Data Cleaning und Record Matching

Seminar Information Integration

Christoph R. Hartel

23. Juni 2006

Betreuer:
Dipl.-Inf. Philipp Dopichaj,
AG Datenbanken und Informationssysteme,
TU Kaiserslautern

Inhaltsverzeichnis

1 Einleitung

Die Integration großer Datenmengen gewinnt immer mehr an Bedeutung. Dabei müssen wir neben der Anbindung heterogener Informationssysteme und der Integration unterschiedlicher Datenmodelle auch zahlreichen Herausforderungen bezogen auf die eigentlichen Daten begegnen. Betrachten wir zur Motivation zunächst folgende Beispiele:

1. Ein Unternehmen übernimmt ein anderes Unternehmen und muss neben der organisatorischen Integration von Mitarbeitern und Informationssystemen auch die Integration der beiden Kundendatenbanken mit jeweils 1 Mio. Datensätzen bewältigen. Da beide Unternehmen im selben Marktsegment tätig sind, überschneiden sich die Daten in vielen Fällen. Um eine reibungslose Kundenbetreuung zu gewährleisten sollen alle Kunden jedoch nur ein einziges Mal in der Datenbank auftauchen. Die Datensätze verfügen weder über ein global eindeutiges Attribut „Kundennummer“, noch über einheitliche Konventionen für die Schreibweise von Eigennamen oder die Formatierung von Adressdaten.
2. Citeseer listet 60 Verweise auf [1], darunter:
   - Ivan P. Felligi and Alan B. Sunter. A theory for record linkage. Journal of the American Statistical Society, 64:1183–1210, 1969.
   - L. Fellegi and A. Sunter. A theory for record linkage. J. of the Amer. Stat. Soc., 64:1183–1210, 1969.
   - Fellegi, I. P. and A. B. Sunter (1969). A theory of record linkage.
   - I. P. Fellegi and A. B. Sunter. A theory for record linkage. Journal of the American Statistical Association, 64(328), pages 1183–1210, Dec. 1969.
   - I. Fellegi and A. Sunter. A theory for record linkage. In JASA, 1969

   Beziehen diese sich wirklich auf denselben Aufsatz? Wie lautetet der korrekte Name des ersten Autors?
3. Welche der folgenden in [2] genannten Eigennamen beziehen sich auf dieselbe Forschungseinrichtung: „AT&T Bell Labs“, „AT&T Labs“, „AT&T Labs–Research“, „AT&T Research“, „Bell Labs“, „Bell Telephone Labs“? Fällt die Antwort auf diese Frage in verschiedenen Kontexten unterschiedlich aus?

Die Bewältigung solcher und ähnlicher Problemstellungen ist Aufgabe des *Data Cleanings*. Formal definieren wir es als den Prozess der Identifikation und Korrektur von Anomalien in einer gegebenen Datenmenge. Es umfasst u. a. die strukturelle und syntaktische Vereinheitlichung von Daten und die Erkennung und Korrektur von Fehlern. Daneben sind insbesondere die Identifikation korrespondierender Datensätze (Record Matching) und deren Zusammenführung in jeweils einen einzigen Datensatz (Record Merging) wichtige Teilprozesse des Data Cleanings.

Anwendung findet Data Cleaning in einer Vielzahl von Bereichen. So ist bspw. das Auffinden korrelierender Datensätze in bestimmten Datenquellen die Grundlage vieler Forschungsarbeiten in Biologie, Medizin und Geschichtswissenschaften. Im administrativen Bereich ist das Data Cleaning u. a. von großer Bedeutung für die Durchführung statistischer Erhebungen (z. B. Volkszählungen)

und für Fahndungsaktivitäten vor allem von Finanzbehörden. Nicht zuletzt ist die Säuberung von Daten von hoher wirtschaftlicher Relevanz sowohl im operativen Bereich (z. B. Säuberung von Kundendaten) als auch zur Fundierung strategischer Entscheidungen im Zusammenhang mit Data Warehousing und Mining (Stichwort: „garbage in, garbage out“) [3,4].

