisse von AC9 bei Variation der Grenzen der Entscheidungsfunktion (ohne offene Entscheidungen, d.h. mit *undetermined*= 0 und *Lowerbound*=*Upperbound*) für die Wohnungsdaten

6.4 Evaluation mit Adreßdaten

In diesem Abschnitt geben wir die Ergebnisse der Testläufe mit Adreßdaten wider. Zur Evaluation stand uns ein Datenbestand mit über 250 000 Kunden-Datensätzen zur Verfügung. Die Daten wurden von der Deutschen Post mit dem Verzeichnis deutscher Postanschriften abgeglichen (dieses Verzeichnis wird ständig aktualisiert, z.B. durch die Berücksichtigung erteilter Nachsendeaufträge und durch die regelmäßige Überprüfung der Zustelladressen durch Briefträger); dieser Service der Dublettenbereinigung wird derzeit mit dem Namen *Double Clean* von der Deutschen Post, Direkt Marketing, angeboten.[15] Der Abgleich förderte über 52 000 Duplikate in dem Datenbestand zu Tage, die wir in einer optimierten Same-Relation abgelegt haben.[16] Wir verwenden diese Same-Relation als Referenz, da wir davon ausgehen, daß dieses Verfahren aufgrund der ständigen Aktualisierung eine große Genauigkeit hat, und deshalb zu einer verschwindend geringen Fehlklassifikation führt.

Tabelle 6.6: Die Charakteristika für die Adreßdaten (Teil 1)

Attribut	accuracy	confidence	anti-confidence	nulls	selectivity
Sex	0.98622	0.98902	0.56352	0	0.00001
FirstName	0.92178	0.97704	0.70849	0.00032	0.02221
LastName	0.92748	0.98571	0.95013	0	0.11925
FullName	0.84248	0.97090	0.99904	0	0.71563
BirthDate	0.73260	0.97679	0.99896	0.07025	0.11029
Day	0.75363	0.98063	0.96299	0.07025	0.00012
Month	0.76123	0.98136	0.91368	0.07025	0.00005
Year	0.75266	0.98056	0.97093	0.07025	0.00044
Zip-Code	0.98726	0.99750	0.96655	0	0.00033
Town	0.95495	0.99103	0.94634	0	0.00432
Street	0.86535	0.97498	0.99836	0.00059	0.20684
HouseNumber	0.90678	0.98220	0.97646	0.01520	0.00805

Unter Berücksichtigung mehrfacher Duplikate lassen sich über 58 000 verschiedene dublette Paare bilden (dies entspricht der Kardinalität von $TC(Same)$). Die Datensätze enthalten neben den Namen und Adressen der Kunden auch das Geburtsdatum (für 93% der Datensätze) sowie eine Anrede (aus der wir das Geschlecht ableiten) und einige Zusatzattribute (Titel, Namenszusatz, Ortszusatz). Es sei bemerkt, daß die Daten zwar zu einem Stichtag zusammengetragen wurden, aber die Aktualität vom letzten Kundenkontakt abhängt (der aber nicht aus diesen Daten ersichtlich ist), so daß die Angaben veraltet sein können. Viele Variationen in den Daten sind durch

[15] Service-Angebote zur Datenbereinigung der Deutschen Post unter `www.postdirekt.de` oder `www.addressfactory.de`.

[16] Es sei bemerkt, daß bei diesem Abgleich sogenannte *Dublettengruppen* gebildet werden, die als zusätzlicher Attributwert jedem Datensatz hinzugefügt werden; `Dublettengruppe`$= 0$ bedeutet, daß kein Duplikat gefunden wurde, alle Datensätze mit demselben Wert `Dublettengruppe`$= k$, $k \in \mathbb{N}, k > 0$ wurden als Duplikat klassifiziert.

unterschiedliche Schreibweisen und Abkürzungen bestimmt, beispielsweise für Straßen und Ortsteil-Bezeichnungen. Da sich die Adresse durch Umzug des Kunden geändert haben kann, sind die Adreßdaten (PLZ, Ort, Straße, Hausnummer) als Differenzierungsschlüssel ungeeignet. Falls das Geburtsdatum fehlt oder nicht korrekt ist, führt es als Differenzierungsschlüssel ebenfalls zu einer hohen Fehlerrate (d.h. nicht gebildete dublette Paare). Wir beschränken deshalb die Vorauswahl, d.h. den Selektor σ, auf die Namen eines Kunden. Als Vorauswahlkriterium verlangen wir, daß *mindestens ein phonetischer Code* des zusammengesetzten Namens (einschließlich Namenszusatz) übereinstimmt. Wir verwenden dafür ein Phonetikmodul, das für deutschsprachige Namen entwickelt wurde, die sogenannte *Kölner Phonetik* (siehe z.B. [Pos69]).[17] Die phonetischen Codes der Namensbestandteile werden in einer (Index-)Tabelle `Idx_PhoneticCode(ID, Code)` gespeichert, so daß der effiziente Zugriff auf die Datensätze, bei denen mindestens ein Code übereinstimmt, entsprechend Abschnitt 5.2.1 auf Seite 135 realisiert werden kann.

Für die Evaluation verwenden wir Lern- und Teststichproben, die jeweils ein Drittel dublette und zwei Drittel nicht-dublette Datensatz-Paare enthalten. Dazu bildeten wir Paar-Stichproben unterschiedlicher Größe zufällig gezogener Datensätze (innerhalb der Vorauswahl) und ergänzten diese um zufällig aus $TC(Same)$ gezogene dublette Paare. Wir zogen drei mal zwölf Stichproben mit jeweils 5 000, 10 000 und 20 000 Datensatz-Paaren, die wir entsprechend mit `P5000a`–`l`, `P10000a`–`l` und `P20000a`–`l` bezeichneten. Die daraus gebildeten Lern- und Test-Stichproben mit jeweils der Hälfte der (dubletten und nicht-dubletten) Paare, d.h. mit 2 500, 5 000 sowie 10 000 Datensatz-Paaren, erhielten den Präfix `L` bzw. `T` statt `P`, beispielsweise wurden die Datensätze der Stichprobe `P10000h` aufgeteilt auf die Lernstichprobe `L10000h` und die Teststichprobe `T10000h`.

Die Parametrisierung der Verfahren ist analog zu den Wohnungsdaten,

[17] Im Gegensatz zu vielen phonetischen Codes, wie dem *Soundex-Code* (vgl. z.B. [New88]) unterscheidet die *Kölner Phonetik* Primär- und Sekundärformen von Namen. Durch Erzeugung alternativer Codes eines Namens lassen sich Variationen in der Schreibweise, insbesondere für zusammengesetzte Namen, gut erfassen.

Tabelle 6.7: Die Charakteristika für die Adreßdaten (Teil 2)

Attribut	delta-confidence	delta-accuracy
FirstName	0.67784	0.97181
LastName	0.90442	0.99127
FullName	0.99912	0.01848
Street	0.97925	0.95794
Year	0.92168	0.76129

Als Abstandsmaße für die delta-confidence und delta-accuracy verwandten wir die *MinimumEditDistance* für *FirstName*, *LastName* (mit $\Delta = 2$) und *Street* (mit $\Delta = 1$), die Funktion ($1-$*fctPercentageOfMatchingBiGrams*) für *FullName* (mit $\Delta = 0.2$) und den Absolutbetrag für *Year* (mit $\Delta = 1$).

wobei wir die in Tabelle 6.8 angegebenen Vergleichsattribute und Vergleichsfunktionen verwandten. Für die Namensattribute setzten wir mehrere Vergleiche um. Insgesamt erhielten wir bei einem Vergleich zweier Kunden-Datensätze 17 Vergleichswerte. Die Klassifikationsmodelle DT1–DT6 für das Entscheidungsbaumverfahren sind die für die Wohnungsdaten verwandten Modelle (vgl. Tabelle 6.3 auf Seite 163), d.h. wir benutzten alle 17 verfügbaren Vergleichsattribute, wobei wir bei den ordinal skalierten Vergleichsattributen wie `LNameEdit` den Wertebereich entsprechend modifizierten. Für die Klassifikation mit Assoziationsregeln und für Record Linkage haben wir eine Auswahl der Attribute vorgenommen, vgl. die Tabellen 6.9 sowie 6.10.

Tabelle 6.8: Adreßdaten: Verwandte Attribute und Vergleichsfunktionen

Nr.	Attributname	Vergleichsfunktion	#Werte	Skala
1	FnameEdit	fctDiscreteMinEditDistance	7	ordinal
2	LnameEdit	fctDiscreteMinEditDistance	7	ordinal
3	ZIP	fctYesNo	2	nominal
4	LNameBigrams	fctPercentageOfMatchingBiGrams	4	ordinal
5	FNameBigrams	1−fctPercentageOfMatchingBiGrams	4	ordinal
6	LNameKoeln	fctHasEqualPhoneticCode	3	nominal
7	FNameKoeln	fctHasEqualPhoneticCode	3	nominal
8	Town	fctYesNoNull	3	nominal
9	Street	fctDiscreteMinEditDistance	7	ordinal
10	HouseNumber	fctHasNonemptyIntersection	3	nominal
11	Year	fctYesNoNull	3	nominal
12	Month	fctYesNoNull	3	nominal
13	Day	fctYesNoNull	3	nominal
14	BirthDate	fctYesNoNull	3	nominal
15	Sex	fctYesNo	2	nominal
16	FullNameBigrams	1−fctPercentageOfMatchingBiGrams	5	ordinal
17	FullNameEdit	fctDiscreteMinEditDistance	7	ordinal

Tabelle 6.9: Adreßdaten: Die Klassifikationsmodelle AC1–AC9 für die Klassifikation mit Assoziationsregeln (AssoClass)

model	rule aggregation method	attributes
AC1	NonConflicts	{1 2 3 9}
AC2	Confidence	{1 2 3 9}
AC3	Rank	{1 2 3 9}
AC4	NonConflicts	{1 2 3 9 10 14}
AC5	Confidence	{1 2 3 9 10 14}
AC6	Rank	{1 2 3 9 10 14}
AC7	NonConflicts	{1 2 3 8 9 10 14 15 16}
AC8	Confidence	{1 2 3 8 9 10 14 15 16}
AC9	Rank	{1 2 3 8 9 10 14 15 16}

Tabelle 6.10: Adreßdaten: Die Klassifikationsmodelle RL1–RL10 für Record Linkage

model	interaction terms	attributes
RL1	{1-way}	{1 2 3 9}
RL2	{2-way}	{1 2 3 9}
RL3	{3-way}	{1 2 3 9}
RL4	{1-way}	{3 4 5 9}
RL5	{2-way}	{3 4 5 9}
RL6	{3-way}	{3 4 5 9}
RL7	{1-way}	{1 2 3 10 14 15}
RL8	{1-way}	{3 4 5 10 14 15}
RL9	{1-way}	{1 2 3 8 9 10 14 15}
RL10	{1-way}	{3 4 5 8 9 10 14 15}

6.4.1 Ergebnisse

6.4.1.1 Messung der Korrektheit

Für die Messung der Korrektheit nahmen wir eine Bestimmung der Fehlerraten für die betrachteten Klassifikationsmodelle auf den 3 mal 12 Stichproben vor. Die Ergebnisse sind in Abbildungen auf den folgenden Seiten zu finden.

In den Abbildungen 6.7 und 6.8 zeigen wir die geschätzten α- und β-Fehler für die Klassifikation mit Assoziationsregeln, AssoClass. Die Abbildung 6.7 zeigt die Ergebnisse der Modelle mit den Aggregationsmethoden *Confidence* und *Rank*. Aus den Resultaten läßt sich ersehen, daß die Attributauswahl einen großen Einfluß ausübt. Die Modelle AC5 und AC6, bei denen die Attribute *FNameEdit*, *LNameEdit*, *Zip*, *Street*, *HouseNumber* und *BirthDate* verwandt wurden, zeigen die geringsten Fehlerraten. Durch die Hinzunahme der Vergleichsattribute *Sex*, *Town* und *FullNameBigram* bei AC8 und AC9 verschlechterten sich jedoch die Fehlerraten. Dieses läßt sich dadurch erklären, daß durch die Regeln, in denen die zusätzlichen Attribute vorkommen, die mit der Aggregationsmethode aus den Regeln gewonnene Klassifikation zu stark an die Lernstichprobe angepaßt wird, was dann höhere Fehlerraten auf der Teststichprobe zur Folge hat.

Die aus den in AC2 und AC3 verwandten Attributen *FNameEdit*, *LNameEdit*, *Zip* und *Street* gelernten Klassifikationen weisen allerdings noch höhere Fehlerraten auf, da diese Attribute nicht genügend Information zur Identifizierung enthalten.

In der Abbildung 6.8 sind die Ergebnisse der drei Modelle AC1, AC4 und AC7 gezeigt, bei denen konfliktfreie Regeln zu einer klaren Entscheidung 0 oder 1 führen, andernfalls wird 0.5 gesetzt. Hier waren die Grenzen mit $Lowerbound = Upperbound = 0.5 + \epsilon$ für $\epsilon = 10^{-12}$ gesetzt, so daß Paare mit widersprechenden passenden Regeln als Duplikat klassifiziert wurden. Dadurch erhalten wir einen minimalen α-Fehler, wobei jedoch der β-Fehler recht groß wird.

Auf der darauffolgenden Seite sind in der Abbildung 6.9 die Ergebnisse des Entscheidungsbaumverfahrens abgebildet. Alle sechs Modelle zeigen

sämtlichst gleich gute Ergebnisse, d.h. unabhängig vom gewählten Maß und unabhängig davon, ob Pruning zum Einsatz kam.

Für das Entscheidungsbaumverfahren und für Record Linkage (Abbildung 6.10 auf Seite 179) ersieht man auch, daß mit wachsender Stichprobengröße sich sowohl die Fehlerraten als auch deren Streuung verringern. Das liegt darin begründet, daß eine gelernte Klassifikation umso stabiler wird, umso größer die Lernstichprobe wird; insbesondere wird für Record Linkage ein log-lineares Modell umso besser geschätzt, desto mehr Daten vorliegen. Bei Record Linkage sieht man, das die Auswahl der Attribute großen Einfluß auf die Korrektheit hat: Die Modelle RL9 und RL10, die die meisten Attribute enthalten, haben den größten α-Fehler. Der Unterschied zwischen den Modellen RL1–RL3 und RL4–RL6 besteht in den Vergleichsfunktionen für den Namen, bei ersteren wurde die *MinimumEditDistance* mit bis zu 6 verschiedenen Werten und bei letzteren der Vergleich mit *Bigrams* mit 4 möglichen Werten vorgenommen — aus den Ergebnissen ist kein signifikanter Unterschied zu erkennen, lediglich die Streuung ist für die *Bigrams* geringer, was aus der geringeren Zahl an Fällen resultiert ($4 \cdot 4 = 16$ statt 36 Fälle).

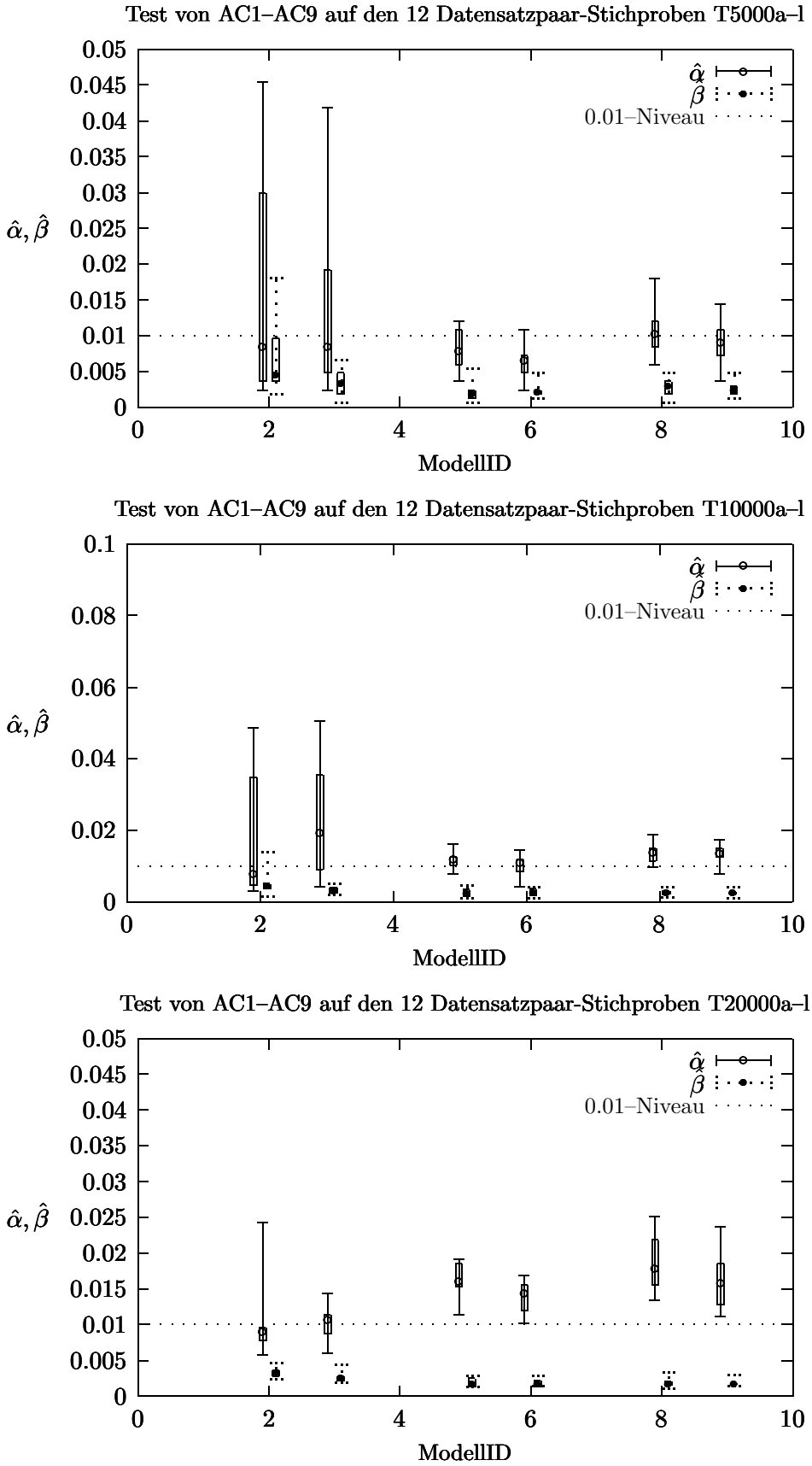
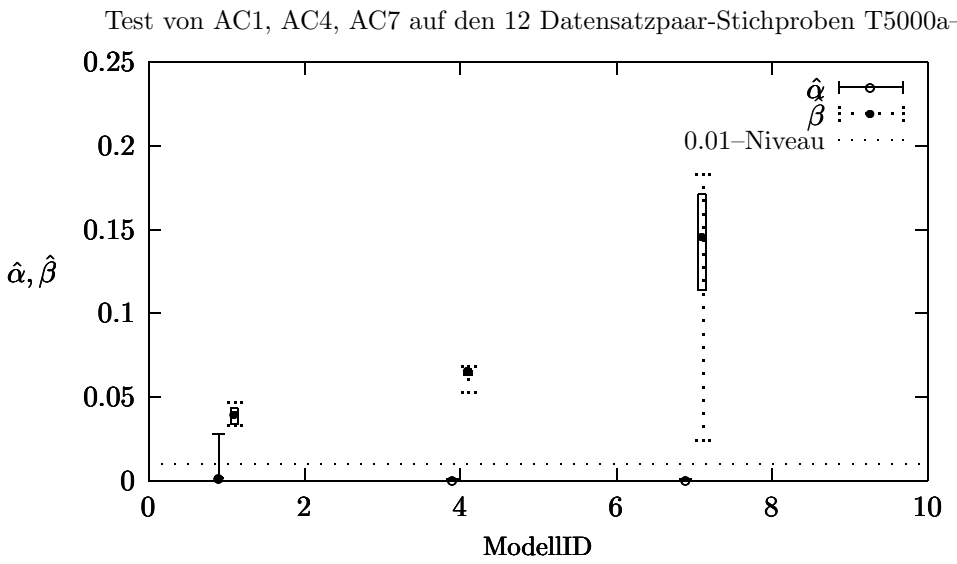
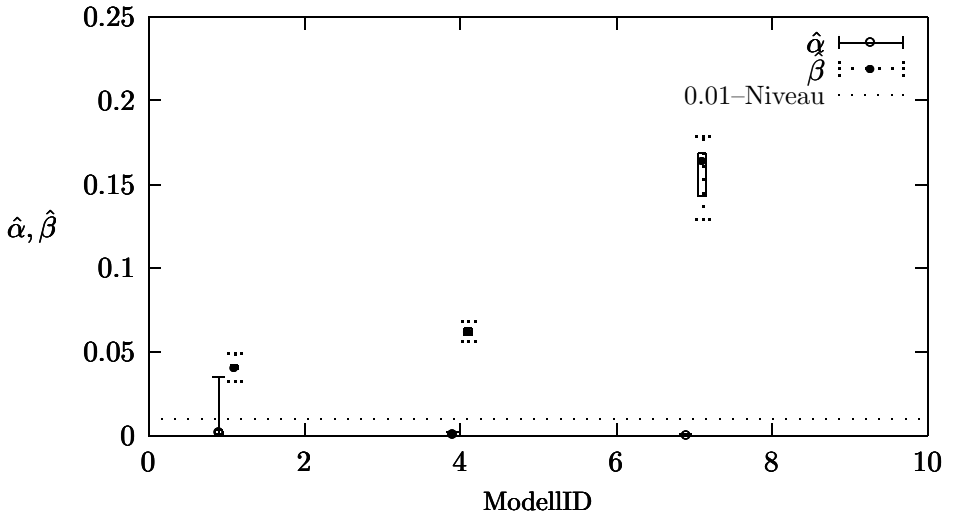


Abbildung 6.7: Die Ergebnisse von AssoClass für die Adreßdaten (ohne Modelle AC1, AC4 und AC7)





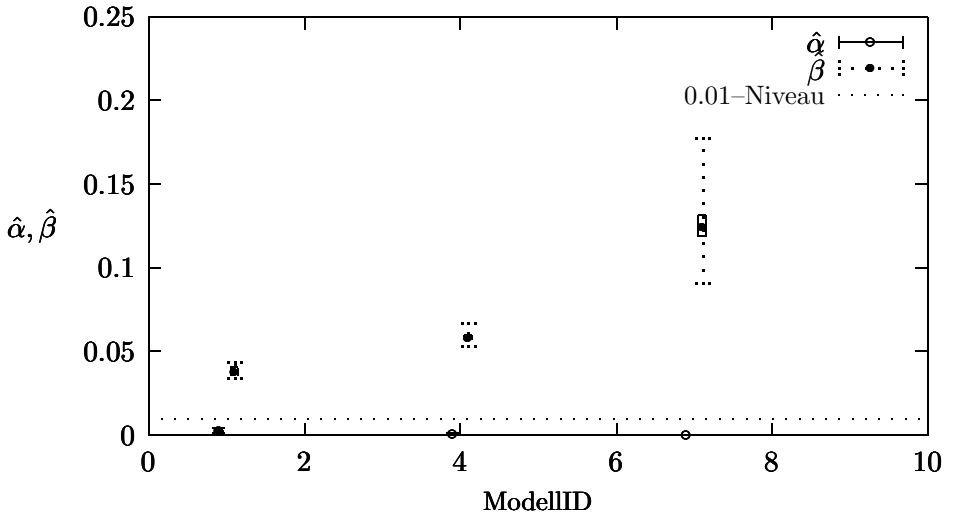


Abbildung 6.8: Die Ergebnisse von AssoClass für die Adreßdaten (nur Modelle AC1, AC4 und AC7)

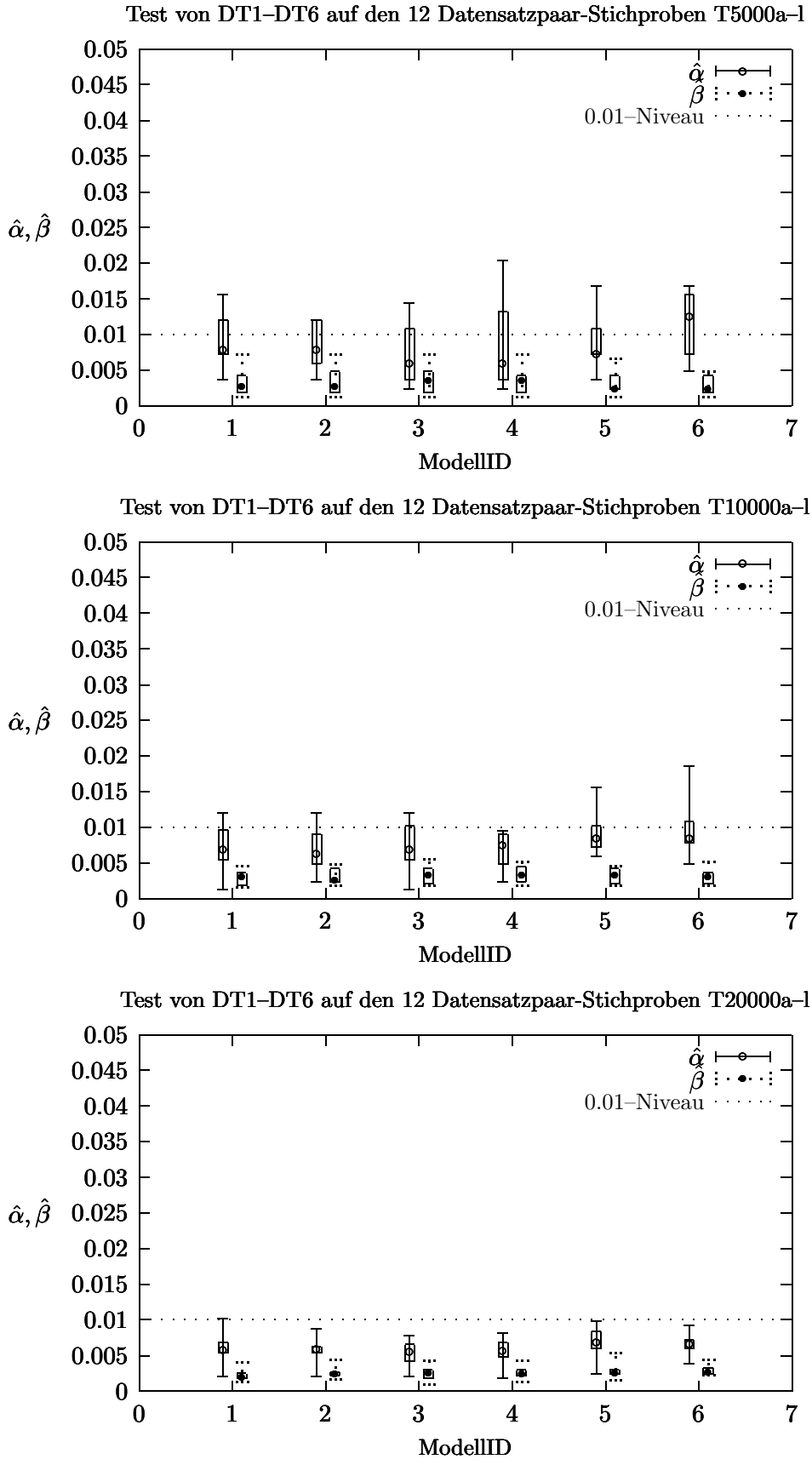


Abbildung 6.9: Die Ergebnisse des Entscheidungsbaumverfahrens für die Adreßdaten

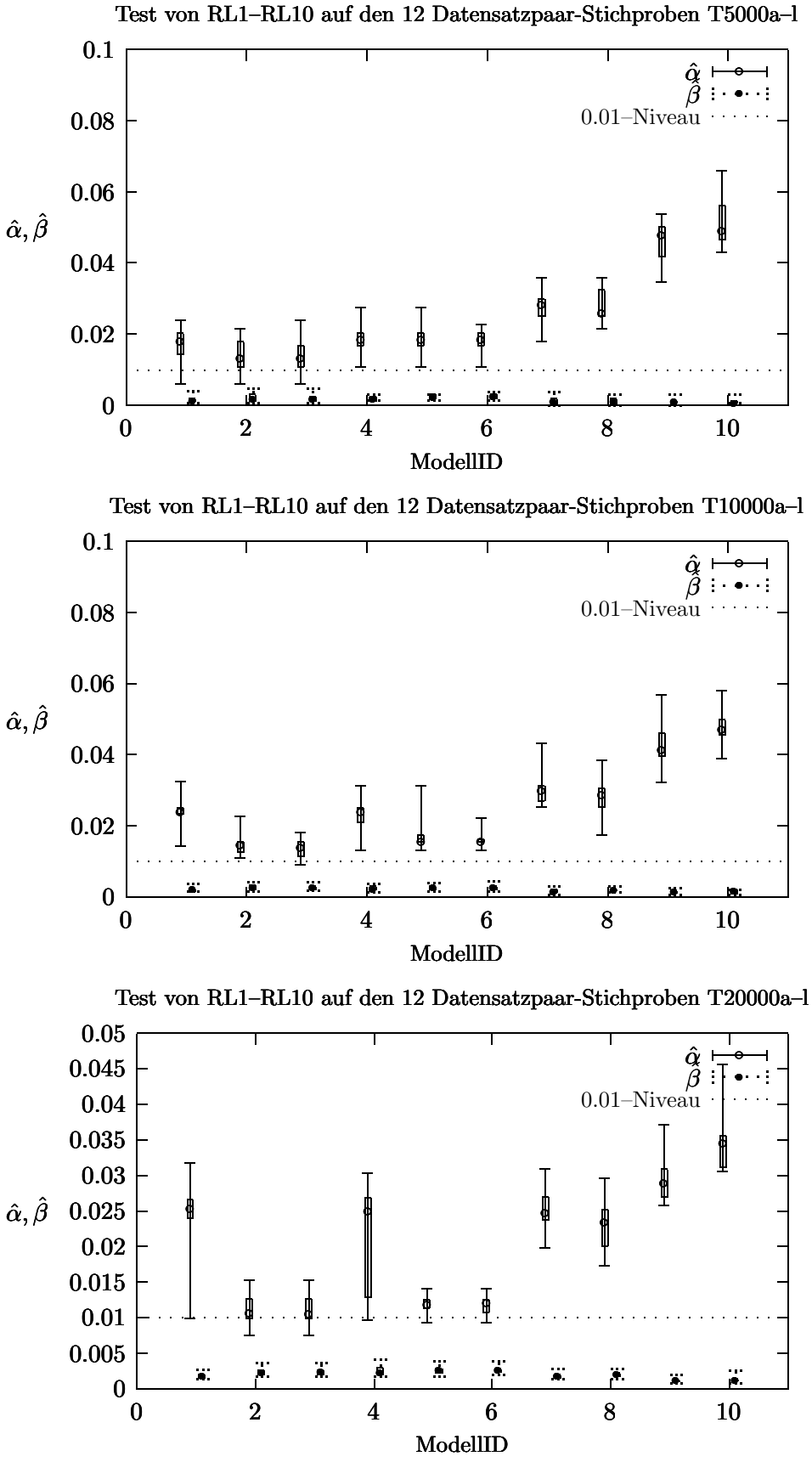


Abbildung 6.10: Die Ergebnisse von Record Linkage für die Adreßdaten

6.4.1.2 Kontrollierbarkeit der Fehlerraten

Wir untersuchen die Kontrollierbarkeit der Fehlerraten einer Klassifikation. Während für das Entscheidungsbaumverfahren nur eine geringe Variabilität gegeben ist (der Wert für *Decision* wird aus der Confidence der verwandten Entscheidungsregel abgeleitet und ist daher zumeist nahe 0 bzw. 1), ermöglicht der Likelihood-Quotient bei Record Linkage und der bei AssoClass aus den passenden Assoziationsregeln abgeleitete Wert eine hohe Variabilität. Wir untersuchen exemplarisch einen Fall, bei dem keine offenen Entscheidungen zugelassen wurden (d.h. *undetermined*= 0). Wir betrachten AssoClass mit dem Modell AC5, bei dem das Rang-Kriterium *Rank* für die Aggregation der passenden Assoziationsregeln verwandt wurde. Die Ergebnisse für die Adreßdaten sind in den Plots der Abbildung 6.11 zu finden. Für *Lowerbound*=*Upperbound*= $0.5 + \epsilon$ erhalten wir die in Abbildung 6.7 gezeigten Fehlerraten, während wir für *Lowerbound*=*Upperbound*= $1 - \epsilon$ das Klassifikationsmodell AC4 (mit konfliktfreien Regeln) erhalten, vgl. Abbildung 6.8 ($\epsilon = 10^{-12}$).

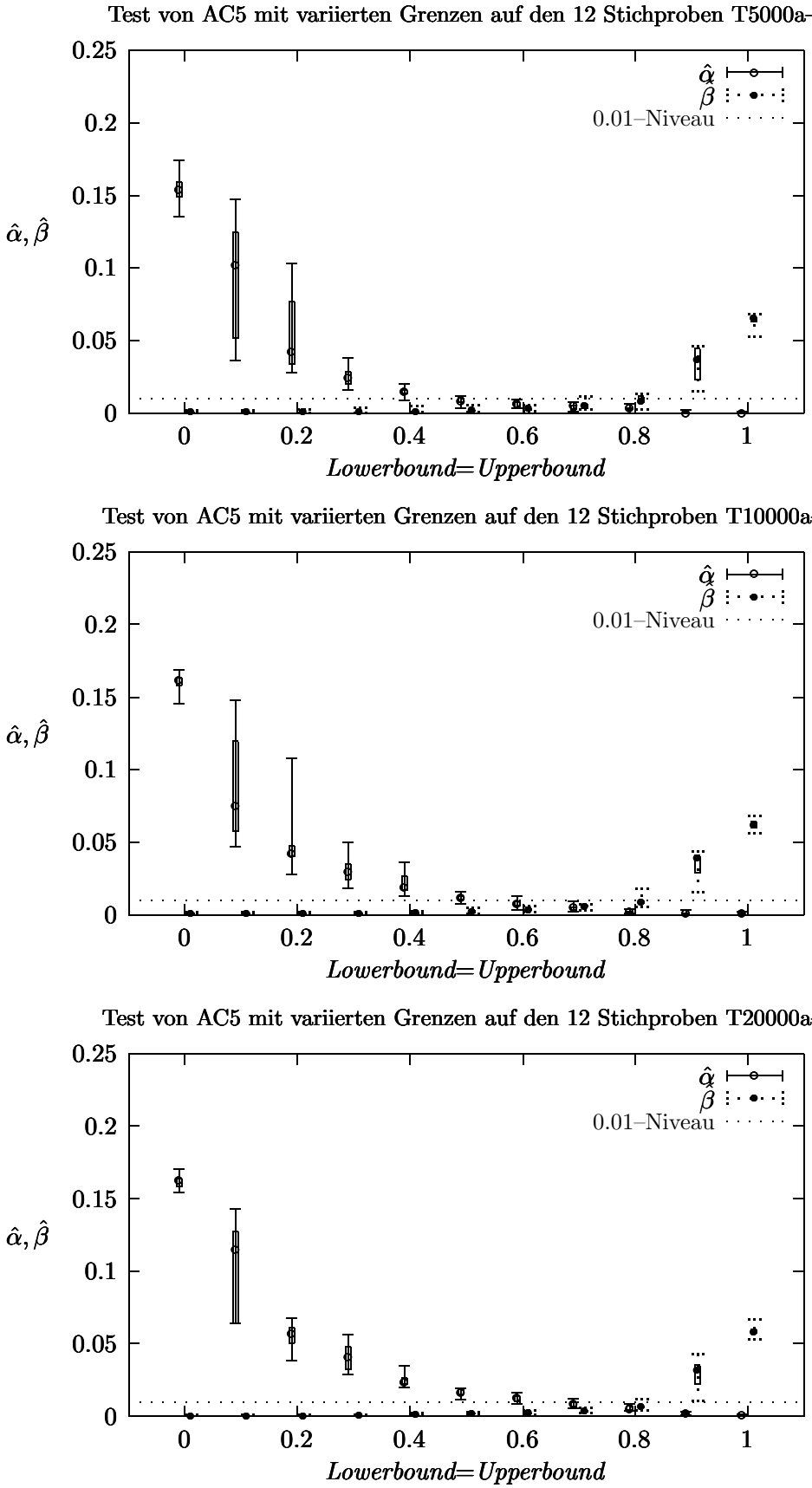


Abbildung 6.11: Die Ergebnisse von AC5 bei Variation der Grenzen der Entscheidungsfunktion (ohne offene Entscheidungen, d.h. mit *undetermined*= 0 und *Lowerbound*=*Upperbound*) für die Adreßdaten

6.5 Evaluation mit bibliographischen Daten

Aus dem bibliographischen Datenbestand des *Kooperativen Bibliotheksverbundes Berlin-Brandenburg* (`www.kobv.de`) wurde uns freundlicherweise ein Auszug von über 10 000 nicht bereinigten Datensätzen (aus verschiedenen Bibliotheken) zur Verfügung gestellt. Der Datenbestand war im sogenannten *MAB2*-Format kodiert (Das *MAB2*-Format ist ein spezielles Text-Datenformat für den Austausch von Daten zwischen Bibliotheken). Wir gewannen daraus Datensätze mit den für bibliographische Daten typischen Attributen: *Author* (Aufzählungstyp, da mehrere Autoren vorkommen), *Title*, *StdNo_Type* und *StdNo* (Art einer standardisierten Nummer, z.B. ISBN oder ISSN und die Nummer selbst) sowie *Year*, *Pages*, *Edition*, *Publisher*, *Address* und *Organization*. Der Datenbestand wies einen hohen Anteil fehlender Werte auf, lediglich für den Titel fand sich zumeist ein Wert.Für die Gewinnung der *Same*-Relation konnten wir die Attribute *StdNo, StdNo_Type* als Identifizierer ausnutzen und erhielten knapp 2 000 dublette Datensatz-Paare. Aufgrund fehlender oder fehlerhafter Werte mußten wir jedoch manuell etwa 900 dublette Datensatz-Paare hinzufügen. Insgesamt fanden wir über 2 800 geordnete, dublette Datensatz-Paare ($TC(Same) = 2\,837$) mit einem hohen Anteil mehrfacher Duplikate. Die optimierte *Same*-Relation enthielt lediglich 1 825 Einträge (d.h. es gab 1 825 Duplikate).

Da eine vorhandene *StdNo* (z.B. eine ISBN) einen globalen Schlüssel für Auflagen von Büchern darstellt, betrachten wir zwei unterschiedliche Gruppen von Klassifikationsmodellen bei der Identifizierung:

- Modelle, die die Werte des Attributes *StdNo* für die Klassifikation verwenden (AC4–AC6,DT1–DT6 und RL6+RL7) sowie
- Modelle ohne Verwendung der Attribute *StdNo* (AC1–AC3, AC4–AC6, DT7–DT12[18] und RL1–RL5, RL8+RL9).

In der Tabelle 6.11 sind die verwandten Attribute und Vergleichsfunktionen notiert, die Charakteristika der Attribute sind in den Tabellen 6.12 und 6.13 zu finden.

[18]Die Modelle DT7–DT12 verwenden identische Parameter wie die Modelle DT1–DT6, lediglich das Attribut *StdNo* wird in ihnen nicht benutzt.

Tabelle 6.11: Bibliotheksdaten: Attribute und Vergleichsfunktionen
Die Funktion *fctCompareMultipleAuthors* führt einen 8-stufigen Vergleich aus, 0: falls alle Autoren übereinstimmen, 1: falls für (mindestens) einen Autor Vor- und Nachname übereinstimmen, 2: falls mindestens ein Name übereinstimmt, 3–6: Werte der MinimumEditDistance des am besten passenden Namens und schließlich 7: für NULL-Werte.

Nr.	Attributname	Vergleichsfunktion	#Werte	Skala
1	Authors	fctCompareMultipleAuthors	8	ordinal
2	TitleSameWords	fctPercentageSameWordsDiscrete	4	ordinal
3	Year	fctYesNoNull	3	nominal
4	Pages	fctYesNoNull	3	nominal
5	Publisher	fctDiscreteMinEditDistance	5	ordinal
6	Address	fctDiscreteMinEditDistance	6	ordinal
7	Organization	fctDiscreteMinEditDistance	7	ordinal
8	StdNo	fctYesNoNullBothNull	4	nominal
9	LNameFirstAuthor	fctDiscreteMinEditDistance	6	ordinal
10	FNameFirstAuthor	fctDiscreteMinEditDistance	6	ordinal
11	TitleMinEdit	fctDiscreteMinEditDistance	6	ordinal
12	Edition	fctHasNonemptyIntersection	2	nominal

Tabelle 6.12: Die Charakteristika für die Bibliotheksdaten (Teil 1)

Attribut	accuracy	confidence	anti-confidence	nulls	selectivity
Edition	0.4258	0.9523	0.9863	0.7084	0.0091
Authors	0.5456	0.9997	0.9947	0.1840	0.5816
Year	0.8950	0.9769	0.9974	0.1071	0.0224
Organization	0.0204	0.9992	0.9333	0.9379	0.0440
LNameFirstAuthor	0.6116	0.9993	0.9971	0.1875	0.4267
Address	0.3842	0.9316	0.9937	0.2000	0.0627
StdNo	0.6754	0.9999	0.9906	0.4741	0.3969
Title	0.6711	0.9993	0.9927	0.0122	0.7358
Pages	0.8583	0.9985	0.9977	0.2513	0.0826
Publisher	0.7385	0.9245	0.9938	0.2458	0.1794
FNameFirstAuthor	0.5918	0.9919	0.9965	0.1875	0.1543

Tabelle 6.13: Die Charakteristika für die Bibliotheksdaten (Teil 2)

Attribut	delta-confidence	delta-accuracy
Authors	0.7385	0.9630
Year	0.9214	0.9359
Organization	0.9552	0.5225
LNameFirstAuthor	0.6250	0.9785
Address	0.8195	0.8502
Title	0.7078	0.9994
Publisher	0.8181	0.8436
FNameFirstAuthor	0.6066	0.9658

Als Abstandsmaße für die delta-confidence und delta-accuracy verwandten wir die *MinimumEditDistance* für *Authors* (mit $\Delta = 7$), für *Organization* (mit $\Delta = 10$), für *FNameFirstAuthor* und *LNameFirstAuthor* (mit $\Delta = 3$) und für *Publisher* und *Address* (mit $\Delta = 4$), die Funktion ($1-$*fctPercentageOfMatchingBiGrams*) für *Title* (mit $\Delta = 0.2$) und den Absolutbetrag für *Year* (mit $\Delta = 1$).