Im Gegensatz zum Schema Matching bewegen wir uns beim Data Cleaning auf der *Instanzebene*, d. h. wir betrachten die tatsächlichen Daten, nicht deren logische Schemata. Allerdings setzen nahezu alle Cleaning-Verfahren voraus, dass für die zu säubernden Daten ein gemeinsames relationales Schema existiert und alle Konflikte auf Schemaebene (z. B. Namenskonflikte von Attributen) vor der Säuberung aufgelöst wurden. Data Cleaning und Schema Matching sind daher eng verbunden[1].

Wir beschränken uns in dieser Arbeit auf die Betrachtung *textueller* Daten, da praktisch alle Publikationen in diesem Bereich implizt von solchen ausgehen. Auch gehen wir nicht auf Verfahren ein, welche auf die Säuberung von Daten ohne vorherige Überführung in ein relationales Modell abzielen. (Interessante Ansätze dieser Kategorie – vgl. z. B. [5] – beschäftigen sich insbesondere mit dem Cleaning semi-strukturierter Daten, etwa XML.)

Der weitere Aufbau dieser Arbeit gestaltet sich wie folgt: In Kapitel 2 beschäftigen wir uns mit den Grundlagen des Data Cleanings wie dem Aufbau des Cleaning-Prozesses und den verschiedenen Arten von Anomalien. Zudem führen wir am Ende des Kapitels eine Notation ein, auf die wir im Verlauf dieser Arbeit zurückgreifen. Die Kapitel 3 bis 5 beschäftigen sich mit den wesentlichen Phasen des Cleaning-Prozesses: dabei gehen wir auf die Datenanalyse- und vorverarbeitung (Kapitel 3), das Matching von Datensätzen (Kapitel 4) und die Zusammenführung korrespondierender Datensätze (Kapitel 5) ein. In Kapitel 6 diskutieren wir Kriterien für die Bewertung des Säuberungsprozesses und des produzierten Ergebnisses und geben schließlich in Kapitel 7 noch einen kurzen Überblick über existierende Frameworks und Werkzeuge zur Realisierung von Data-Cleaning-Lösungen. Kapitel 8 fasst die Ergebnisse dieser Arbeit zusammen und gibt einen kurzen Ausblick.

2 Grundlagen des Data Cleanings

In diesem Kapitel gehen wir auf eine Reihe grundlegender Aspekte des Data Cleanings ein. Dazu diskutieren wir zunächst in Abschnitt 2.1 die verschiedenen Arten von Datenanomalien. In Abschnitt 2.2 gehen wir auf den Prozess des Data Cleanings näher ein. Schließlich definieren wir in Abschnitt 2.3 noch einige wesentliche Begriffe und führen eine semi-formale Notation ein, auf die wir in den nachfolgenden Kapiteln zurückgreifen.

[1] Zur Integration von Data Cleaning und Schema Matching vgl. auch Abschnitt 3.2.

2.1 Anomalien von Daten

Eine *Anomalie* definieren wir in Anlehnung an [6] als Eigenschaft einer Menge von Datensätzen, die dazu führt, dass diese Datensätze eine falsche Repräsentation der Miniwelt darstellen. Datensätze, die keine Anomalien enthalten, bezeichnen wir als *sauber* (engl. clean).

Es existieren zahlreiche Ansätze zur Klassifizierung von Anomalien, die jedoch jeweils auf bestimme Aspekte fokussieren und keine scharfe Abgrenzung ermöglichen (vgl. z. B. [4] zu Anomalien bei der Integration mehrerer Datenquellen oder [7] zu Anomalien bei redundanten Datensätzen). Wir unterscheiden daher in Anlehnung an Müller und Freytag [6] lediglich *syntaktische* und *semantische Anomalien*[2], die sich wie folgt zusammensetzen:

Syntaktische Anomalien

- *Lexikalische Fehler* sind Anomalien in der Struktur der Datensätze. Da wir relationale Schemata zugrunde legen (vgl. Kapitel 1), können wir die korrekte logische Strukturierung der Datensätze entsprechend des jeweiligen Schemas als gegeben ansehen. Möglich sind allerdings *fehlerhaft zugeordnete Werte* (engl. misfielded values) und *eingebettete Werte* (engl. embedded values). Eine fehlerhafte Zuordnung ist z. B. ein als Vorname erfasster Nachname. Sie tritt insbesondere durch ein „Verrutschen“ in der Eingabespalte bei der Dateneingabe auf. Eingebettete Werte entstehen durch die Zusammenfassung mehrerer logischer Attribute in einem einzigen Attribut (z. B. „Adresse“ statt „Straße“, „Hausnummer“, „PLZ“ und „Ort“).
- *Formatierungsfehler* sind Abweichungen von der für die Werte eines bestimmten Attributs vorgesehenen Formatierungskonvention, z. B. „Vorname Nachname“ statt „Nachname, Vorname“. Ebenfalls als Formatierungsfehler können wir Abkürzungen („Fa.“ statt „Firma“), Synonyme („Entwickler“ vs. „Programmierer“), Dummy-Werte statt eines NULL-Wertes („999“ statt NULL) und unterschiedliche Darstellungsformen von Konstanten („Männlich“ / „Weiblich“ vs. „M“ / „F“) auffassen.

Semantische Anomalien

[2] Müller und Freytag [6] unterscheiden mit den *Abdeckungsanomalien* (engl. Coverage Anomalies) eine weitere Klasse von Anomalien, wobei sie auf das Fehlen von Informationen aus der Miniwelt im Datenmodell abstellen. Das sich daraus ergebende Problem der Vollständigkeit von Daten liegt jedoch nicht im Fokus des Data-Cleaning-Prozesses, sondern anderer Techniken der Informationsintegration, so dass wir darauf in dieser Arbeit nicht näher eingehen.

– *Verletzung von Integritätsbedingungen*[3] einzelner Attribute (z. B. Alter kleiner 0) oder funktionaler Abhängigkeiten zwischen Attributen (z. B. keine Entsprechung der Attribute „PLZ“ und „Ort“)
– *Fehlerhafte Daten* verletzen keine Integritätsbedingungen, decken sich aber nicht mit den Eigenschaften des Bezugsobjekts in der Miniwelt (vgl. Abschnitt 2.3). Einfache Beispiele sind typographische Fehler („Müllre“ statt „Müller“) und Konvertierungsfehler („Müller“ statt „Müller“). Ebenfalls fehlerhafte Daten entstehen z. B. durch vorsätzliche Verschleierung (etwa Aliase) (vgl. [8]), unterschiedliche Interpretation von Werten (Einheit Euro statt Pfund), unterschiedliche Aggregationsstufen (Tagesumsatz vs. Monatsumsatz), u. ä.
– *Duplikate* sind Datensätze, die isoliert betrachtet weder syntaktische noch semantische Anomalien aufweisen, jedoch Redundanzen enthalten (z. B. zwei identische Datensätze). Auf die exakte Definition des Begriffs und die sich aus dem Vorhandensein von Duplikaten ergebenden Probleme gehen wir in Kapitel 4 ausführlich ein.

2.2 Der Data-Cleaning-Prozess

In Kapitel 1 haben wir Data Cleaning als einen Prozess definiert. Innerhalb dieses Prozesses unterscheiden wir in Anlehnung an [4] mehrere Phasen, die Abbildung 1 zeigt. Im Folgenden geben wir einen kurzen Überblick über den Cleaning-Prozess. Im weiteren Verlauf der Arbeit gehen wir auf ausgewählte Phasen detailliert ein.