Tabelle 6.14: Bibliotheksdaten: Die Klassifikationsmodelle RL1–RL9 für Record Linkage

model	interaction terms	attributes
RL1	{1-way}	{1 3 4 11},
RL2	{2-way}	{1 3 4 11}
RL3	{3-way}	{1 3 4 11}
RL4	{1-way}	{1 2 3 4}
RL5	{2-way}	{1 2 3 4}
RL6	{2-way}	{1 2 3 4 8}
RL7	{3-way}	{1 2 3 4 8}
RL8	{1-way}	{1 2 3 4 9}
RL9	{1-way}	{1 2 3 4 5 7 9}

Tabelle 6.15: Bibliotheksdaten: Die Klassifikationsmodelle AC1–AC9 für AssoClass

model	rule aggregation method	attributes
AC1	NonConflicts	{1 3 4 11}
AC2	Confidence	{1 3 4 11}
AC3	Rank	{1 3 4 12}
AC4	NonConflicts	{1 2 3 4 8 11}
AC5	Confidence	{1 2 3 4 8 11}
AC6	Rank	{1 2 3 4 9 12}
AC7	NonConflicts	{1 2 3 4 10 11}
AC8	Confidence	{1 2 3 4 9 11}
AC9	Rank	{1 2 3 4 9 11}

6.5.1 Ergebnisse der Tests

Wir geben in den Abbildungen jeweils die Werte für den *alpha*- und *beta*-Fehler bzw. *precision* und *recall* wieder. Wie schon bei den Wohnungs- und Adreßdaten sticht das Entscheidungsbaum-Verfahren durch die niedrigsten Fehlerraten hervor. Die Verwendung des Attributes *StdNo* bei den Modellen (AC4–AC6,DT1–DT6 und RL6+RL7) führt erwartungsgemäß zu niedrigeren Fehlerraten (bzw. höheren Werten für *precision* und *recall*), wobei diese für die Modelle DT1–DT6 nur unwesentlich besser sind als die der Modelle DT7–DT13. Lediglich die Modelle RL6+RL7 für Record Linkage zeigen eine Verschlechterung gegenüber den Modellen RL4+RL5 (die bis auf die Nicht-Verwendung des Attributs *StdNo* identisch sind). Das läßt sich durch den höheren Schätzfehler für die Multinomialverteilung erklären, die dann zu einer schlechteren Anpassung an die Daten führt, vgl. die Diskussion in der Auswertung der Tests, Abschnitt 6.6.1.1 auf Seite 193. Die Klassifikation mit AssoClass führt für die Bibliotheksdaten zu keinen zufriedenstellenden Resultaten, selbst bei Verwendung der *StdNo* in den Modellen AC4–AC6 erhalten wir deutlich schlechtere Werte als die Modelle RL1–RL5 und DT1–DT12. Aufgrunddessen untersuchen wir hier auch nicht die Skalierbarkeit der Fehlerraten. Das schlechte Abschneiden von AssoClass und mögliche Verbesserungen diskutieren wir im Anschluß in der Auswertung der Tests im Abschnitt 6.6.1.2 auf Seite 196.

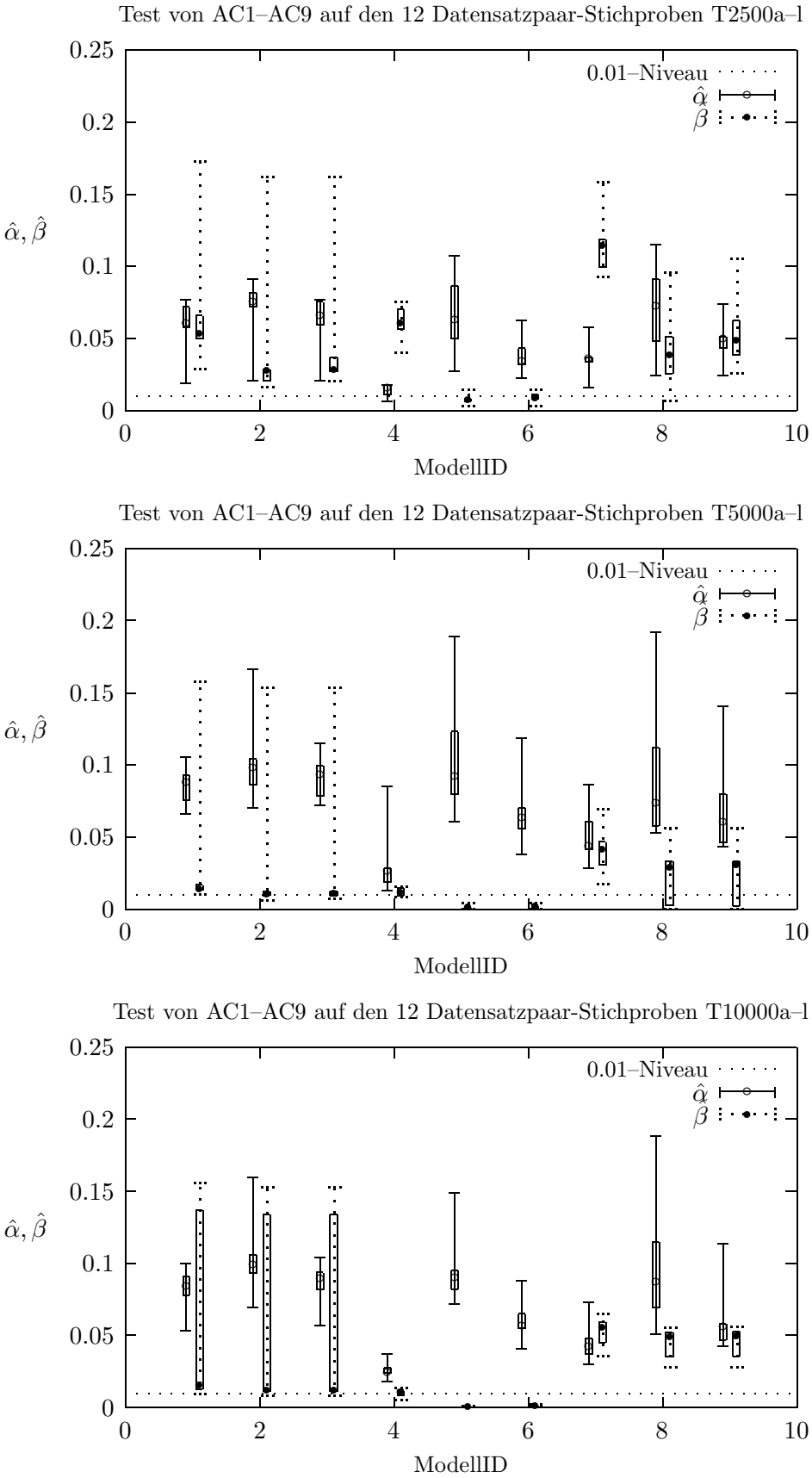


Abbildung 6.12: Die Ergebnisse (1) von AssoClass für die Bibliotheksdaten

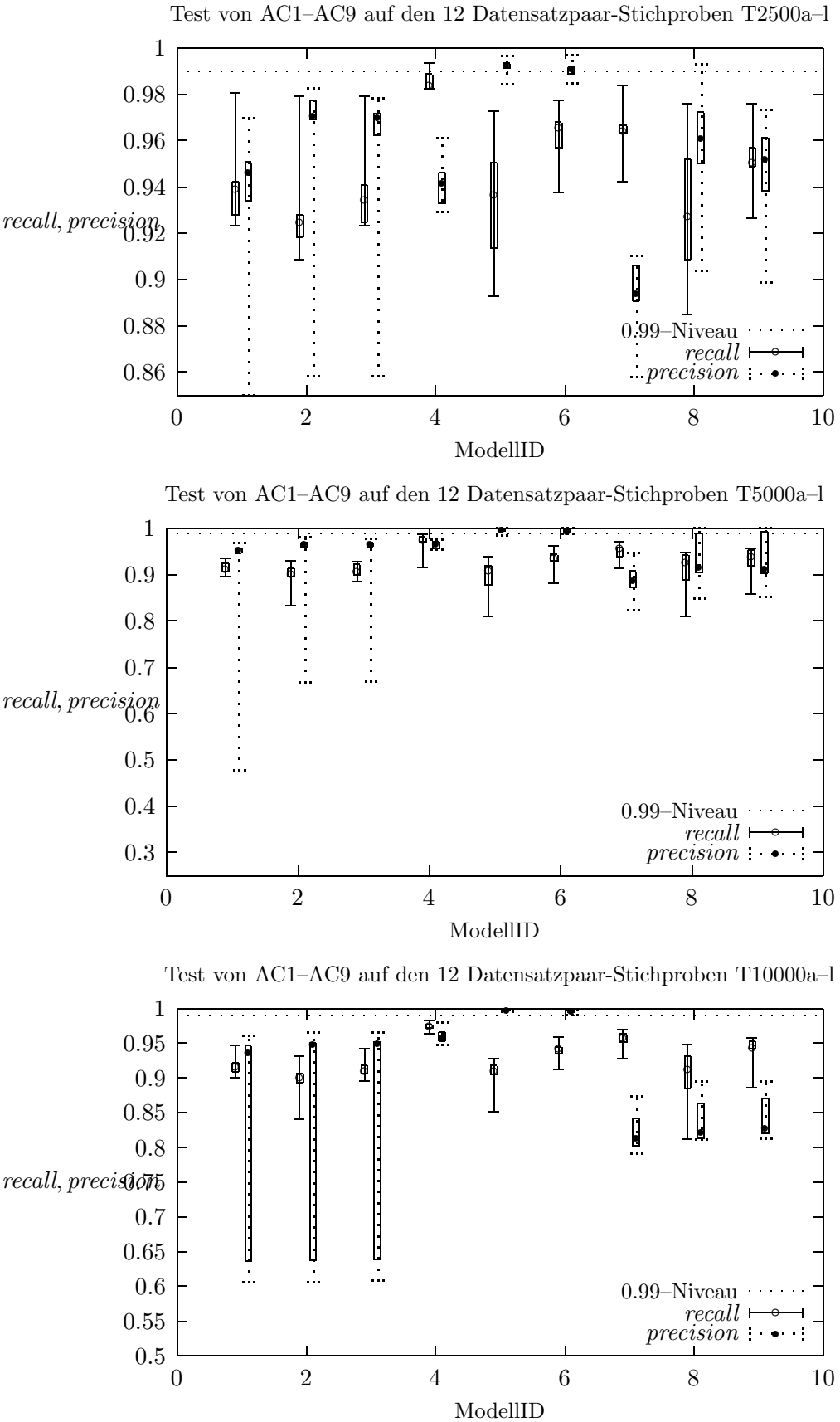


Abbildung 6.13: Die Ergebnisse (2) von AssoClass für die Bibliotheksdaten

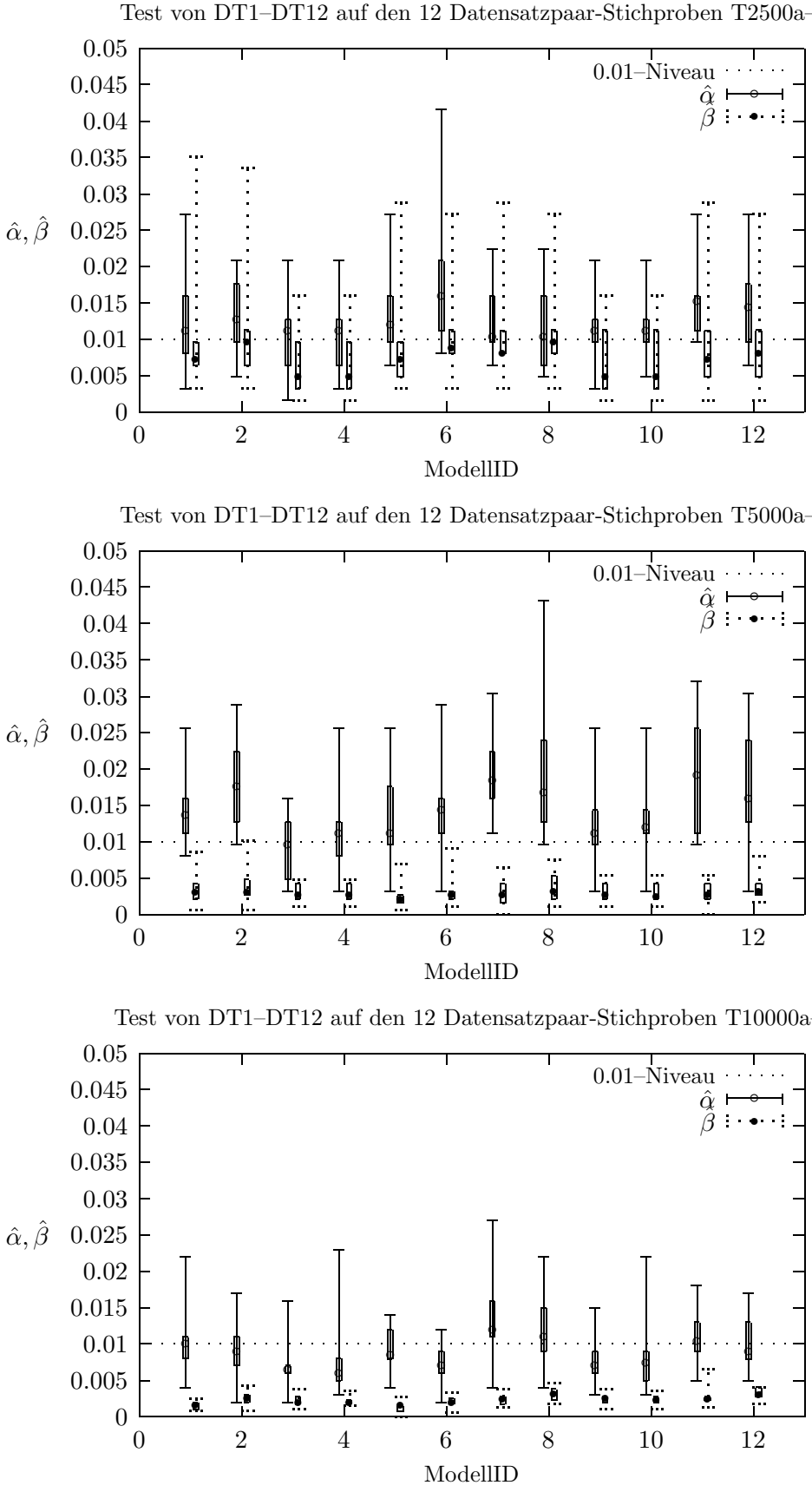


Abbildung 6.14: Die Ergebnisse (1) des Entscheidungsbaumverfahrens für die Bibliotheksdaten

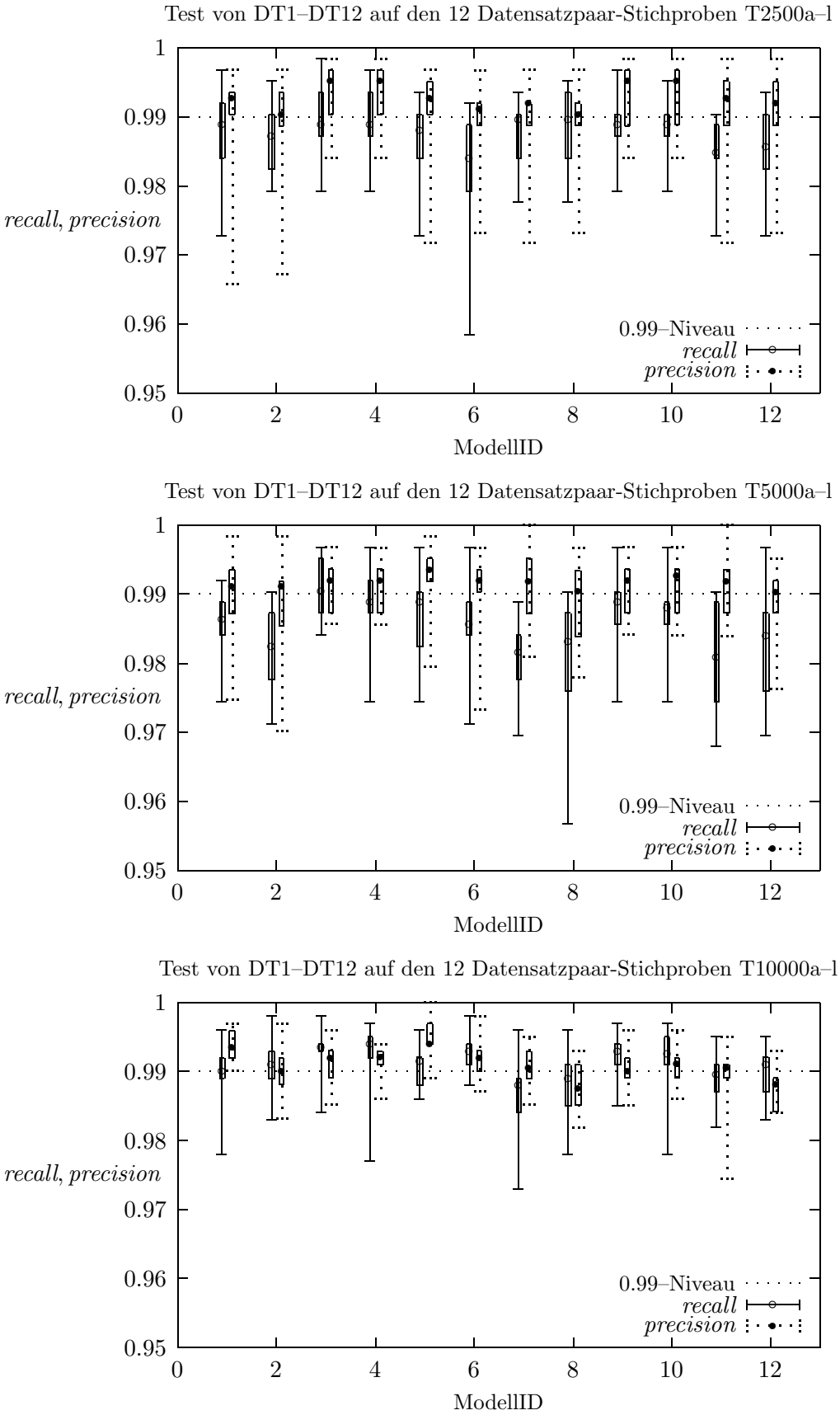


Abbildung 6.15: Die Ergebnisse (2) des Entscheidungsbaumverfahrens für die Bibliotheksdaten