1. *Datenanalyse*: Die Analyse bildet den Auftakt des Data Cleanings. Sie identifiziert Anomalien in der Datenbasis und extrahiert über das Schema hinausgehende Metadaten für die späteren Phasen. (Vgl. Abschnitt 3.1)
2. *Workflow-Definition*: In dieser Phase wird anhand der durch die Datenanalyse gewonnenen Informationen ein Transformations-Workflow zur Bereinigung der Datenbasis definiert. Der Workflow sollte vorrangig aus automatisierten Aktivitäten bestehen, um einen effizienten Ablauf zu gewährleisten (vgl. Abschnitt 6.1), kann jedoch auch manuelle Aktivitäten beinhalten. Die Definition sollte zudem deklarativ erfolgen, um Deployment und Wartung des Workflows zu erleichtern [9,4].
3. *Workflow-Verifikation*: Vor der Ausführung müssen wir den Workflow auf Effektivität (d. h. Behebung der identifizierten Anomalien) und Effizienz (d. h. ausreichend schnelle Ausführbarkeit) überprüfen. Dazu ziehen wir i. d. R. eine Kopie eines kleinen, aber möglichst repräsentativen Ausschnitts der realen Datenbasis heran, auf der wir die Transformation ausführen und

[3] Die Integritätsbedingungen einer Datenquelle stellen i. d. R. eine Obermenge der im Schema spezifizierten Integritätsbedingungen dar, so dass dieses Problem trotz der Annahme einer im Sinne des jeweiligen Schemas korrekten Datenbasis besteht. Auf die Erkennung nicht explizit angegebener Integritätsbedingungen gehen wir im Rahmen der Datenanalyse in Abschnitt 3.1 ein.

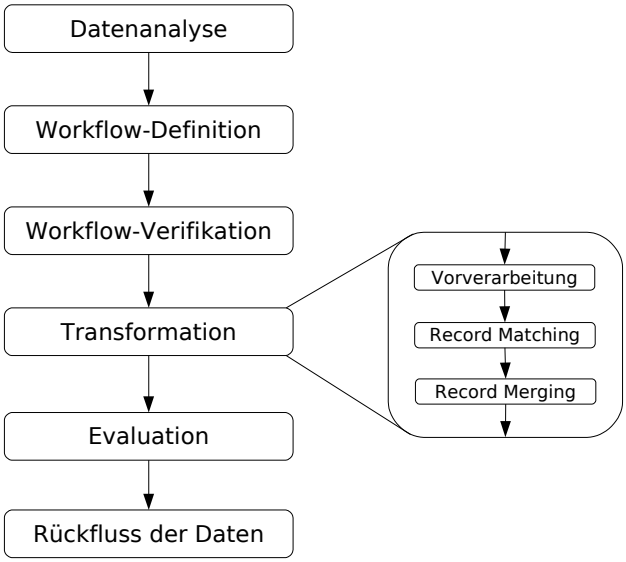

Abbildung 1. Der Data-Cleaning-Prozess

das Ergebnis analysieren. Die Phasen 1 bis 3 durchlaufen wir iterativ[4], bis schließlich ein geeigneter Workflow erstellt ist [4].

4. *Transformation* (Workflow-Ausführung): Der in den vorausgegangenen Phasen definierte Workflow wird auf der gesamten Datenbasis ausgeführt. Die Ausführung können wir wiederum als Teilprozess bestehend aus drei Phasen auffassen:
   (a) *Vorverarbeitung*: Bereinigt alle identifizierten Anomalien mit Ausnahme der Erkennung und Behandlung von Duplikaten. (Vgl. Abschnitt 3.2)
   (b) *Record Matching*: Identifiziert Duplikate in der Datenbasis. (Vgl. Kapitel 4)
   (c) *Record Merging*: Führt alle identifizierten Duplikate zu jeweils einem einzigen Datensatz zusammen. (Vgl. Kapitel 5)
5. *Evaluation*: Überprüfung der bereinigten Daten anhand definierter Qualitätskriterien (vgl. Abschnitt 6.1), um potentielle (durch den Reinigungsprozess erzeugte) Fehler zu identifizieren und das Verfahren für zukünftige Anwendungen zu optimieren. Insbesondere gleichen wir dabei die tatsächliche Qualität der Transformation mit der in der Verifikationsphase aufgestellten Prognose ab und analysieren ggf. die Gründe von Abweichungen.