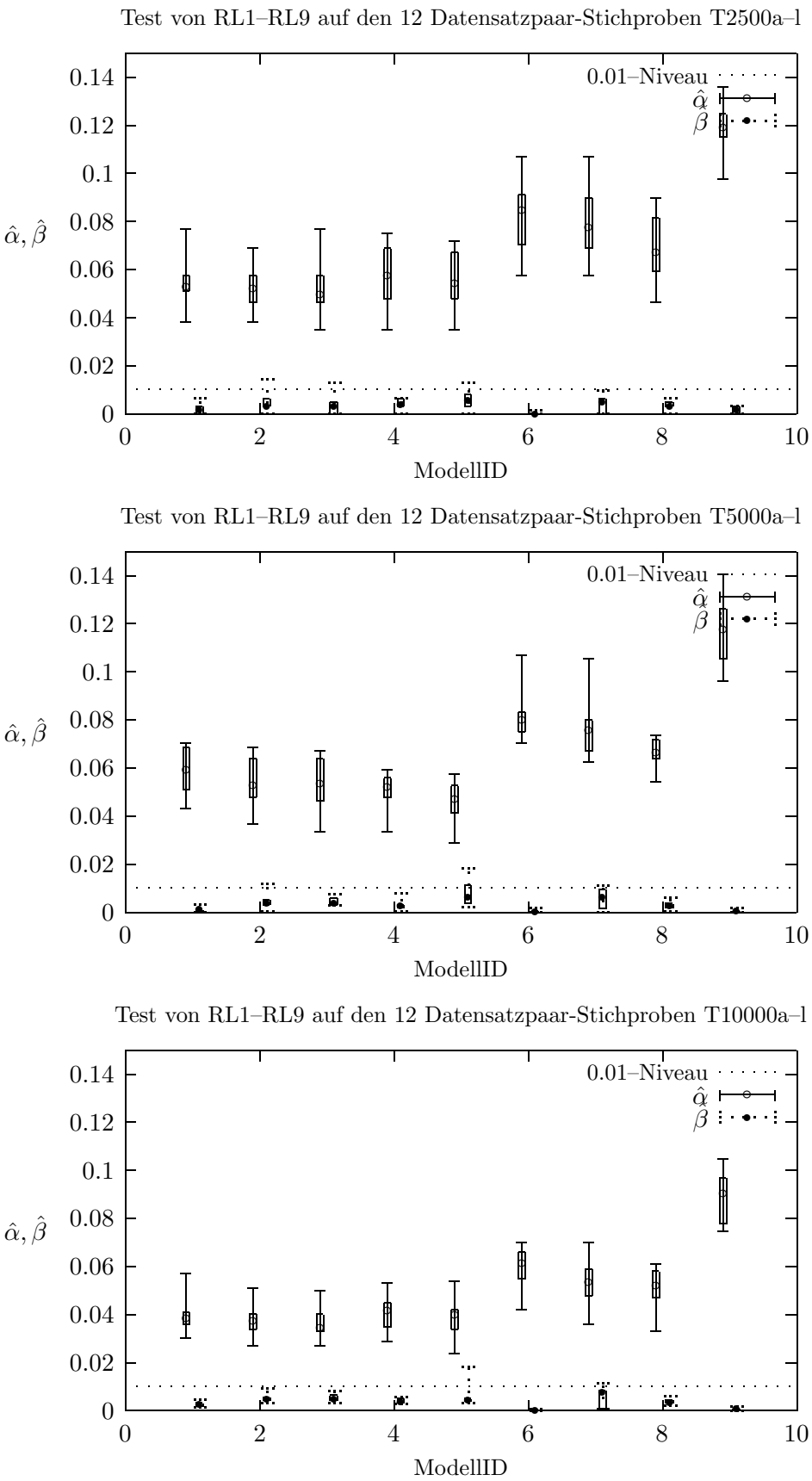


Abbildung 6.16: Die Ergebnisse (1) von Record Linkage für die Bibliotheksdaten

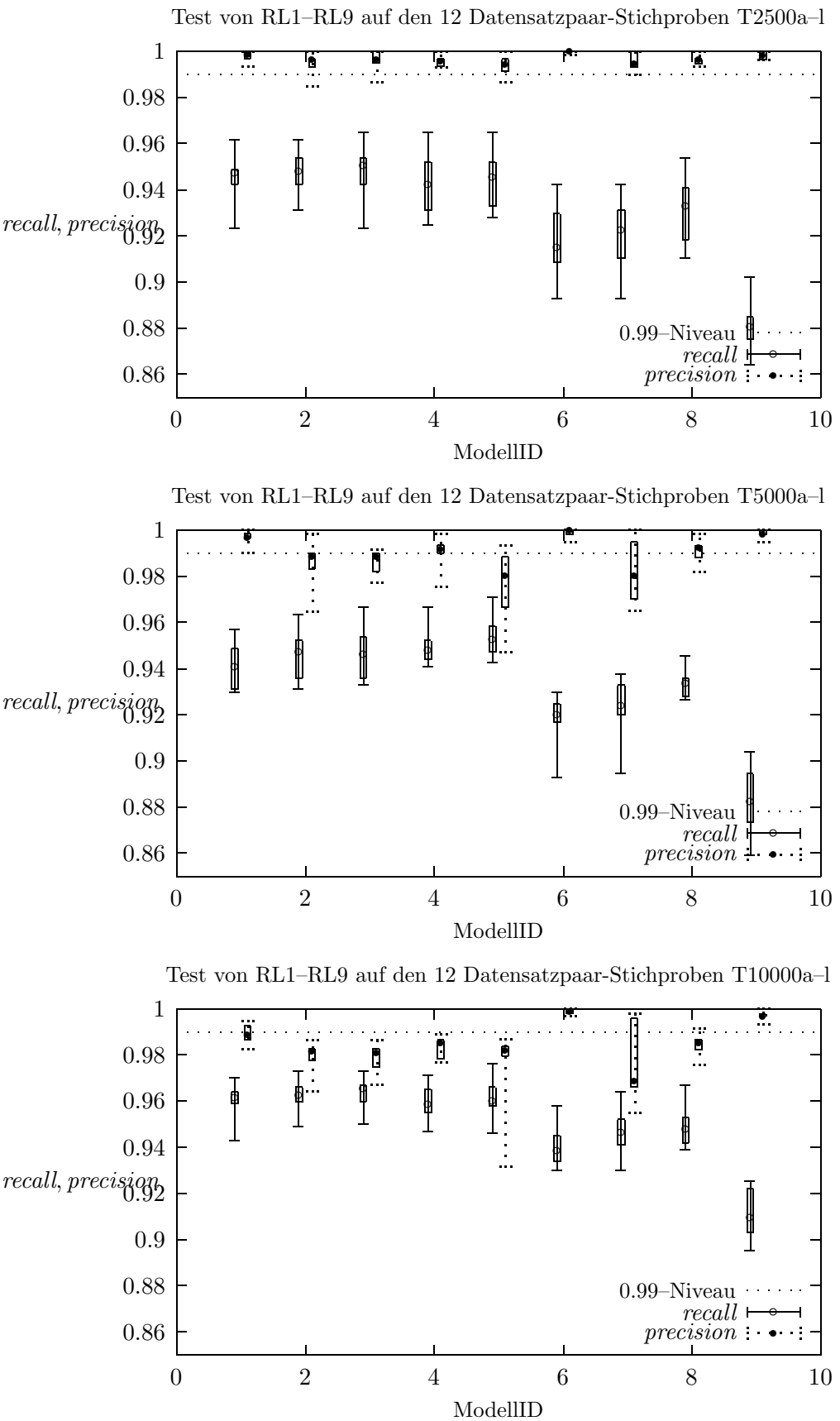


Abbildung 6.17: Die Ergebnisse (2) von Record Linkage für die Bibliotheksdaten

6.6 Auswertung

6.6.1 Ergebnisse

Zunächst fassen wir die Ergebnisse der drei Tests zusammen.

Das Entscheidungsbaumverfahren

- Das Entscheidungsbaumverfahren zeigt in allen Tests die besten Ergebnisse aller drei Methoden, d.h. (1) *hohe Korrektheit:* die Fehlerraten sind ausnahmslos sehr gering für alle Stichproben (Median unterhalb 1%), und (2) *große Stabilität:* eine deutliche Erhöhung der Anpassungsgüte —d.h. eine Verringerung des Streuungsintervalles $(\max - \min)$— läßt sich durch eine Erhöhung der Stichprobengröße erreichen (sehr deutlich läßt sich dieses aus den Tests für die Bibliotheksdaten ersehen, die jeweils vorgenommene Verdopplung des Anteils nicht-dubletter Paare in den Stichproben senkte den resultierenden $\hat{\beta}$-Fehler vgl. Abbildung 6.14 auf Seite 188).

- Die Wahl des Maßes für die Partitionierung hat keinen erkennbaren Einfluß auf die Korrektheit (wir verwendeten die drei Maße *Informationsgewinn*, *Informationsgewinnverhältnis* und *Gini-Index*).

- Ebenso ohne Einfluß auf die Korrektheit bleibt das zusätzliche Beschneiden (*engl.: prunen*) eines gelernten Entscheidungsbaumes. Bei einigen Stichproben verringert sich der Klassifikationsfehler, bei anderen erhöht er sich durch das Beschneiden. Dieses Resultat liegt darin begründet, daß bei der sehr hohen Anpassungsgüte (α- und β-Fehler geringer als 1%) auf unseren Stichproben durch eine Verkürzung des Baumes nicht notwendigerweise eine weitere Verbesserung der Anpassungsgüte zu erreichen ist.

- Obwohl alle der sechs getesteten Klassifikationmodelle ein ähnliches Verhalten zeigen, sticht das Modell DT3, bei dem das *Informationsgewinnverhältnis* ohne *pruning* zum Einsatz kam, heraus (bzw. die Modelle DT3, DT9 und DT15 für die Bibliotheksdaten).

- Wie in Abschnitt 6.2.4.1 erläutert, verwendeten wir beim Entscheidungsbaumverfahren ordinale Skalen für diejenigen Vergleichsattribute, die ordinale Skalen hatten und andernfalls nominale Skalen (kardinale Skalen traten nicht auf, können aber ebenfalls verwandt werden). Da in den Tests der anderen beiden Klassifikationsmethoden ordinale Skalen als nominale Skalen verwandt wurden, wollen wir prüfen, inwiefern die Berücksichtigung der Skalierung Einfluß auf die Klassifikationsgüte hat. Dazu wendeten wir das Entscheidungsbaumverfahren exemplarisch auf die Bibliotheksdaten bei *nominaler Skalierung* der ausgewählten Attribute an (ohne *Std_No*, wir haben die Paare DT7/DT13,...,DT12/DT18, die bis auf die erwähnte Skalierung identisch sind, DT13–DT18 haben nominale Skalen). Die Ergebnisse sind in der Abbildung 6.18 zu finden. Es zeigt sich, daß das Ausnutzen der

Ordinal-Skala in allen Modellen zu einer Verbesserung der Klassifikationsgüte führt. Doch auch ohne Verwendung dieser Skalierung sind die Fehlerraten deutlich unterhalb derjenigen von Record Linkage oder AssoClass, vgl. die Abbildungen 6.12 und 6.16 auf Seite 190ff.

6.6.1.1 Record Linkage

Zusammenfassend läßt sich feststellen, daß unter den getesteten Klassifikationsmodellen von Record Linkage diejenigen mit wenigen Attributen (d.h. 4–5) bessere Ergebnisse zeigten als die mit einer größeren Zahl an Attributen. Die Fehlerraten fallen jedoch deutlich höher aus als beim Entscheidungsbaumverfahren (etwa doppelt so hoch). Um zu überprüfen, ob bei Anwendung des Entscheidungsbaumverfahrens auf diese wenigen Attribute ähnlich hohe Fehlerraten auftreten, wandten wir die Klassifikationsmodelle DT1–DT6 für die Bibliotheksdaten erneut an. Das geschah unter ausschließlicher Verwendung der Attribute wie $Authors, TitleMinEdit, Year, Pages$, die schon bei Klassifikationsmodellen RL1–RL3 von Record Linkage verwandt wurden. Die Skalierung wurde für alle Attribute nominal gesetzt. Die Ergebnisse sind in der Abbildung 6.19 zu finden, die entsprechenden Fehlerraten für die Modelle RL1–RL3 sind ebenfalls dort abgebildet. Es zeigt sich, daß sich zwar die Fehlerraten des Entscheidungsbaumverfahrens deutlich verschlechtern (vgl. Abbildung 6.18), jedoch liegen sie immer noch unterhalb derer von Record Linkage. Wir schließen daraus, daß das Entscheidungsbaum-Verfahren für diese Stichproben bei gleicher Information über die Paare (identische Attributauswahl, Skalen nominal) besser in der Lage ist, die klassifizierende (identifizierende) Information auszunutzen. Man ersieht jedoch, daß bei diesen wenigen Attributen keine Stabilität vorliegt (d.h. keine verbesserte Anpassung durch Erhöhung der Stichprobengröße).

In den Klassifikationsmodellen für Record Linkage haben wir die Ordnung der maximalen Interaktionen zwischen den Attributen (1-Faktor-, 2-Faktor- oder 3-Faktor-Interaktionen) sowie die Attributauswahl variiert. Beide Variationen beeinflussen die Klassifikationsgüte:

- Einerseits läßt sich feststellen, daß die Attributauswahl einen starken Einfluß auf die Klassifikationsgüte hat, d.h. mit zunehmender Zahl an Attributen steigen die Fehlerraten und ihre Streuungsbreite deutlich, und
- Andererseits beeinflußt die gewählte maximale Ordnung der Interaktionen ebenfalls die Güte, sofern Abhängigkeiten zwischen den ausgewählten Attributen bestehen, die Fehlerraten und ihre Streuungsbreite verringern sich in diesem Fall.

Die sich verschlechternde Anpassung bei Hinzunahme weiterer Attribute liegt in der daraus resultierenden Modellgröße begründet, da die Zahl der Vergleichswert-Kombinationen exponentiell mit der Attributanzahl wächst, wir erhalten schon für *zwei* Vergleichswerte je Attribut eine Zahl von 2^k

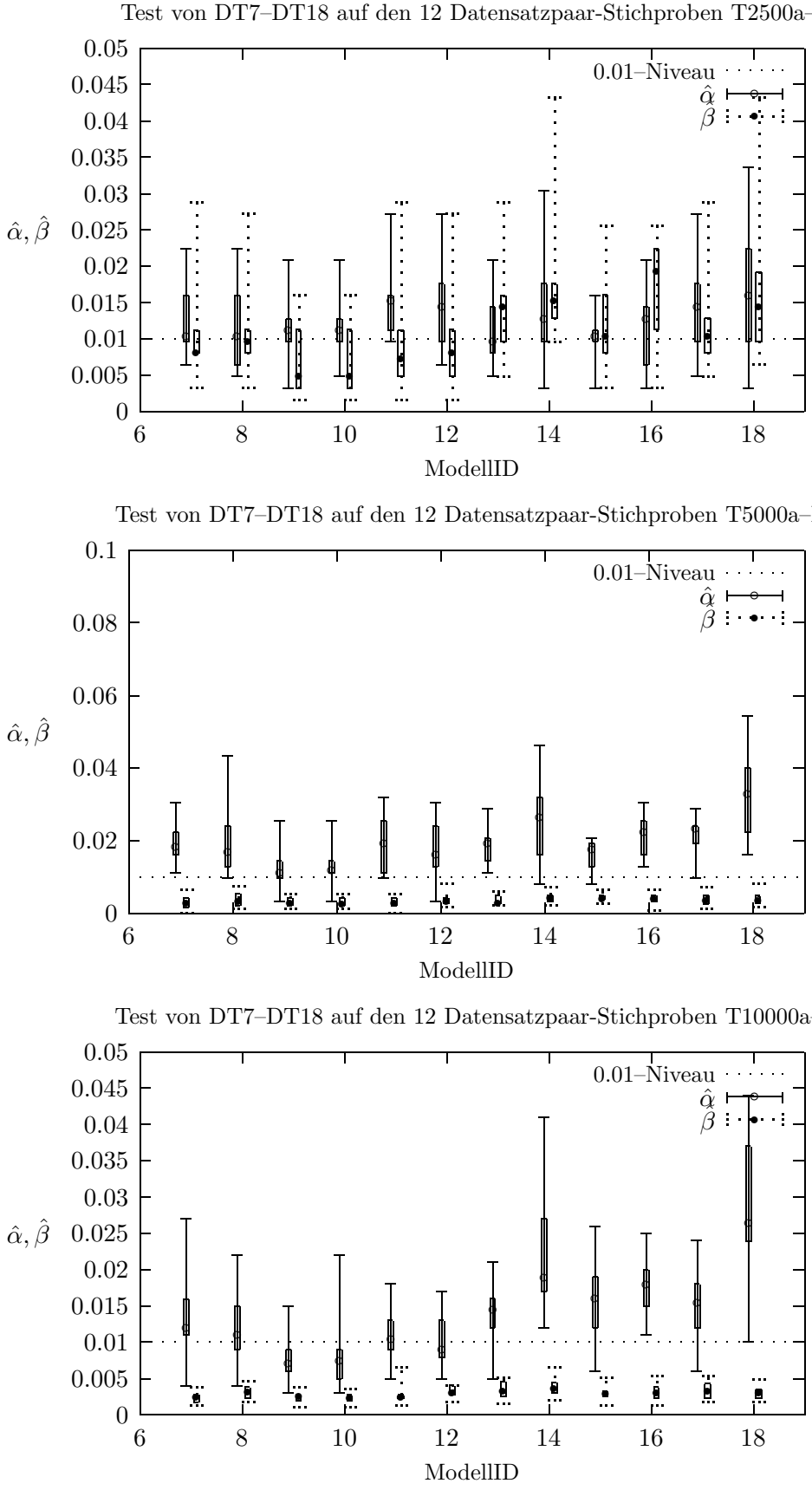


Abbildung 6.18: Die Ergebnisse des Entscheidungsbaumverfahrens auf den Bibliotheksdaten bei Variation der Skalen (nominal/ordinal für DT7–DT12, ausschließlich ordinal für DT13–DT17)

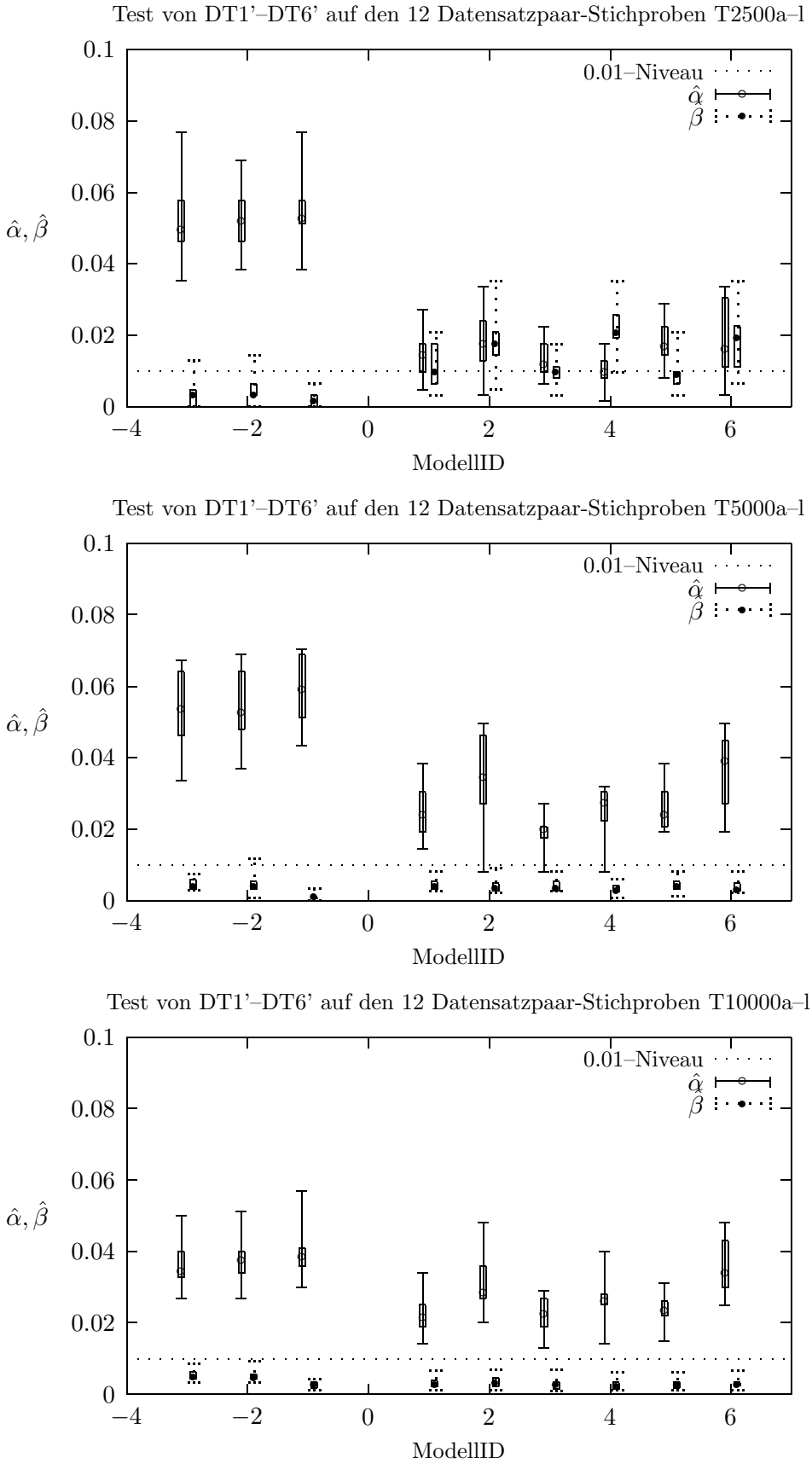


Abbildung 6.19: Die Ergebnisse von DT1'–DT6' auf den Bibliotheksdaten bei Restriktion auf dieselben Attribute wie die Modelle RL1–RL3 (vgl. Abbildung 6.16, RL1–RL3 sind hier zum Vergleich abgebildet als ModellID's -1,-2 und -3)

verschiedenen Fällen. Problematisch für die Schätzung der Multinomialverteilung ist aber nicht nur die wachsende Anzahl der Fälle, sondern auch die große Zahl von nicht realisierten Fällen (sogenannten leeren Zellen). Obwohl diese nicht realisierten Fälle für die Schätzung der Multinomialverteilung mit der Wahrscheinlichkeit 0 gesetzt werden (*engl.: fitted zeros*), ist der Schätzer verzerrt und hat deshalb höhere Fehlerraten.