[4] Vielversprechend ist in diesem Kontext der Einsatz von Werkzeugen mit direktem Benutzer-Feedback (vgl. z. B. [10]), welche den iterativen Durchlauf der verschiedenen Phasen „on the fly“ mit jeder Aktion des Benutzers ausführen. Vergleiche dazu auch Abschnitt 7.1.

6. *Rückfluss der bereinigten Daten*: In dieser Phase ersetzen wir die Daten in der bzw. den ursprüngliche(n) Datenquelle(n) durch die Ausgabe der Transformation, um eine erneute Reinigung derselben Daten in späteren Anwendungen zu vermeiden [4]. Dieser Schritt ist optional, wenngleich aus Effizienzgründen sinnvoll.

2.3 Notation

An dieser Stelle führen wir einige grundlegende Begriffe und Notationen ein, die wir im Folgenden benötigen. Einen *Datensatz* (engl. Record) definieren wir als eine geordnete Menge von Attributbelegungen $R = (a_0 = v_0, a_1 = v_1, \ldots, a_n = v_n)$. Wir nehmen dabei an, dass jeder Datensatz R genau ein Objekt in dem zugrunde liegenden konzeptionellen Modell – im Folgenden „Miniwelt“ genannt – repräsentiert (z. B. einen Kunden), welches wir *Bezugsobjekt* von R nennen[5].

Die ebenfalls geordnete Menge der von einem Datensatz belegten[6] Attribute bezeichnen wir als die *Attributmenge* des Datensatzes $R.A = (a_1, \ldots, a_n)$. Die Menge der möglichen Belegungen eines Attributs a ist seine *Wertemenge* V_a. Den Wert eines Attributs in einem Datensatz R liefert uns dann die Funktion $v : R.A \longrightarrow V$, wobei wir abkürzend v_i für den Wert des i-ten Attributs in R schreiben. Für die Mächtigkeit einer beliebigen Menge X verwenden wir die gängige Notation $|X|$.

Schließlich definieren wir noch ein *Data Set DS* als eine (nicht notwendigerweise geordnete) Menge von Datensätzen mit gleicher Attributmenge. Ein intuitives Beispiel für ein Data Set ist eine Tabelle in einem relationalen Datenbanksystem, wobei das relationale Schema die Attributmenge darstellt.

3 Datenanalyse und -vorverarbeitung

Zu Beginn des Data-Cleaning-Prozesses müssen wir die zu säubernden Daten analysieren, um Anomalien zu identifizieren und Metadaten zu gewinnen [6]; in Abschnitt 3.1 diskutieren wir daher Verfahren zur Datenanalyse und deren Ergebnisse. Die gewonnen Informationen benutzen wir dann im Rahmen der Datenvorverarbeitung in der Transformationsphase, auf die wir in Abschnitt 3.2 eingehen.

3.1 Analysetechniken

Die Datenanalyse liefert über das Schema hinausgehende Metadaten, die zum Einen aus (potentiellen) Integritätsbedingungen für die gegebene Datenmenge

[5] Diese Einschränkung dient lediglich dazu, die Begrifflichkeiten insbes. im Zusammenhang mit dem Record Matching (vgl. Abschnitt 4) zu vereinfachen. Da wir durch Schematransformation beliebige Schemata so zerlegen können, dass die korrespondierenden Datensätze diese Bedingung erfüllen, stellt die Annahme keine Beschränkung der Allgemeinheit dar.

[6] Wir bezeichnen dabei auch das Vorhandensein eines NULL-Wertes als Belegung.

und zum Anderen aus Informationen über möglicherweise inkorrekte (d. h. gegen diese Bedingungen verstoßende) Datensätze – oft als *Outliers* oder zweifelhafte Datensätze (engl. dubious Records) bezeichnet – bestehen. Zur Gewinnung dieser Daten unterscheiden wir in Anlehnung an [11] vier verschiedene Verfahrensarten – statistische, musterbasierte, distanzbasierte und regelbasierte Verfahren –, auf die wir im Folgenden kurz eingehen.