Diesem Umstand kann man begegnen, indem man zusätzliche Annahmen in die Schätzung der Multinomialverteilung für ein log-lineares Modell mit aufnimmt, beispielsweise die Annahme nach

- *Monotonie*: Forderung nach Monotonie der Wahrscheinlichkeitsverteilung bei ordinalen oder kardinalen Skalen (z.B. gelte für die Koeffizienten der zwei-Faktor Interaktionen zweier Vergleichs-Attribute R_i, R_j: $u_{ij}(r_i, r_j) < u_{ij}(r'_i, r'_j)$ falls $r_i < r'_i$ oder $r_j < r'_j$), oder
- *konstante Koeeffizienten*: Forderung nach einer konstanten Abhängigkeitsstruktur zwischen bestimmten Attributen (z.B. gelte für die Koeffizienten der drei-Faktor Interaktionen dreier Vergleichs-Attribute R_i, R_j, R_k: $u_{ijk}(r_i, r_j, r_k) = u_{ijk}(r'_i, r'_j, r'_k)$ für alle $r_i, r'_i, r_j, r'_j, r_k, r'_k$).

Mit zusätzlichen Annahmen verringert sich die Zahl der zu schätzenden Parameter der Multinomialverteilung, so daß die Anpassung einer Klassifikation sich erhöht.

Abschließend bemerken wir, daß beim Record Linkage eine große Stabilität gegeben ist: Mit wachsender Stichprobengröße verringert sich der Schätzfehler und mithin die Fehlerraten.

6.6.1.2 Die Klassifikation mit Assoziationsregeln (AssoClass)

Die Klassifikation mit Assoziationsregeln zeigt im Vergleich zu den beiden anderen Methoden—bis auf eine unten zu diskutierende Ausnahme— keine überzeugenden Ergebnisse. Das mag darin begründet liegen, daß die vorgestellten Aggregationsmethoden für widersprüchliche Regeln ein *lokaler Ansatz* sind, d.h. sie nutzen immer nur die Information (d.h. die Regeln) aus, die für einen Vergleichswert passen. Im Gegensatz dazu betrachtet man bei Lernen eines Entscheidungsbaumes immer Partitionen des Vergleichsraumes, und versucht diejenigen zu finden, die homogen bezüglich des klassifizierenden Attributes sind. Nichtsdestotrotz enthalten Assoziationsregeln die für eine Partitionierung nötige Information, die von den Entscheidungsbaum-Verfahren erzeugten Entscheidungsregeln sind ebenfalls enthalten. Weitergehende Forschung zu AssoClass wird sich unter anderem mit der Problematik auseinandersetzen, welche Vorgehen Entscheidungsbäume aus widerspruchsfreien Regeln erzeugen, die identisch/ähnlich sind zu denen des Entscheidungsbaumverfahrens. Diese können dann modifiziert werden, um wünschenswerte Klassifkationseigenschaften zu erreichen.

Wie für die Wohnungs- und Adreßdaten dargestellt, erlaubt die Vielzahl an Assoziationsregeln eine Skalierbarkeit der Fehlerraten, so daß alternativ der α- oder β-Fehler minimiert werden kann. Diese breite Skalierbarkeit ist

weder für das Entscheidungsbaum-Verfahren möglich, noch in diesem starken Maße für Record Linkage. Beim Entscheidungsbaum-Verfahren können wir die Confidence der Entscheidungsregel für die Skalierbarkeit verwenden, es sind aber nur wenige Stufen möglich (je eine pro Entscheidungsregel); Bei Record Linkage ist durch die Verschiebung der Grenzen λ_L, λ_U zwar eine Skalierbarkeit des Klassifikationsverhaltens gegeben, aufgrund des hohen Schätzfehlers der nicht oder selten beobachteten Fälle steigt die Fehlklassifikation des jeweils nicht kontrollierten Fehlers auf bis über 50%. Falls z.B. der *alpha*-Fehler minimiert werden soll, sinkt die *precision* der Klassifikation (auf Werte nahe Null falls der Anteil nicht-dubletter Paare in der Stichprobe groß ist), da zwar fast keine Duplikate mehr verworfen werden, dafür aber bis über 50% der nicht-dubletten Paare fälschlicherweise als Duplikate klassifiziert werden.

6.6.2 Einflußfaktoren für die Anpassungsgüte einer Klassifikation

Im Folgenden diskutieren wir, welche Faktoren starken Einfluß auf die Klassifikation haben. Wir unterscheiden dabei zwischen

- Faktoren, die durch die Daten gegeben sind, sowie
- Faktoren, die durch eine Klassifikationsmethode und ihre Parametrisierung bestimmt sind.

Wir diskutieren im Folgenden die Daten-abhängigen Faktoren. Die durch die Klassifikationsmethoden bestimmten Faktoren haben wir im vorigen Abschnitt besprochen. Starken Einfluß haben die in Kapitel 4 diskutierten Qualitätsaspekte. Wir gehen auf die zwei wichtigsten semantischen Constraints näher ein:

- *semantische Schlüssel*: Je höher die Güte (confidence und accuracy) der approximativen Schlüssel ist, und umso geringer der Anteil der Tupel mit NULL-Werten für die in solchen Schlüsseln enthaltenen Attribute ist, desto genauer und vollständiger können diese Schlüssel auf die Daten angewandt werden. Falls semantische Schlüssel existieren (und diese korrekt sind) ist eine Identifikation mit ihnen möglich (die ISBN/ISSN ist ein Beispiel für einen semantischen Schlüssel, es gab aber bei den Bibliotheksdaten einen hohen Anteil von Datensätzen mit NULL-Werten.
- *Anzahl von Duplikaten und Überlappungsgrad von Teilmengen*: Falls genau eingegrenzt werden kann, wieviele Duplikate für einzelne Datensätze oder in Teilmengen von A zu finden sind, kann die Klassifikation darauf abgestimmt werden.

Beispiel 6.13 *(bekannter Überlappungsgrad).* Eine Tabelle A lasse sich in zwei Teilmengen A_1, A_2 zerlegen und es sei bekannt, daß beide Teilmengen

duplikatfrei sind sowie daß zu jedem Datensatz aus A_1 genau ein Duplikat in A_2 existieren muß, d.h. es gelten die semantischen Constraints

$$[\text{duplicates}(A_i) = 0],\ i = 1, 2$$
$$[\text{overlap}(\{a\}, A_2) = 1],\ a \in A_1.$$

Für eine durchgeführte Klassifikation von Paaren (sinnvollerweise mit einem Selektor $\sigma(A \times A) \subset A_1 \times A_2$) kann im Anschluß eine Bereinigung durchgeführt werden. Es kann jeweils der beste Treffer

$$(a, b) := \text{argmin}\big(\delta(a, b) \mid (a, b) \in A_1 \times A_2\big)$$

aus $(a, b) \in A_1 \times A_2$ als Duplikat klassifiziert werden. Lediglich bei einer Bewertung nahe 1, oder in inkonsistenten Fällen (bei identischer Bewertung mehrerer Paare $(a, b), (a, b')$ oder falls $b \in A_2$ bester Treffer mehrerer Datensätze $a, a' \in A_1$ ist) ist manuell zu prüfen bzw. eine konfliktauflösende Regel anzuwenden (beispielsweise kann b demjenigen Datensatz als Duplikat zugeordnet werden, dessen Bewertung besser (d.h. niedriger) ausfällt).

Eine große Bedeutung für die Anpassungsmöglichkeit jeglicher Klassifikationsmethode kommt der Definition des Vergleichsraumes R zu. Der mehrdimensionale Vergleichsraum R wird aufgespannt durch die verwandten Vergleichsfunktionen $f = (f_1, \ldots, f_k)$—d.h. R ist die Menge aller ihrer Vergleichswert-Kombinationen (auch als Vergleichsvektoren bezeichnet). Trivialerweise können zwei Datensatzpaare nur dann unterschiedlich klassifiziert werden, wenn sie unterschiedliche Vergleichsvektoren haben. Für die Anpassung einer Klassifikation ist also relevant, inwiefern die bedingten Verteilungen (hier für den diskreten Fall angegeben)

$$p_0(r) := \mathbf{P}\{\, r = f(a, b) \in R \mid a \equiv b \,\}, \tag{6.21a}$$
$$p_1(r) := \mathbf{P}\{\, r = f(a, b) \in R \mid a \not\equiv b \,\} \tag{6.21b}$$

voneinander verschieden sind.[19]Die Wahrscheinlichkeiten $p_0(r)$ und $p_1(r)$ aus (6.21) lassen sich auf Stichproben von Datensatzpaaren mit der relativen Häufigkeit der Vergleichsvektoren schätzen, für die $Same$ = 'Yes' bzw. $Same$ = 'No' gilt.

Wir betrachten die beiden Extremfälle:

- Falls beide Verteilungen gleich sind, d.h. falls $P_0(r) \approx P_1(r)$ für alle $r \in R$ gilt, kann mit keinem Klassifikationsverfahren (aus Beispieldaten mit Werten aus R) ein gute Klassifikation gewonnen werden.

- Falls die Masse beider Verteilungen auf zwei Teilmengen $R', R'' \subset R$, $R' \cap R'' = \emptyset$ verteilt ist, d.h. falls ein kleines $c > 0$ existiert (etwa $c = 0.01$), so daß $\sum_{r \in R'} P_0(r) < c$ und $\sum_{r \in R''} P_1(r) < c$ gilt, wird es Klassifikationen geben, deren Fehler durch c beschränkt sind (d.h. $\alpha, \beta < c$).

[19]Es sei bemerkt, daß aus diesen beiden Verteilungen (6.21) (respektive ihrer Schätzungen) der Likelihood-Quotient beim Record Linkage berechnet wird.

Eine hohe *Trennschärfe* der Vergleichsvektoren im Vergleichsraum $r \in R$ bezüglich des klassifizierenden Attributes *Same* ist also eine gute Voraussetzung, um eine Klassifikation auf R zu lernen.

Beispiel 6.14 *(Simulation).* Die Trennschärfe sei mit einer Simulation von zweidimensionalen stetigen Vergleichswerten in [0,1] verdeutlicht, vgl. Abbildung 6.20 auf Seite 200. Wir zeigen drei Simulationen. Bei der 1. Simulation ist keine Trennung zwischen den Verteilungen beider Klassen zu erkennen, lediglich im linken unteren Rand ist eine der Klassen stärker vertreten. In der 2. Simulation ist eine mittlere Trennschärfe zu vermerken (die Masse der einen Verteilung liegt auf der linken unteren Seite von $[0,1]^2$ während die Masse der zweiten auf der rechten oberen Seite von $[0,1]^2$ liegt). Beide Verteilungen haben jedoch im mittleren Bereich (etwa bei $[0.2, 0.7] \times [0.3, 0.8]$) viele Werte, so daß eine Klassifikation in diesem Bereich ungenau sein wird, d.h. ein großer Anteil von Paaren mit diesen Vergleichswerten fehlerhaft klassifiziert werden wird. Die dritte Simulation zeigt das wünschenswerte Verhalten: die Klassen sind gut getrennt (in etwa durch die Gerade $y = 1 - x$), der Vergleichsraum zeigt eine hohe Trennschärfe auf. Der Fehler einer Klassifikation dürfte hier sehr gering ausfallen.

Bemerkung 6.15. Wie aus dem Beispiel 6.14 zu ersehen ist, ist die Verteilung der Vergleichsvektoren $r \in R$ im Vergleichsraum (falls die Information *Same* = 'Yes'/'No' nicht berücksichtigt wird) zusammengesetzt aus zwei Verteilungen, resultierend in einer Verteilung mit mehreren Modi. Sofern eine gute Trennschärfe der Modi gegeben ist, läßt sich diese Verteilung unter zusätzlichen Annahmen (z.B. Gaußsche Normalverteilung) als multimodale Verteilung schätzen. Wir erhalten dann also (i bezeichne die Modi)

$$\mathbf{P}\{\, r \in R \,\} = \sum_i \mathbf{P}\{\, r \in R : \, i\text{-ter Modus} \,\}\,.$$

Es sei also eine hohe Trennschärfe gegeben und die Wahrscheinlichkeit dubletter Paare in der Vorauswahl sei hinreichend hoch (z.B. bei sehr niedriger Selektivität und geringer Fehlerrate einer Vorauswahl). Dann kann zur Analyse von Vergleichswert-Häufigkeiten zufällig gebildeter Datensatzpaar-Stichproben (d.h. auch ohne Vorliegen einer *Same*-Relation) die Schätzung einer multimodalen Verteilung vorgenommen werden, vgl. z.B. Stützle [Stü03].

Nun wollen wir die Trennschärfe für die im Test benutzten Datenbestände untersuchen. Dazu bilden wir Kontingenztabellen[20] auf Datensatzpaar-Stichproben. In den Tests haben wir bis zu 17 Vergleichsattribute verwandt, zur Veranschaulichung betrachten wir eine Auswahl von Vergleichsattribut-Paaren mit zwei- und drei-dimensionalen Kontingenztabellen, jeweils separat für *Same* = 'Yes' und *Same* = 'No'. In den Abbildungen 6.21–6.26 sind die resultierenden Diagramme gezeigt, es ist die logarithmische Skalierung der z-Achse (Count) zu beachten.[21]

[20] In Microsoft Excel lassen sich Kontingenztabellen als *Pivot-Tabellen* darstellen, wenn man den *Count* für die Häufigkeit der Vergleichsvektoren als Maß verwendet.

[21] Wir verwenden die logarithmische Skalierung der z-Achse, da bei linearer Skalierung die kritischen Fälle kaum zu erkennen sind (das sind diejenigen, bei denen beide Verteilungen Werte größer Null aufweisen).

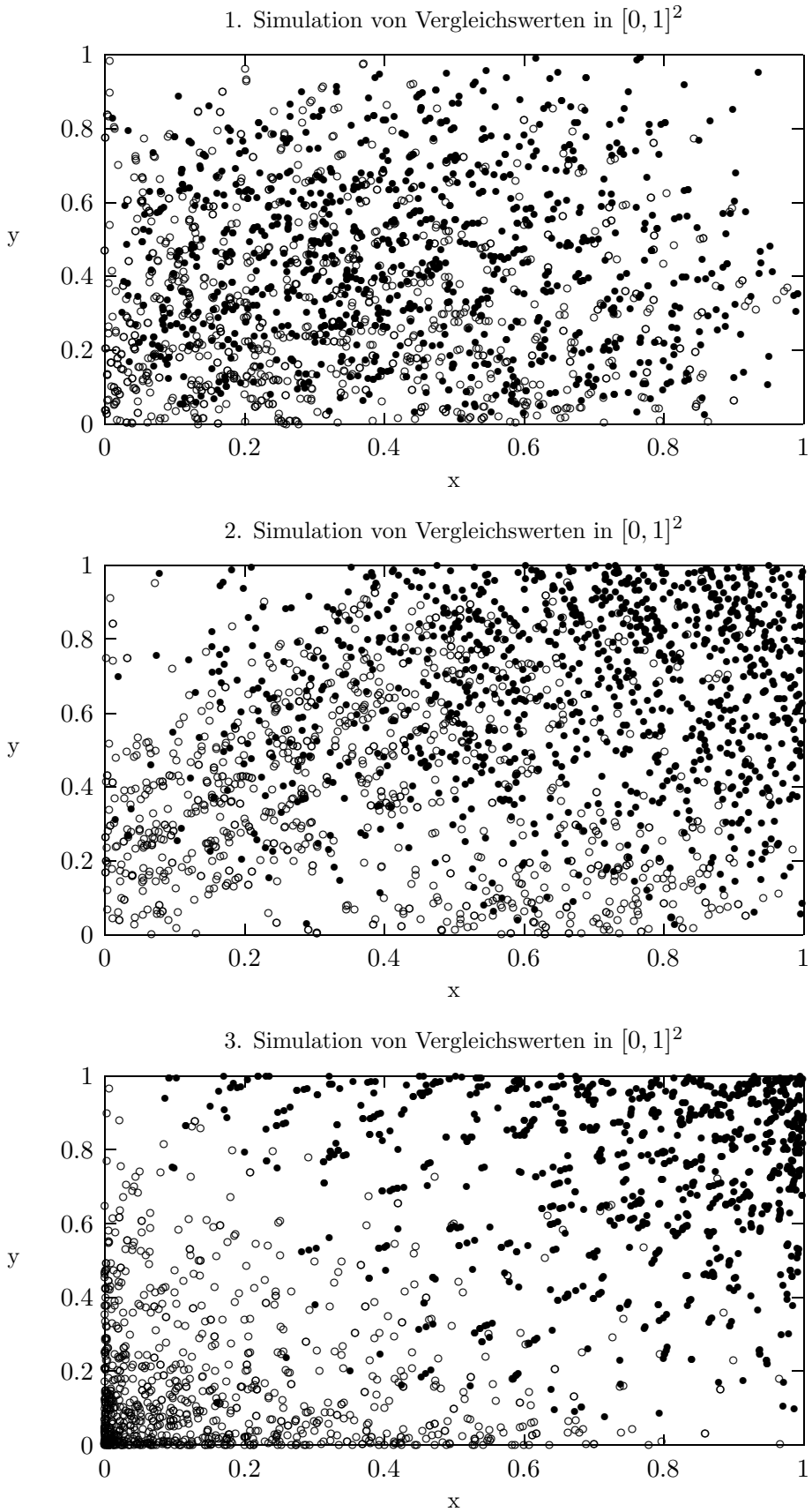


Abbildung 6.20: Simulationen zweier Klassen von zweidimensionalen Vektoren, die unterschiedliche Trennschärfe haben (auf einer kardinalen Skala)

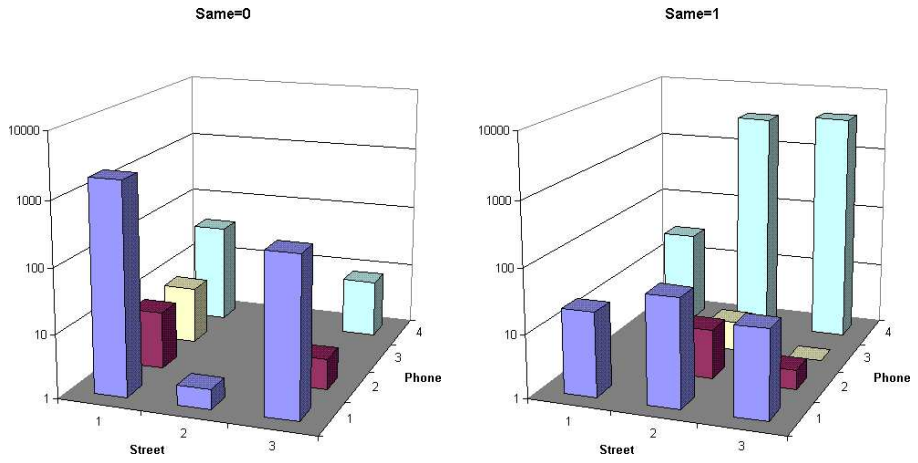


Abbildung 6.21: Häufigkeiten der Vergleichswert-Kombinationen für die Attribute *Street* und *Phone* (jeweils für Paare von Duplikaten $Same = 0$ und Nicht-Duplikaten $Same = 1$)

Für die Wohnungsdaten betrachten wir eine Kombination von Vergleichs-Attributen.

- *Street* und *Phone* (Abbildung 6.21): 97% der Werte für $Same =$ 'Yes' findet man bei den beiden Vergleichsvektoren $(Street, Phone) = (1, 1)$ und $= (1, 3)$, während für 98% der Paare mit $Same =$ 'No' die Vergleichswerte $(Street, Phone) = (2, 4)$ und $= (3, 4)$ angenommen wurden. Die betrachtete Attributkombination hat also eine hohe Trennschärfe im Vergleichsraum, wobei jedoch zu beachten ist, daß für die Vergleichswerte $(1, 1)$ und $(1, 3)$ jeweils 20 Fälle mit $Same =$ 'No' auftraten, so daß eine Klassifikation, die ausschließlich auf dem Vergleich der Attribute *Street* und *Phone* basiert, fehlerbehaftet ist. Durch die Hinzunahme weiterer Vergleichs-Attribute kann aber die Trennschärfe im Vergleichsraum weiter erhöht werden.