Statistische Outlier-Erkennung Statistische Methoden dienen in erster Linie der Bestimmung (wahrscheinlich) zulässiger Wertebereiche für Attribute. Bspw. berechnen wir gängige Größen wie den Durchschnitt, die Varianz und das Minimum [12] von Attributwerten. Datensätze, deren Attribute außerhalb gegebener Toleranzgrenzen liegen, stufen wir als Outlier ein. Die benötigten Größen lassen sich effizient (d. h. in linearer Zeit) bestimmen und liefern gute Resultate bei der Outlier-Identifikation. Allerdings betrachten die Verfahren keine Beziehungen zwischen Attributwerten[7].

Distanzbasierte Outlier-Erkennung Diese Verfahren berechnen Abstände zwischen Datensätzen basierend auf Ähnlichkeitsmetriken wie z. B. Editierabständen oder phonetischer Ähnlichkeit von Attributwerten. Anhand dieser Werte können wir die Datensätze in Cluster (also Gruppen mit jeweils geringen Abständen der Datensätze untereinander) einteilen. Datensätze, die zu allen Clustern einen hohen Abstand aufweisen, betrachten wir dann als Outlier [11]. Die verwendeten Metriken sind analog zu den im Rahmen des Record Matching verwendeten (vgl. Abschnitt 4.3), so dass wir an dieser Stelle nicht näher darauf eingehen. Der wesentliche Nachteil der distanzbasierten Verfahren besteht darin, dass sich aus dem dafür notwendigen Vergleich aller Datensätze eine quadratische Laufzeit ergibt, während die anderen Verfahren in deutlich geringeren Schranken (z. T. sogar in $O(N)$) liegen.

Musterbasierte Outlier-Erkennung Hierbei gehen wir von einer Menge von Mustern (engl. Pattern) aus, die jeweils auf viele Datensätze zutreffen. Ein Muster besteht aus einer Reihe bestimmter Merkmalen von Attributwerten eines Datensatzes, z. B. der Länge eines Strings oder fehlende Werte bei bestimmten Attributen. Datensätze, auf die keines der Muster zutrifft, klassifizieren wir als Outlier. Die zugrunde liegende Muster-Menge kann entweder manuell vorgegeben sein oder durch Verfahren des Maschinenlernens gewonnen werden. Allerdings weisen Datenmengen in praktischen Anwendungskontexten oft nur sehr geringe Korrelationen einzelner Attribute auf, so dass keine trennscharfen Muster existieren und diese Verfahren daher schlechte Ergebnisse liefern [13].

[7] Unter dem Begriff „statistische Verfahren“ werden in diesem Kontext üblicherweise nur solche Verfahren subsumiert, die sich auf einzelne Attribute beziehen. Insbesondere die sog. regelbasierten Verfahren sind jedoch letztlich ebenfalls statistische Modelle, so dass diese Bezeichnungen irreführend sind.

Regelbasierte Outlier-Erkennung Regelbasierte Verfahren verallgemeinern Muster-basierte Verfahren, indem sie nicht isolierte Attribute, sondern Beziehungen zwischen mehreren Attributen (z. B. zwischen PLZ und Ort oder zwischen Alter und Gehalt) betrachten. Im einfachsten Fall hat eine Regel die Form (v_i, v_j, σ), wobei v_i, v_j jeweils der Wert des i-ten bzw. j-ten Attributs ist und σ ein beliebiger binärer Operator, z. B. „=“. Der Anteil der Datensätze, in welchen v_i und v_j zusammen auftreten, bezeichnen wir als *Unterstützung* der Regel (engl. Support); den Anteil dieser Datensätze, in denen außerdem $v_i \, \sigma \, v_j$ gilt, bezeichnen wir als *Konfidenz* (engl. Confidence) der Regel [13]. Die Regeln werden automatisch aus der Datenmenge abgeleitet, wobei wir nur solche Regeln betrachten, deren Konfidenz ein definiertes Minimum überschreitet. Alle Datensätze, welche die gefundenen Regeln nicht erfüllen, betrachten wir als Outlier.