Für die Adreßdaten geben wir drei Beispiele an, wobei wir für die ersten beiden Attribut-Kombinationen keine hohe Trennschärfe erreichen:

- *Sex* und *FirstNameEdit* (Abbildung 6.22): 94% der Paare mit $Same =$ 'Yes' findet man zwar bei dem Vergleichsvektor $(FirstNameEdit, Sex) = (1, 1)$, dieser Vergleichswert tritt aber auch für 42% der Paare mit $Same =$ 'No' auf. Das läßt sich damit begründen, daß die Übereinstimmung eines phonetischen Codes (Vor- oder Nachname) die Vorauswahl-Bedingung für die Paarbildung war.

- *Fullname* und *Birthdate* (Abbildung 6.23): Wir betrachten die Vergleiche von *FullnameEdit* und *Birthdate*. Es ergibt sich eine Konzentration von 99% der Paare mit $Same =$ 'No' auf die vier Werte mit $FullnameEdit \geq 5$ und $Birthdate \geq 2$, jedoch auch 4% der Paare mit $Same =$ 'Yes' haben diese Vergleichswerte (für 95%, d.h. für fast alle anderen Paare mit $Same =$ 'Yes' ist $FullnameEdit < 5$).

- *Fullname*, *Sex* und *Street* (Abbildung 6.24): Dies ist eine drei-dimensionale Kontingenztabelle, auf der y-Achse sind deshalb die Vergleichswert-Kombinationen der Attribute *Sex* und *Street* abgetragen. 98.5% aller Paare mit $Same$ = 'Yes' haben die Vergleichsvektoren ($Fullname$, $Sex, Street) = (i, 1, 1)$, $(1, j, k)$ oder $(3, j, k)$ (i, j und k sind hier beliebig Werte), während nur 0.5% der Paare mit$Same$ = 'No' diese Werte annehmen. Umgekehrt sind bei den Werten $(Fullname, Sex, Street) = (i, j, k)$, $i, j \geq 5$ 99% der Vergleichswerte der nicht-dubletten und weniger als 1% der dubletten Paare konzentriert. Die Kombination dieser drei Attribute hat also eine sehr hohe Trennschärfe.

Für die Bibliotheksdaten betrachten wir zwei Kombinationen

- *Address* und *Publisher* (Abbildung 6.25): Aus der Nicht-Übereinstimmung des Verlagsortes und Verlages läßt sich nicht die Verschiedenheit zweier Bücher ableiten, was aus der Abbildung 6.25 ersichtlich ist. Immerhin stimmt der Verlag bei etwa drei Vierteln aller dubletten Paare überein. Interessanterweise enthält die Stichprobe etwa 9% der nicht-dubletten Paare, bei denen *Address* oder *Publisher* übereinstimmen.

- *Authors* und *TitleMinEdit* (Abbildung 6.26): Die Kombination dieser beiden Attribute hat eine sehr hohe Trennschärfe für die Häufigkeitsverteilung der nicht-dubletten Paare, ihre Masse liegt zu 99.9% bei den Vergleichswerten, bei denen $Authors > 2$ oder $TitleMinEdit > 2$ gilt, jedoch findet man dort auch 12% der dubletten Paare. Es wird somit eine Klassifikation geben, die Paare, deren Vergleichswerte für $Authors \leq 2$ oder für $TitleMinEdit \leq 2$ liegen, fast fehlerfrei als Duplikate klassifizieren kann. Für die verbleibenden Fälle sind aber —um dort den Fehler zu minimieren— die Werte weiterer Vergleichs-Attribute mit heranzuziehen.

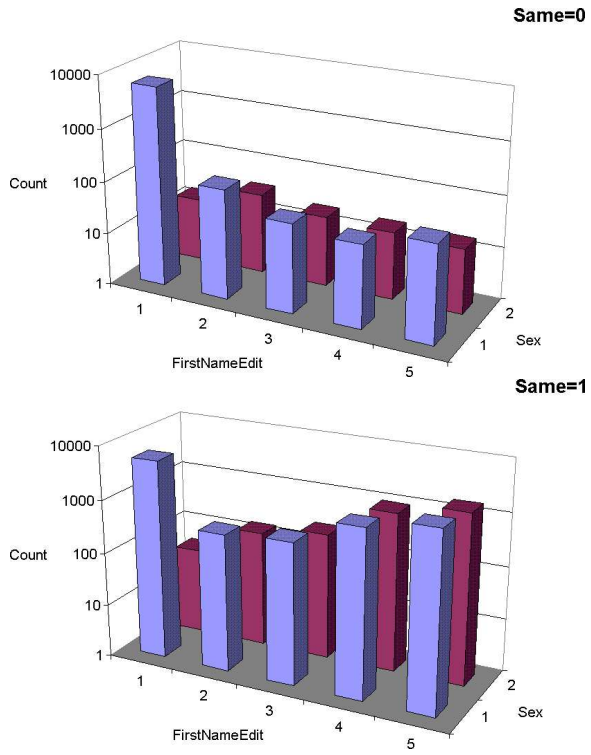


Abbildung 6.22: Häufigkeiten der Vergleichswert-Kombinationen der Attribute *Sex* und *FirstNameEdit* für die Adreßdaten (jeweils für Paare von Duplikaten $Same = 0$ und Nicht-Duplikaten $Same = 1$)

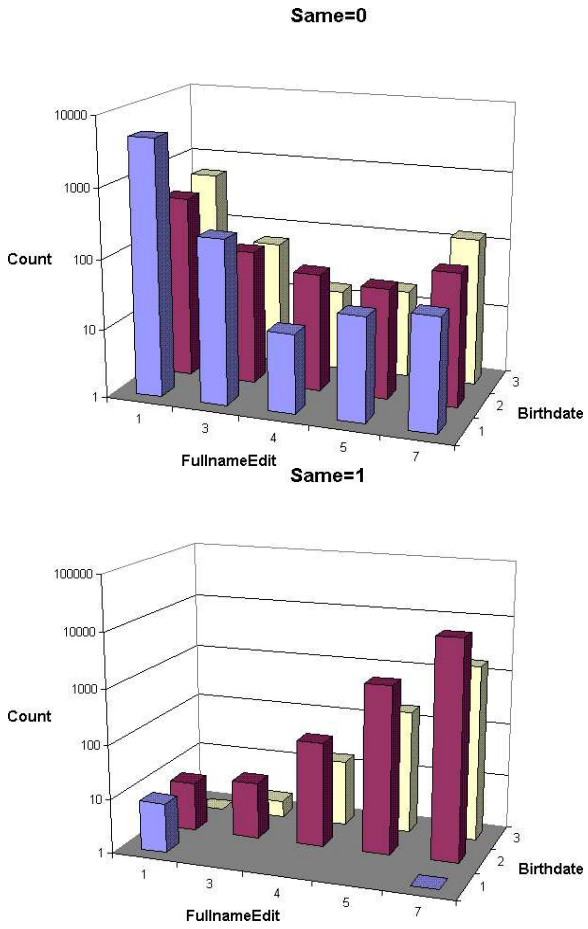


Abbildung 6.23: Häufigkeiten der Vergleichswert-Kombinationen der Attribute *FullnameEdit* und *Birthdate* für die Adreßdaten (jeweils für Paare von Duplikaten $Same = 0$ und Nicht-Duplikaten $Same = 1$)

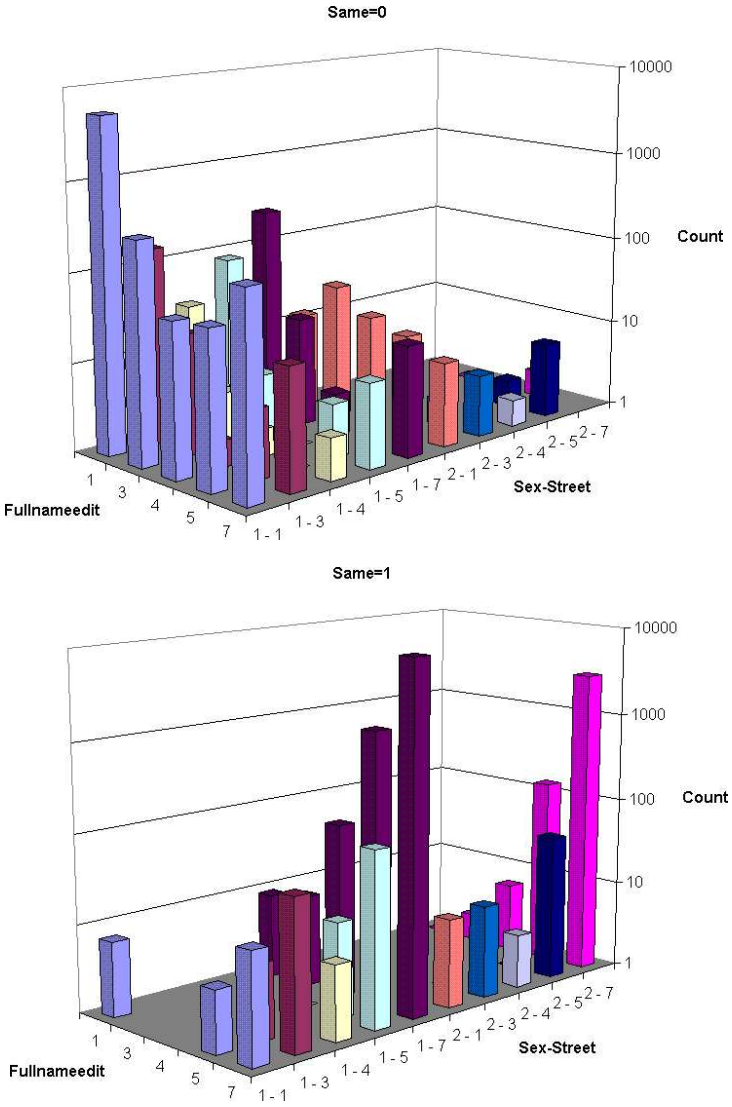


Abbildung 6.24: Häufigkeiten der Vergleichswert-Kombinationen der Attribute *FullnameEdit*, *Sex* und *Street* für die Adreßdaten (jeweils für Paare von Duplikaten $Same = 0$ und Nicht-Duplikaten $Same = 1$)

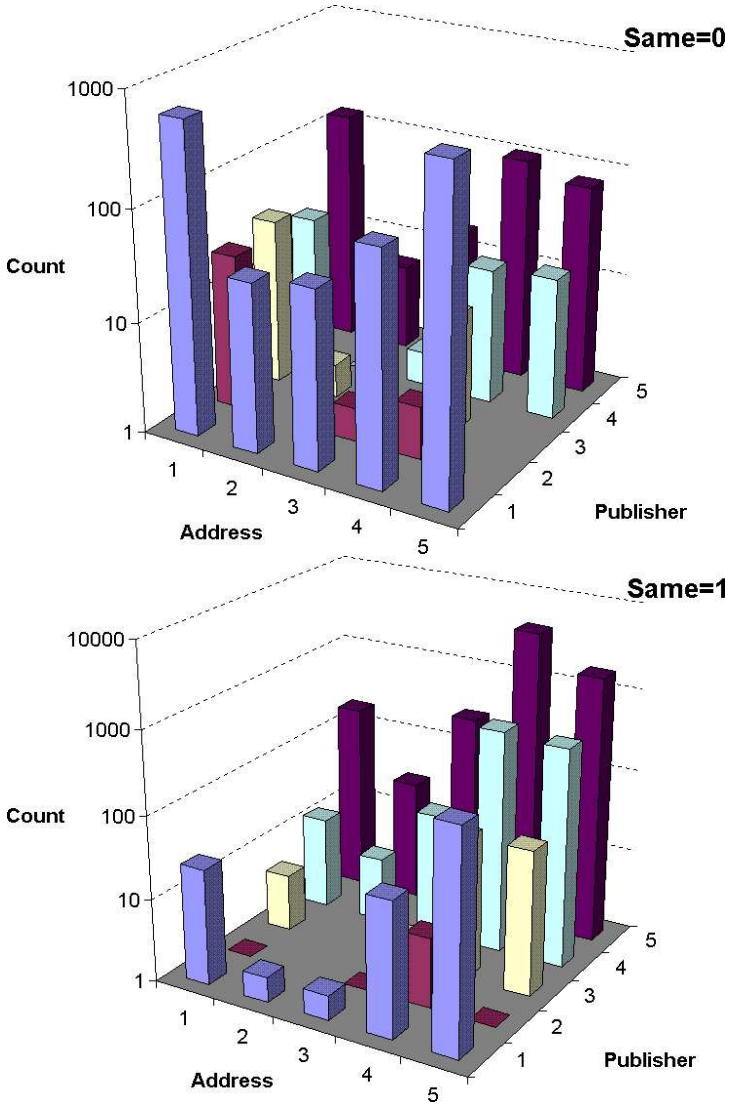


Abbildung 6.25: Häufigkeiten der Vergleichswert-Kombinationen der Attribute *Address* und *Publisher* für die Bibliotheksdaten (jeweils für Paare von Duplikaten $Same = 0$ und Nicht-Duplikaten $Same = 1$)

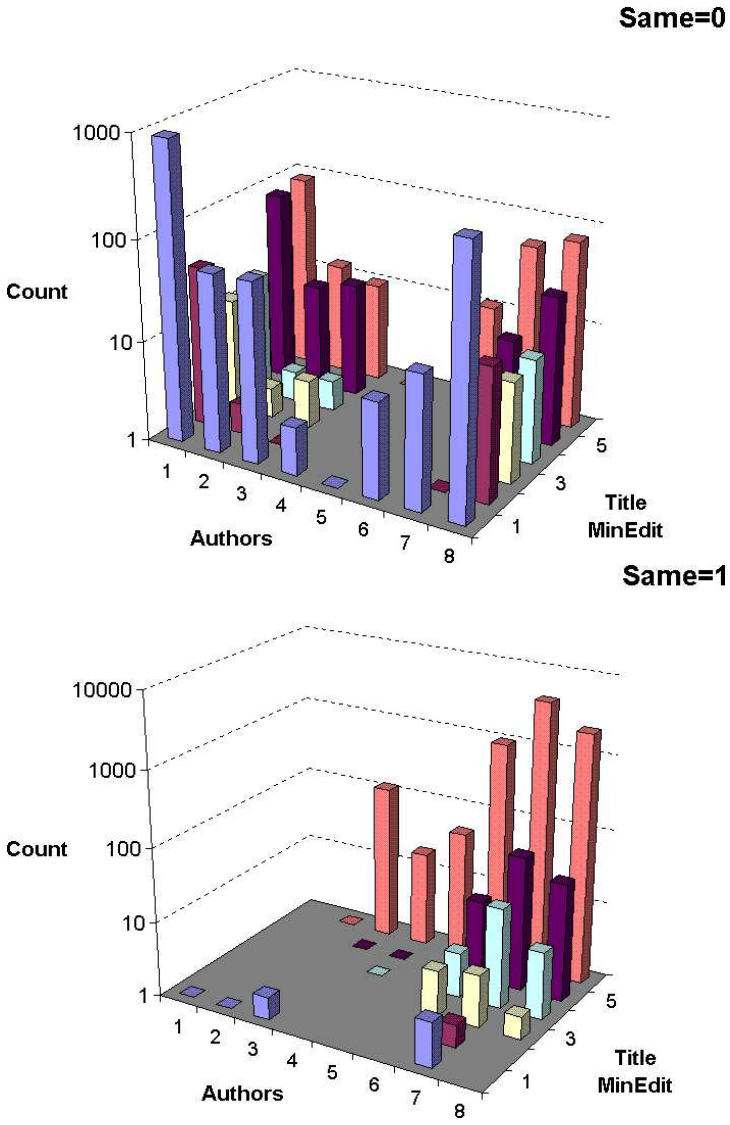


Abbildung 6.26: Häufigkeiten der Vergleichswert-Kombinationen der Attribute *Authors* und *TitleMinEdit* für die Bibliotheksdaten (jeweils für Paare von Duplikaten $Same = 0$ und Nicht-Duplikaten $Same = 1$)

Zusammenfassung und Ausblick

Objektidentifizierung ist notwendig, wenn die Daten von Realwelt-Objekten über mehrere Datenbanken verteilt sind. Die Objektidentifizierung wird erschwert, falls es keinen *globalen* und *konsistenten Identifizierer* in den Datenbanken gibt. Für viele Anwendungen tritt dieser Fall ein, beispielsweise bei der Integration von Informationen aus dem Internet, beim Data Warehousing oder bei einem registerbasierten Zensus. Sofern kein gemeinsamer Identifizierer gegeben ist, muß die Objektidentifizierung auf den identifizierenden Informationen basieren, die in den Daten eines Realwelt-Objektes selbst enthalten sind. Aufgrund fehlerhafter Daten schlagen Identifizierungs-Mechanismen wie *natürliche Schlüssel* oft fehl — Variationen und Fehler in den Daten müssen berücksichtigt werden. Folglich kann auch eine fehlerfreie Objektidentifizierung nicht mehr garantiert werden.

Wir entwickelten ein generisches formales Modell für die Objektidentifizierung, bei dem keine Annahmen über globale Schlüssel getroffen werden müssen. In diesem Modell wird Objektidentifizierung als ein spezielles Klassifikations-Problem aufgefaßt. Für die Klassifikation ist jedoch eine spezielle Vorverarbeitung (*engl.: preprocessing*) erforderlich. Das formale Modell besteht im Kern aus drei aufeinanderfolgenden Schritten:

1. *Conversion*: Ableitung (gemeinsamer) identifizierender Attribute aus den Datenbanken,
2. *Comparison*: Vergleich dieser Attribute für Paare von Elementen,
3. *Classification*: Klassifikation der Paare basierend auf den Vergleichswerten.

Wir erläutern die Verwendung mehrerer Klassifikationsmethoden, wie z.B. Entscheidungsbaum-Verfahren, die für Personendaten bewährte statistische Methode Record Linkage, sowie eine Generalisierung der Sorted Neighborhood Method. Die Wahl von Ableitungs- und Vergleichsfunktionen wird ausgiebig diskutiert.

Das formale Modell wird ergänzt um folgende Aspekte:

- Identifikationskonzepte in relationalen und objekt-orientierten Datenbanken sowie Constraints, die ihre Anwendbarkeit erlauben,

- Charakteristika für die Datenqualität, die die Eigenschaften eines vorliegenden Objektidentifizierungs-Problems determinieren sowie eine Formalisierung nutzbarer Metadaten wie z.B. semantischer Schlüssel und Differenzierungs-Schlüssel,

- Wahl einer optimalen Vorauswahl von Datensatzpaaren, so daß große Datenmengen effektiv verarbeitet werden können, wobei Indexstrukturen zum Einsatz kommen, sowie

- Ergänzung der Mediator-Wrapper Software-Architektur um einen spezialisierten Mediator, den sogenannten *Identificator*, um die Objektidentifizierung umzusetzen.

Ausgehend von unserem formalen Modell führten wir eine Evaluation von drei Methoden durch (Entscheidungsbaum-Verfahren, Record Linkage und AssoClass, eine auf Assoziationsregeln basierende Klassifikationsmethode). Wir verwendeten drei Datenbestände: Aus Online-Ausgaben von Berliner Tageszeitungen extrahierte Wohnungsanzeigen, einen Auszug aus einem Bibliothekskatalog sowie eine Kunden-Datenbank mit Adreßdaten. Wir geben Qualitätskriterien für Objektidentifizierungs-Verfahren an, beispielsweise Korrektheit. Wir zeigen, daß die Skalierbarkeit bezüglich der Datenbank-Größe ausschließlich von der verwandten Vorauswahl-Methode abhängt: Relationale Operatoren wie z.B. Gruppierung, haben Komplexität $O(n^2)$, während Vorauswahl-Methoden, die die Zahl der mit einem Element zu vergleichenden Elemente durch ein $k \ll n$ beschränken, linear skalieren, $O(k \cdot n)$. Die Testumgebung sowie die Durchführung der Tests einschließlich Stichprobenziehung ist beschrieben. Die Ergebnisse des Korrektheits-Tests sind überraschend: AssoClass und das bewährte Verfahren Record Linkage liefern deutlich schlechtere Ergebnisse als das Entscheidungsbaum-Verfahren. Das mag zwar in der für diese Methoden notwendigen Beschränkung auf wenige der verfügbaren Attribute für Record Linkage und AssoClass liegen, jedoch verhält sich das Entscheidungsbaum-Verfahren selbst bei Beschränkung auf diese Attribute noch etwas besser. Wir schließen daraus, daß, falls in vielen Attributen identifizierende Informationen enthalten sind, der Teile-und-Herrsche (*engl.: divide and conquer*) Ansatz, der dem Entscheidungsbaum-Verfahren immanent ist, die korrektesten Ergebnisse liefert. Wenngleich dieses Verfahren auch die beste Korrektheit erbrachte, kann bei Record Linkage und stärker noch bei AssoClass eine Kontrolle der beiden verschiedenen Arten der Falschklassifikationen vorgenommen werden.

Der Fortschritt in der Informationstechnik und die exponentiell ansteigenden Datenmengen in aller Welt erfordern Methoden, die eine Integration von Daten aus verschiedenen Quellen ermöglichen. Aufgrunddessen ist die Lösung der Objektidentifizierungs-Probleme in heutigen und zukünftigen Informationssystemen unabdingbar. Basierend auf dem hier vorgestellten Modell, können Objektidentifizierungs-Verfahren spezifiziert, umgesetzt und bewertet werden. Zudem ist es möglich das bestmögliche Verfahren zu finden, daß die Anforderungen einer speziellen Anwendung bezüglich Korrektheit, Performanz, Kosten oder weiterer Qualitätskriterien erfüllt.

Für die Lösung eines gegebenen Objektidentifizierungs-Problems schlagen wir das folgende Vorgehen vor:

1. *Analyse der Datenqualität*

   - Zusammentragen von Informationen über die Daten (Metadaten[22]) und über bisher umgesetzte Lösungen (möglicherweise gibt es überprüfte Beispieldaten),
   - Gewinnung (Ableitung) von Attributen für den Datenbestand, die in der Realwelt zum Identifizieren oder Unterscheiden von Objekten verwandt werden[23], vgl. Kapitel 2,
   - Ermittlung semantischer Constraints für diese Attribute,[24] vgl. Kapitel 4,

2. *Vorauswahl von Datensatz-Paaren*

   - Wahl eines Selektors σ, beispielsweise durch Kombination von (approximativen) Differenzierungsschlüsseln mit für die Anwendung hinreichend kleiner Fehlerrate $\mathrm{err}(\sigma)$ und Selektionsrate $\mathrm{sel}(\sigma)$, vgl. Kapitel 5,
   - Erneute Ermittlung (Aktualisierung) der semantischen Constraints unter Berücksichtigung der Vorauswahl σ,[25]

3. *Wahl des Klassifikationsmodells*, d.h. der Vergleichs-Attribute, der Vergleichsfunktionen (Definition des Vergleichsraumes) sowie des Klassifikationsverfahrens

   - Auswahl von Attributen, die einzeln oder im Verbund identifizerend sind, d.h. *approximative Schlüssel* mit Werten der (Δ)-confidence und anti-confidence nahe 1, sowie von Vergleichsfunktionen für sie,
   - Analyse der Verschiedenheit der Häufigkeits-Verteilungen von Vergleichsvektoren dubletter und nicht-dubletter Paare (z.B. mit Kontingenztabellen für diskrete Vergleichsfunktionen, vgl. Abschnitt 6.6 auf Seite 192),
   - Wahl der Klassifikationsmethode und der Modellparameter,[26]

[22] Metadaten sind z.B. Anzahl erwarteter Duplikate, Art der Datenentstehung, Erfahrungen bezüglich typischer Fehler

[23] Falls die Daten aus unterschiedlichen Quellen stammen, ist vor der Konversion eine partielle Einbettung ihrer Schemata in ein globales Schema notwendig

[24] Charakteristika: null, selectivity, accuracy, confidence, anti-confidence, sowie, falls Metriken *dist* für sie definiert werden können: Δ-accuracy, Δ-confidence und Δ-anti-confidence; weitere semantische Constraints: duplicates, overlap für Teilmengen der Daten und semantic-key, diff-key, approx-key, Δ-diff-key und approx-diff-key für Attribute.

[25] Die erneute Ermittlung der semantischen Constraints ist notwendig, da sich durch die Vorauswahl ihre Werte in der Regel ändern.

[26] Aufgrund der Ergebnisse des Benchmarkings wird das *Entscheidungsbaumverfahren* mit dem *Informationsgewinnverhältnis* als Maß ohne *Pruning* sowie ohne Beschränkung auf eine Attributauswahl empfohlen. Es sei bemerkt, daß die Anpassungsgüte eines gelernten Entscheidungsbaumes sich in der Regel nicht durch die mehrfache (redundante) Verwendung von Vergleichs-Attributen verschlechtert.

4. *Test/Anwendung von Klassifikationsmethoden*

   - Ziehung einer (oder mehrerer) Stichproben mit dubletten und nicht-dubletten Datensatz-Paaren (aus der Vorauswahl σ) zum Lernen und Testen von Klassifikationen (vgl. Abschnitt 6.2.2),
   - Berechnen der Vergleichswerte für alle Paare sowie Aufteilung in je eine Lern- und Test-Stichprobe,
   - Lernen einer (oder mehrerer) Klassifikationen,
   - Analyse der Ergebnisse,[27] und, sofern noch nicht zufriedenstellend, Veränderung von Modellparametern oder Modifikation der verwandten Vergleichs-Attribute/Vergleichsfunktionen und wiederholter Test,
   - Speicherung der gelernten Entscheidungsfunktion δ.

Damit erhalten wir eine Entscheidungsfunktion ($\delta \circ \sigma$), die zur Objektidentifizierung verwandt werden kann. Es werden nur diejenigen Paare von Datensätzen gebildet und verglichen und mit δ klassifiziert, die innerhalb der mit dem Selektor σ bestimmten Vorauswahl liegen. Alle anderen Paare werden von vornherein als nicht-dublett klassifiziert. Um die zusammengesetzte Entscheidungsfunktion ($\delta \circ \sigma$) auf große Datenbestände anwenden zu können, ist eine effiziente Umsetzung des für die Vorauswahl verwandten Selektors σ notwendig, beispielsweise durch die Verwendung von Indexstrukturen (vgl. Abschnitt 5.2 auf Seite 133).

Ausblick

Der in dieser Arbeit vorgestellte Ansatz ist einerseits eine Grundlage für die Entscheidungsunterstützung beim Design von anwendungsspezifischen Objektidentifizierungs-Verfahren und bildet andererseits eine Basis für weitergehende Forschungen. Als forschungsrelevante Themen für die Objektidentifizierung in multiplen Datenbanken betrachten wir:

- Untersuchung des Einflusses (empirischer) funktionaler Abhängigkeiten in den Daten auf die Fehlerraten verschiedener Klassifikationsmethoden,
- Die Anwendung von Identifizierungs-Ansätzen, bei denen Objekt-Referenzen verfolgt werden für praktische Anwendungen, sowie die Umsetzung der Objektidentifizierung für Datenbanken mit komplexen Schemata,
- Die Verwendung anderer Klassifikationsmethoden (z.B. Fuzzy-basierte Ansätze),
- Das Lernen von Vergleichsfunktionen mit hoher Trennschärfe aus Beispielstichproben dubletter und nicht-dubletter Paare,

[27] z.B. α- und β-Fehler, Überprüfen der falsch klassifizierten Paare (es liefert oft Hinweise für Modifikationen)

- Das Anwenden von Objektidentifizierungs-Ansätzen, um (1) Informationen aus Datenbanken für eine Population von Realwelt-Objekten (automatisch) zu integrieren (vgl. z.B. Neiling et al. [NSS03], Crescenzi et al. [CMM01] oder Raghavan et al. [RGM01]) oder um (2) partielle Einbettungen mehrerer Schemata in ein globales Schema zu gewinnen (Schema-Matching, vgl. Do et. al [DR02, DMR03]),

- Das Zusammenwirken der Objektidentifizierung im Prozeß der Datenbereinigung (*engl.: Data Cleansing*), vgl. z.B. Galhardas et al. [GFS+01b, GFS+01a], d.h. inwiefern eine verbesserte Bereinigung zur korrekteren Objektidentifizierung führt und wie die Ergebnisse einer Objektidentifizierung zur Bereinigung des Datenbestandes verwandt werden können (z.B. Auflösen von Wert-Konflikten).

Als bedeutsam für praktische Anwendungen halten wir die Festlegung von Domänen-spezifischen Benchmarks (z.B. für Adreß- oder Census-Daten). Die Notwendigkeit solch einer Benchmark wollen wir am Beispiel des registerbasierten Zensus erläutern.

Bemerkung 6.16 *(registerbasierter Zensus).* Seit mehreren Jahren gibt es Überlegungen und Anstrengungen in einigen Ländern, die kostenintensive Vollerhebung für einen Zensus (zumindest teilweise) durch die Verwendung der über die Bürger gesammelten Daten aus administrativen Registern zu ersetzen (vgl. Jensen [Jen83], Marquis et. al [MWP96], Carter et. al [CM96], McMillen [McM98]). Sofern ein globaler Identifizierer für Bürger in den Registern mitgeführt wird, kann dieser zur Verknüpfung der Daten verwandt werden, wie z.B. die Social Security Number in den USA.

Für Deutschland ist die nächste Volkszählung als registerbasierter Zensus geplant. D.h. es ist vorgesehen, Daten aus administrativen Registern dafür zu erheben (z.B. Melderegister, Daten der Bundesanstalt für Arbeit, der Sozialversicherung u.v.a.m.) Informationen zusammenzutragen und je Bürger zu verknüpfen um daraus eine Volkszählungsdatei zu bilden (die administrativen Daten sollen um stichprobenartig direkt von den Bürgern zu erhebende Daten ergänzt werden). Da in Deutschland aber kein globaler Identifizierer gegeben ist, muß ein Objektidentifizierungs-Verfahren für die Verknüpfung verwandt werden. Die Qualität der Volkszählungsdatei hängt offensichtlich nicht nur von der Qualität der administrativen Daten,[28] sondern auch von der Güte des verwandten Objektidentifizierungs-Verfahrens ab. Aufgrunddessen ist eine hohe Qualität nur durch gewissenhaftes Entwickeln und Testen auf Anwendungs-relevanten Daten möglich. Ein Test zur Verknüpfung von Melderegisterdaten mit Daten von Wohnungsinhabern aus der Gebäude- und Wohnungs-Zählung führte zu Fehlerraten unter 1%, vgl. Fürnrohr et. al [FRR02]. Hierbei wurde die Adresse als Differenzierungsschlüssel verwandt, d.h. nur Datensätze mit gleicher Adresse wurden miteinander verglichen und der beste Treffer bezüglich eines mehrstufigen (hierarchischen) Vergleichs, der einen Mindestwert überschritt, wurde als Duplikat klassifiziert.

[28] Die Qualität der Melderegister-Daten wird beispielsweise von Below [Bel94] untersucht

Ich danke den Professoren Prof. Dr. Hans-Joachim Lenz und Prof. Dr. Bernhard Thalheim herzlichst für die intensive Betreuung und Unterstützung während des Entwicklungsprozesses dieser Arbeit.

Literaturverzeichnis

[ABB92] Peter Adman, Stephen W. Baskerville, and Katherine F. Beedham. Computer-assisted record linkage: Or how best to optimize links without generating errors. *History and Computing*, 4:2–15, 1992.

[Abr74] J. R. Abrial. Data semantics. In J. W. Klimbie and K. L. Koffeman, editors, *Data Base Management. Proceedings of the IFIP Working Conference on Data Base Management, Corsica, France*, pages 1–60. NorthHolland, 1974.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

[AJ97] Wendy Alvey and Bettye Jamerson, editors. *Record Linkage Techniques — 1997. Proceedings of an International Workshop and Exposition. March 20-21, 1997 in Arlington, Virginia*, Washington, DC, 1997. Federal Committee on Statistical Methodology, Office of Management and Budget.

[AK89] S. Abiteboul and P. Kanellakis. Object identity as a query language primitive. In *ACM SIGMOD International Conf. on Management of Data*, pages 159–173, 1989.

[AKA91] D. W. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[AM93] J.B. Armstrong and J.E. Mayda. Model-based estimation of record linkage error rates. *Survey Methodology, A Journal of Statistics Canada*, 19(2):137–147, 1993.

[Bam72] Günter Bamberg. *Statistische Entscheidungstheorie*. Physica-Verlag, Würzburg-Wien, 1972.

[BBKK01] Christian Böhm, Bernhard Braunmuller, Florian Krebs, and Hans-Peter Kriegel. Epsilon grid order: An algorithm for the similarity join on massive high-dimensional data. In *ACM SIGMOD Conference 2001: Santa Barbara, CA, USA*, pages 379–388, 2001.

[BBKM00] C. Böhm, S. Berchtold, H.-P. Kriegel, and U. Michel. Multidimensional index structures in relational databases. *Journal for Intelligent Information Systems*, 15:51–70, 2000.

[BD83] Dina Bitton and David J. DeWitt. Duplicate record elimination in large data files. *ACM Transactions on Database Systems*, 8(2):255–265, 1983.

[Bel93] Thomas R. Belin. Evaluation of sources of variation in record linkage through a factorial experiment. *Survey Methodology, A Journal of Statistics Canada*, 19(1):13–29, 1993.

[Bel94] Sigrun Below. Qualität der Einwohnermelderegister und ihre statistische Nutzung — Eine Literaturstudie. *Berliner Statistik*, (6):106–123, 1994.

[BFH75] Yvonne M. M. Bishop, Stephen E. Fienberg, and Paul W. Holland. *Discrete Multivariate Analysis. Theory and Practice*. MIT Press, Cambridge, 1975.

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression trees*. Chapman & Hall, 1984.

[Bis83] J. Biskup. A foundation of codd's relational maybe-operations. *ACM Trans. Database Systems*, 8(4):608–636, 1983.

[BK98] Christian Borgelt and Rudolf Kruse. Attributauswahlmaße für die Induktion von Entscheidungsbäumen: Ein Überblick. In Gholamreza Nakheizadeh, editor, *Data Mining: Theorie und Anwendungen*, pages 77–98. Physica-Verlag, Heidelberg, Germany, 1998.

[BK02] Christian Borgelt and Rudolf Kruse. Induction of association rules: Apriori implementation. In *Proc. 15th Conf. on Computational Statistics (Compstat 2002, Berlin, Germany)*, Heidelberg, Germany, 2002. Physika Verlag.

[BKK96] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An index structure for high-dimensional data. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *Proceedings of the 22nd International Conference on Very Large Databases*, pages 28–39, San Francisco, U.S.A., 1996. Morgan Kaufmann Publishers.

[BKWZ93] S. Bartlett, D. Krewski, Y. Wang, and J.M. Zielinski. Evaluation of error rates in large scale computerized record linkage studies. *Survey Methodology, A Journal of Statistics Canada*, 19(1):3–12, 1993.

[BM03] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2003).*, 2003.

[BO99] Tolga Bozkaya and Z. Meral Özsoyoglu. Indexing large metric spaces for similarity search queries. *TODS*, 24(3):361–404, 1999.

[Bor03a] Christian Borgelt. Software for association rule induction (version 4.08, 2003.03.12)., 2003. available from `http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html`.

[Bor03b] Christian Borgelt. Software for decision and regression tree induction (version 3.12, 2003.02.02), 2003. available from `http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html`.

[BR95] Thomas R. Belin and Donald B. Rubin. A method for calibrating false-match rates in record linkage. *Journal of the American Statistical Association*, 90(430):694–707, 1995.

[BS01] Glenn B. Bell and Anil Sethi. Matching records in a national medical patient index. *Communications of the ACM*, 44(9):83–88, September 2001.

[BT99] C. Beeri and B. Thalheim. Identification as a primitive of data models. In T. Ripke T. Polle and K.-D. Schewe, editors, *Fundamentals of Information Systems*, pages 19–36. Kluwer, 1999.

[Bur88] R. D. Burgess. Evaluation of reverse record check estimates of undercoverage in the canadian census of population. *Survey Methodology*, 14(2):137–156, 1988.

[CCLZ02] T. Churches, P. Christen, J. Lu, and J.X. Zhu. Preparation of name and address data for record linkage using hidden markov models. *BioMed Central Medical Informatics and Decision Making*, 2(9), 2002.

[CCZ02] P. Christen, T. Churches, and J.X. Zhu. Probabilistic name and address cleaning and standardization. In *The Australian Data Mining Workshop, 2002*, November 2002.

[CDEV] M. Cochinwala, S. Dalal, A.K. Elmagarmid, and V.S. Verykios. Record matching: Past, present and future. Submitted to ACM Computing Surveys.

[CGS97] K. Selcuk Candan, John Grant, and V. S. Subrahmanian. A unified treatment of null values using constraints. *Information Sciences*, 98(1-4):99–156, 1997.

[CH90] J.B. Copas and F.J. Hilton. Record linkage: Statistical models for matching computer models. *Journal of the Royal Statistical Society, Series A*, 153:287–320, 1990.

[Chr90] Ronald R. Christensen. *Log-Linear Models.* Texts in Statistics. Springer, New York; Berlin; Heidelberg et al., 1990.

[CM96] Richard Carter and Kathy McClean. Using administrative data in the Canadian Census: experiences and plans. *Statistical Journal of the United Nations Economic Commission for Europe (ECE)*, 13:375–383, 1996.

[CMM01] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01), Orlando, 2001*, pages 109–118, 2001.

[Cod86] E. F. Codd. Missing information (applicable and inapplicable) in relational databases. *SIGMOD Record*, 15(4):53–78, 1986.

[Coh98] William W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of the 1998 ACM SIGMOD Conf.*, Seattle, Washington, 1998.

[Cou83] B. Courcelle. Fundamental properties of infite trees. *Theoretical Computer Science*, 25:95–169, 1983.

[CPZ97] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 426–435. Morgan Kaufmann, 1997.

[CRF03a] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the ACM Workshop on Data Cleaning, Record Linkage and Object Identification, Washington DC, August 2003.*, 2003.

[CRF03b] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string metrics for matching names and addresses. In *International Joint Conference on Artificial Intelligence, Proceedings of the Workshop on Information Integration on the Web, Acapulco, Mexico, August 2003*, 2003.

[CS92] Abhirup Chatterjee and Arie Segev. Resolving data heterogeneity in scientific and statistical databases. In Hans Hinterberger and James C. French, editors, *Proceedings of the Sixth International Working Conference on Scientific and Statistical Database Management*. Eidgenössische Technische Hochschule Zürich, Departement Informatik, Institut für Wissenschaftliches Rechnen, 6 1992.

[DeG88] Yves DeGuire. Postal address analysis. *Survey Methodology, A Journal of Statistics Canada*, 14(2):317–325, 1988.

[DLN02a] Daniel Delic, Hans-Joachim Lenz, and Mattis Neiling. Improving the quality of association rule mining by means of rough sets. In *Proceedings of the Workshop SMPS 2002, Warsaw, September 9-11, 2002*, 2002.

[DLN02b] Daniel Delic, Hans-Joachim Lenz, and Mattis Neiling. Rough sets and association rules - which is efficient? In *Proc. 15th Conf. on Computational Statistics (Compstat 2002, Berlin, Germany)*, Heidelberg, Germany, 2002. Physika Verlag.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[DLS80] J. N. Darroch, Steffen L. Lauritzen, and T. P. Speed. Markov fields and log-linear interaction models for contingency tables. *The Annals of Statistics*, 8(3):522–539, 1980.

[DMR03] H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In A. Chaudhri, M. Jeckle, E. Rahm, and R. Unland, editors, *Web, Web-Services, and Database Systems. NODe 2002 Web and Database-Related Workshops, Erfurt, Germany, October 7–10, 2002, Revised Papers*, number 2593 in LNCS, pages 219–226, Berlin, New York, 2003. Springer–Verlag.

[DR02] H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *Proceedings of the Conference VLDB '02, 2002.*, 2002.

[Efr79] B. Efron. Computers and the theory of statistics. *SIAM Review*, 21:460–480, 1979.

[EVE02] Mohamed G. Elfeky, Vassilios S. Verykios, and Ahmed K. Elmagarmid. Tailor: A record linkage toolbox. In *Proceedings of the 18th International Conference on Data Engineering (ICDE02), San Jose, California, USA*, 2002.

[Fai96] Martha F. Fair. Record linkage in an information age society. In *Proceedings of the Census Bureau's Conference and Technology Interchange, March 17-21, 1996 (Annual Research Conference)*, Washington, DC, 1996. U.S. Bureau of the Census.

[FC00] Amy Feekin and Zhengxin Chen. Duplicate detection using k-way sorting method. In *Proc. of SAC'00, March 19-21, Como, Italy*, pages 323–327, 2000.

[Fee99] A. J. Feelders. Handling missing data in trees: Surrogate splits or statistical imputation. In *Principles of Data Mining and Knowledge Discovery*, pages 329–334, 1999.

[Fre81] D.A. Freedman. Bootstrapping regression models. *SIAM Review*, 9:1218–1228, 1981.

[FRR02] M. Fürnrohr, B. Rimmelspacher, and T.v. Roncador. Zusammenführung von Datenbeständen ohne numerische Identifikatoren - ein Verfahren im Rahmen der Testuntersuchungen zu einem registergestützen Zensus. *Bayern in Zahlen*, (7):308–321, Juli 2002.

[FS69] Ivan P. Fellegi and Alan B. Sunter. A theory of record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[Gal01] Helena Galhardas. *Data Cleaning: Model, Language and Algorithms.* Phd thesis, University of Versailles, September 2001.

[GBVR03] Lifang Gu, Rohan Baxter, Deanne Vickers, and Chris Rainsford. Record linkage: Current practice and future directions. Technical Report 03/83, CSIRO Mathematical and Information Sciences, GPO Box 664, Canberra 2601, Australia, April 2003.

[GFS+01a] Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, and Cristian-Augustin Saita. Improving data cleaning quality using a data lineage facility. In *Proc. of the Int. Workshop on Design and Management of Data Warehouses, Interlaken, Switzerland*, 2001.

[GFS+01b] Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, and Cristian-Augustin Saita. Declarative data cleaning: Language, model and algorithms. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01), Orlando, 2001*, 2001.

[GFSS00] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. Ajax: An extensible data cleaning tool. In *Proc. of SIGMOD 2000*, 2000. demo paper.

[GG98] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[Gil99] L. Gill. OX-LINK: The oxford medical record linkage system. In Alvey and Jamerson [AJ97], pages 15–33.

[Gra91] G. Grahne. *The Problem of Incomplete Information in Relational Databases.* Springer-Verlag, Berlin, 1991.

[Gra93] Jim Gray, editor. *The Benchmark Handbook for Database and Transaction Systems (2nd Edition).* Morgan Kaufmann, 1993.

[GRL98] GRLS: generalized record linkage system/Canlink, 1998.

[Gus97] D. Gusfield. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, New York, 1997.

[GZ88] G. Gottlob and R. Zicari. Closed world databases opened through null values. In *14th VLDB 1988, Los Angeles, CA*, 1988.

[Har86] David Harel. Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness. *Journal of the ACM*, 33:224 – 248, 1986.

[Her96] Mauricio Antonio Hernandez. *A Generalization of Band Joins and The Merge/Purge Problem*. Phd thesis, Columbia University, 1996.

[HS95] Mauricio A. Hernandez and Salvatore J. Stolfo. The merge/purge problem for large databases. In *ACM SIGMOD Conference 1995*, pages 127–138, 1995.

[HS98] Mauricio A. Hernandez and Salvatore J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1):9–37, 1998.

[IL84] T. Imielinski and W. Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31:761–791, 1984.

[Jar89] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[Jen83] Poul Jensen. Toward to a register-based statistical system — some Danish experiences. *Statistical Journal of the United Nations Economic Commission for Europe (ECE)*, pages 341–365, 1983.

[Joh97] Sandra Johnson. Technical issues related to the probalistic linkage of population-based crash and injury data. In Alvey and Jamerson [AJ97], pages 222–226.

[JR99] Vanja Josifovski and Tore Risch. Integrating heterogenous overlapping databases through object-oriented transformations. In Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik, and Michael L. Brodie, editors, *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 435–446. Morgan Kaufmann, 1999.

[KA85] Beth Kilss and Wendy Alvey, editors. *Record Linkage Techniques — 1985. Proceedings of the Workshop on Exakt Matching Methodologies in Arlington, Virginia May 9–10, 1985*, Internal Revenue Service Publication, Washington, DC, 1985. Department of the Treasury, Statistics of Income Division.

[KAA+92] William Kent, Rafi Ahmed, Joseph Albert, Mohammed Ketabchi, and Ming-Chien Shan. Object identification in multi-database systems. Tech Report HPL-92-68, Hewlett-Packard Laboratories, Palo Alto, May 22 1992.

[KC86] S. N. Khoshafian and G. P. Copeland. Object identity. In N. K. Meyrowitz, editor, *Proceedings OOPSLA 1986, Annual Conference on ObjectOriented Programming Systems, Languages, and Applications, Portland, Oregon, USA*, pages 406–416, 1986. ACM SIGPLAN Notices, 21(11).

[Ken78] William Kent. *Data and Reality*. North-Holland, 1978.

[Ken91a] William Kent. Breakdown of the information model in multi-database systems. *SIGMOD Record*, 20(4):10–15, December 1991.

[Ken91b] William Kent. Solving domain mismatch and schema mismatch problems with an object-oriented database programming language. In Guy M. Lohman, Amílcar Sernadas, and Rafael Camps, editors, *17th International Conference on Very Large Data Bases, September 3-6, 1991, Barcelona, Catalonia, Spain, Proceedings*, pages 147–160. Morgan Kaufmann, 1991.

[Kle94] H.-J. Klein. How to modify SQL queries in order to guarantee sure answers. *ACM SIGMOD RECORD*, 23(3):14–20, 1994.

[Kle97] Hans-Joachim Klein. *Gesicherte und mögliche Antworten auf Anfragen an relationale Datenbanken mit partiellen Relationen*. Habilitation thesis, Institut für Informatik und Praktische Mathematik, ChristianAlbrechtsUniversität Kiel, 1997.

[Kle02] H.-J. Klein. Null values in relational databases and sure information answers. In *Semantics in Databases*, number LNCS 2582, pages 102–121. Springer-Verlag, 2002.

[KPPS03] Hans-Peter Kriegel, Martin Pfeifle, Marco Pötke, and Thomas Seidl. The paradigm of relational indexing: A survey. In Gerhard Weikum, Harald

Schöning, and Erhard Rahm, editors, *Tagungsband der 10. Tagung Datenbanksysteme für Business, Technologie und Web BTW'03, Leipzig, Germany, February 26 - 28, 2003*, number P-26 in GI-Edition - Lecture Notes in Informatics (LNI), Bonn, 2003. Köllen Verlag.

[KS96] Vipul Kashyap and Amit Sheth. Semantic and schematic similarities between database objects: a context-based approach. *The VLDB Journal*, 5:276–304, 1996.

[Lau96] S.L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[Leh94] Erich Leo Lehmann. *Testing Statistical Hypothesis*. Chapman & Hall, New York, 2nd edition, 1994. reprint of the 2nd edition from 1986.

[Lei60] Gottfried Wilhelm Leibniz. *Fragmente zur Logik*. Akademie-Verlag, Berlin, 1960. herausgegeben von Franz Schmidt.

[Len98] Hans-Joachim Lenz. Multi-data sources and data fusion. In *Proceedings of NTTS Sorrento 1998*, November 1998.

[Lip79] W. Lipski. On semantic issues connected with incomplete information databases. *ACM Trans. Database Systems*, 4(3):262–296, 1979.

[Lip81] W. Lipski. On databases with incomplete information. *Journal of the ACM*, 28:41–70, 1981.

[LL95] M. Levene and G. Loizou. A correspondence between variable relations and three-valued propositional logic. *International Journal of Computer Mathematics*, 55:29–38, 1995.

[LL99] Mark Levene and George Loizou. Database design for incomplete relations. *ACM Transactions on Database Systems*, 24(1):80–126, 1999.

[LN98] Hans-Joachim Lenz and Mattis Neiling. Zur Summierbarkeit von Häufigkeiten im data cube (in multi-dimensionalen Tabellen). discussion paper 1998/28, Fachbereich Wirtschaftswissenschaft der Freien Universität Berlin, 1998.

[LNE89] J.A. Larson, S.B. Navathe, and R. Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.

[LR87] R.J.A. Little and D.B. Rubin. *Statistical Analysis with Missing Data*. Wiley, New York, 1987.

[LR94] Chuanhai Liu and Donald B. Rubin. The ECME algorithm: A simple extension of EM and ECM with faster monotone convergence. *Biometrika*, 81(4):633–48, 1994.

[LR01] M. D. Larsen and D. B. Rubin. Iterative automated record linkage using mixture models. *Journal of the American Statistical Association*, 79:32–41, 2001.

[LSPR93] E.-P. Lim, J. Srivastava, S. Prabhakar, and J. Richardson. Entity identification in database integration. In A. Elmagarmid and E. Neuhold, editors, *Proceedings of the 9th IEEE Int'l Conf. on Data Engineering*, pages 294–301. IEEE Computer Society Press, 1993.

[LSPR96] Ee-Peng Lim, Jaideep Srivastava, Satya Prabhakar, and James Richardson. Entity identification in database integration. *Information Sciences*, 89(1):1–38, 1996.

[LSS96] E.-P. Lim, J. Srivastava, and S. Shekhar. An evidentional reasoning approach to attribute value conflict resolution in database integration. *IEEE Transaction Knowledge and Data Engineering*, 8(5):707–723, 1996.

[Mah36] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. In *Proceedings of the National Institute of Sciences of India*, pages 49–55, Calcutta, 1936.

[McM98] David B. McMillen. The census under attack. *CHANCE*, 11(2):50–55, 1998.

[ME96] Alvaro E. Monge and Charles Elkan. The field matching problem: Algorithms and applications. In *Knowledge Discovery and Data Mining*, pages 267–270, 1996.

[MGB74] Alexander McFarlane Mood, Franklin A. Graybill, and Duane C. Boes. *Introduction to the Theory of Statistics*. McGraw-Hill series in probability and statistics. McGraw-Hill, Tokyo, 1974.

[MGMR02] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proc. 18th ICDE Conference*, 2002.

[Mon97] Alvaro Edmundo Monge. *Adaptive detection of approximately duplicate database records and the database integration approach to information discovery*. Phd thesis, University of California, San Diego, 1997.

[MR93] Xiao-Li Meng and Donald B. Rubin. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2):267–78, 1993.

[MST94] Donald Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine learning, neural and statistical classification*. Horwood, New York, 1994.

[MW90] T. Matsushima and G. Wiederhold. A model of object-identities and values. Technical Report STAN-CS-90-1304, Dep. of Computer Science, Stanford University, California, USA, 1990.

[MWP96] Kent H. Marquis, Signe Wetrogan, and Henry Palacios. Towards a U.S. population database from administrative records. Working Papers in Survey Methodology SM-96/06, U.S. Bureau of the Census, Washington, DC, 1996.

[Nei98] Mattis Neiling. Data Fusion with Record Linkage. In *3. Workshop Föderierte Datenbanken, Magdeburg, 1998*, 1998. c.f. `http://wwwiti.cs.uni-magdeburg.de/fdb98/online-proc/`.

[Nei99] Mattis Neiling. Datenintegration durch Objekt-Identifikation. In Ralf-Detlef Kutsche, Ulf Leser, and Johann Christoph Freytag, editors, *4. Workshop Föderierte Datenbanken, Berlin, Germany, 25.-26. November 1999*, pages 117–143. TU Berlin, FB 13, 1999. CEUR-WS/Vol-25.

[New67] H. B. Newcombe. Record linking: The design of efficient systems for linking records into individual and family histories. *American Journal of Human Genetics*, 19, 1967.

[New88] Howard B. Newcombe. *Handbook of Record Linkage*. Oxford University Press, Oxford, 1988.

[NFL92] Howard B. Newcombe, Martha F. Fair, and Pierre Lalonde. The use of names for linking personal records. *Journal of the American Statistical Association*, 87:1193–1208, 1992.

[NJ03] Mattis Neiling and Steffen Jurk. The object identification framework. In *Proceedings of the ACM Workshop on Data Cleaning, Record Linkage and Object Identification, Washington DC, August 2003 (co-located with the ACM KDD 2003 Conf.)*, 2003.

[NJLN03] Mattis Neiling, Steffen Jurk, Hans-J. Lenz, and Felix Naumann. Object identification quality. In *Procs. of the Intl. Workshop on Data Quality in Cooperative Information Systems (DQCIS2003), Siena, Italy, January 10–11, 2003*, 2003.

[NKAJ59] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[NL00a] Mattis Neiling and Hans-Joachim Lenz. Data fusion and object identification. In *Proceedings of the Intl. Conf. on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet. l'Aquila, Italy, July 31 – August 6, 2000 (SSGRR 2000)*, 2000.

[NL00b] Mattis Neiling and Hans-Joachim Lenz. Data integration by means of object identification in information systems. In Hans Robert Hansen, Martin Bichler, and Harald Mahrer, editors, *Proceedings of the 8th European Conference on Information Systems (ECIS 2000), Vienna, Austria, July 2000*, volume 1, pages 364–371, New York, 2000. IEEE.

[NL02] Mattis Neiling and Hans-Joachim Lenz. Supplement of information: Data integration by classification of pairs of records. In *Classification, Automation, and New Media. Proceedings of the 24th Annual Conference of the Gesellschaft für Klassifikation e.V., University of Passau, March 15–17, 2000*, pages 252–258, Berlin, New York, 2002. Springer–Verlag.

[NM01] Mattis Neiling and Roland Müller. The good into the pot, the bad into the crop. Preselection of record pairs for database integration. In *Proceedings of the 1st Workshop DBFusion 2001, Gommern, May 3-4, 2001*, 2001.

[NSS03] Mattis Neiling, Markus Schaal, and Martin Schumann. Wrapit: Automated integration of web databases with extensional overlaps. In A. Chaudhri, M. Jeckle, E. Rahm, and R. Unland, editors, *Web, Web-Services, and Database Systems. NODe 2002 Web and Database-Related Workshops, Erfurt, Germany, October 7–10, 2002, Revised Papers*, number 2593 in LNCS, pages 219–226, Berlin, New York, 2003. Springer–Verlag.

[Nyg92] Lars Nygaard. Name standardisation in record linking: An improved algorithmic strategy. *History and Computing*, 4(2):63–74, 1992.

[O'N94] Peter O'Neill. *Database Management*. Morgan-Kaufmann, 1994.

[PBGG89] Jan Paredaens, Paul De Bra, Marc Gyssens, and Dirk Van Gucht. *The Structure of the Relational Database Model*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1989.

[Pos69] H.J. Postel. Die Kölner Phonetik - ein Verfahren zur Identifizierung von Personennamen auf Grundlage der Gestaltanalyse. *IBM-Nachrichten*, 19:925–931, 1969.

[Qui93] J. R. Quinlan. *Q4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[Ras98] Jochen Rasch. *On Value-Based Identification in ObjectOriented Data Models*. PhD thesis, Institut für Informatik und Praktische Mathematik, Christian--Albrechts-Universität Kiel, Kiel, Dezember 1998.

[RBC03] Peter Christen Rohan Baxter and Tim Churches. A comparison of fast blocking methods for record linkage. In *Proceedings of the Workshop on Data Cleaning, Record Linkage and Object Consolidation at the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington DC, August 2003*, 2003.

[Rec89] Christiane Recke. *Theorie und Anwendung log-lineare Modelle*. PhD thesis, Fachbereich Wirtschaftswissenschaften, Freie Universität Berlin, 1989.

[RGM01] Sriram Raghavan and Hector Garcia-Molina. Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01), Orlando, 2001*, pages 129–138, 2001.

[Rob81] J. Robinson. The K-D-B-tree: A search structure for large multidimensional dynamic indexes. In *In Proc. ACM PODS'81*, pages 10–18, 1981.

[Sal89] Gerald Salton, editor. *Automatic Text Processing*. Addison-Wesley, Reading, Massachusetts, 1989.

[SC99] Jaideep Srivastava and Ping-Yao Chen. Warehouse creation — a potential roadblock to data warehousing. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):118–126, 1999.

[Sch96] Jürgen Schürmann. *Pattern Classification*. John Wiley & Sons, New York, 1996.

[Sch97] Klaus-Dieter Schewe. Fundamentals of object oriented database modelling. *Intelligent Systems*, 1997.

[Sch01] Klaus-Dieter Schewe. On the unification of query algebras and their extension to rational tree structures. In *Proceedings of the 12th Australasian conference on Database technologies*, pages 52–59. IEEE Computer Society Press, 2001.

[SDLL92] L. Swain, J. D. Drew, B. Lafrance, and K. Lance. The creation of a residential address register for coverage improvement in the 1991 canadian census. *Survey Methodology*, 18(1):127–141, 1992.

[Sha48] C.E. Shannon. The mathematical theory of communication. *The Bell Systems Technical Journal*, 27:379–423, 1948.

[SPS] SPSS: The software package for statistics. Version 7.0.

[SSS01] E. Schallehn, K. Sattler, and G. Saake. Extensible grouping and aggregation for data reconciliation. In *Proc. 4th Int. Workshop on Engineering Federated Information Systems, EFIS'01, Berlin, Germany*, 2001.

[ST93] Klaus-Dieter Schewe and Bernhard Thalheim. Fundamental concepts of object oriented databases. *Acta Cybernetica*, 11(1–2):49–84, 1993.

[Stü03] Werner Stützle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *Journal of Classification*, 2003. To appear.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[SW93] Fritz Scheuren and William E. Winkler. Regression analysis of data files that are computer matched. *Survey Methodology, A Journal of Statistics Canada*, 19(1):39–58, 1993.

[Tho90] Wolfgang Thomas. *Automata on infinite objects*, pages 133–191. Elsevier and The MIT Press, 1990.

[TKM01] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26(8), 2001.

[TPC] The TPC benchmarks. c.f. `http://www.tpc.org`.

[Ull88] J. D. Ullmann. *Principle of Database and Knowledge-Management Systems. Vol. I: Classical Database Systems.* Computer Science Press, Rockville, USA, 1988.

[Vap98] V. N. Vapnik. *Statistical Learning Theory.* Wiley & Sons, 1998.

[Vas79] Y. Vassiliou. Null values in database management, a denotational semantics approach. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 162–169, 1979.

[Vas81] Y. Vassiliou. Functional dependencies and incomplete information. In *Proceedings of the Conference on Very Large Databases*, pages 260–269, 1981.

[VEE+00] V.S. Verykios, A.K. Elmagarmid, M. Elfeky, M. Cochinwala, and S. Dalal. On the completeness and accuracy of the record matching process. In *Proceedings of the MIT Conference on Information Quality 2000*, pages 54–69, Boston, MA, 2000.

[VEH00] V.S. Verykios, A.K. Elmagarmid, and E.N Houstis. Automating the approximate record matching process. *Journal of Information Sciences*, 126:83–98, 2000.

[Ver96] Jeroen K. Vermunt. Log-linear event history analysis: a general approach with missing data, unobserved heterogeneity, and latent variables. Technical report, Tilburg University, 1996.

[Ver97] Jeroen K. Vermunt. LEM: log-linear and event history analysis with missing data. Tilburg University, The Netherlands, 1997. software available from `http://www.kub.nl/faculteiten/fsw/organisatie/ /departementen/mto/software2.html`.

[VME02] V. Verykios, G. Moustakides, and M. Elfeky. A bayesian decision model for cost optimal record matching. *The VLDB Journal*, 2002.

[Whi90] Joe Whittaker. *Graphical Models in Applied Multivariate Analysis.* John Wiley & Sons, Chichester; New York et al., 1990.

[Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[Win89] W. Winkler. Near automatic weight computation in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methodology*, pages 667–671. American Statistical Association, 1989.

[Win90] W. Winkler. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In *Proceedings of the Section on Survey Research Methodology*, pages 354–359. American Statistical Association, 1990.

[Win93] William E. Winkler. Improved decision rules in the Fellegi-Sunter model of record linkage. The Research Report Series RR-93/12, U.S. Bureau of the Census, 1993.

[Win95] William E. Winkler. Matching and record linkage. In B. G. Cox, editor, *Business Survey Methods*, pages 355–384. J. Wiley, New York, 1995.

[Win01a] William E. Winkler. Quality of very large databases. Statistical Research Report Series RR2001/04, U.S. Bureau of the Census, Washington D.C., 2001.

[Win01b] William E. Winkler. Record linkage software and methods for merging administrative lists. Statistical research report series, U.S. Bureau of the Census, Washington D.C., 2001.

[Win02] W. Winkler. Record linkage and bayesian networks. In *Proceedings of the Section on Survey Research Methodology*. American Statistical Association, 2002.

[WL90] Nanny Wermuth and Steffen Lilholt Lauritzen. On substantive research hypotheses, conditional independence graphs and graphical chain models. *Journal of the Royal Statistical Society, Series B*, 52(1):21–50, 1990.

[WM89] Y. Richard Wang and Stuart E. Madnick. The inter-database instance identification problem in integrating autonomous systems. In *Proceedings of the Fifth International Conference on Data Engineering, February 6-10, 1989, Los Angeles, California, USA*, pages 46–55. IEEE Computer Society, 1989.

[Yan02] W. Yancey. Improving em parameter estimates for record linkage parameters. In *Proceedings of the Section on Survey Research Methodology*. American Statistical Association, 2002.

[Yan03] W. Yancey. An adaptive string comparator for record linkage. In *Proceedings of the Section on Survey Research Methodology*. American Statistical Association, 2003.

[Zan84] C. Zaniolo. Database relations with null values. *Journal of Computer and System Sciences*, 28:142–166, 1984.

[ZHKF95] G. Zhou, R. Hull, R. King, and J. Franchitti. Using object matching and materialization to integrate heterogeneous databases. In S. Laufmann, S. Spaccapietra, and T. Yokoi, editors, *Proc. of 3rd Intl. Conf. on Cooperative Information Systems (CoopIS95), Vienna, Austria, May 9-12, 1995.*, pages 4–18, 1995.