Es existieren zahlreiche Varianten von Regel-basierten Verfahren, die i. d. R. auf eine bestimme Art von Werten abstellen, z. B. ordinal skalierte Werte (vgl. [13]), numerische Intervalle (vgl. [12]), u. v. m. Diese Verfahren – sofern sie die Ableitung und Überprüfung von Regeln in $O(N)$ ermöglichen – sind eine vielversprechende Analysemethode, die eine hohe Genauigkeit bei der Outlier-Identifizierung ermöglicht.

3.2 Vorverarbeitungstechniken

Die Vorverarbeitung von Datensätzen hat zwei Aufgaben: Validierung und Normalisierung. Unter *Validierung* verstehen wir die Erkennung und Beseitigung bestimmter semantischer Anomalien, namentlich Verletzungen von Integritätsbedingungen und fehlerhafte Daten (vgl. Abschnitt 2.1)[8]. Die *Normalisierung* überführt Datensätze in ein syntaktisch und lexikalisch einheitliches Format, beseitigt also alle in Abschnitt 2.1 aufgeführten syntaktischen Anomalien.

Ein wichtiges Merkmal dieser Techniken ist, dass sie i. d. R. von der Anzahl der Datensätze nur linear abhängen, d. h. bezogen auf N Datensätze in $O(N)$ liegen. Da spätere Phasen – insbes. das Record Matching (vgl. Abschnitt 4) – im schlimmsten Fall in $O(N^2)$ liegen, ist das Vornehmen von Korrekturen in der Vorverarbeitungphase deutlich effizienter und unterstützt zudem die nachfolgenden Phasen[9]. In den folgenden beiden Abschnitten gehen wir kurz auf verschiedene Aspekte und mögliche Verfahren zur Daten-Validierung und -Normalisierung ein. Einen umfassenden Überblick über die Vorverarbeitung von Daten gibt z. B. [14].

[8] Die Erkennung und Entfernung von Duplikaten erfolgt hingegen aufgrund ihrer Komplexität nicht im Rahmen der Validierung, sondern dediziert in den Phasen Matching und Merging (vgl. Abschnitte 4 und 5).

[9] Bspw. müssen wir im Record Matching keine komplexen Ähnlichkeitsmaße zur Berücksichtigung von Abkürzungen einsetzen, wenn wir diese bereits in der Vorverarbeitung durch ihre jeweiligen Langformen ersetzen. Auch können wir z. B. durch eine Korrektur typographischer Fehler u. a. das Clustering während des Matchings und das Zusammenführen von Datensätzen effizienter und effektiver durchführen.

Normalisierung Die Normalisierung umfasst verschiedene Bereiche, darunter vor allem den sog. Attribute Split und die Standardisierung der Syntax von Attributwerten. Als *Attribute Split*[10] bezeichnen wir die Aufteilung eines Stringwertigen Attributs in mehrere Attribute, um eine genauere Repräsentation des Bezugsobjekts zu erreichen und damit die spätere Säuberung zu erleichtern [4]. Ein Beispiel für einen Attribute Split wäre die Aufteilung eines Attributs „Adresse“ in die Attribute „Straße“, „Hausnummer“, „PLZ“, „Ort“ und „Land“. Diese Aufteilung erfordert i. d. R. domänenspezifisches Wissen

Die *Standardisierung* umfasst u. a. das Angleichen von Schreibweisen wie z. B. das Ersetzen von Abkürzungen durch ihre jeweiligen Langformen und die Vereinheitlichung der Reihenfolge von Termen in Attributwerten. Beispiele für die Standardisierung sind das Ersetzen der Zeichenfolge „AG“ durch „Aktiengesellschaft“ und die Transformation von „Müller, Hans“ nach „Hans Müller“.

Im weiteren Sinne können wir auch die *Schemaintegration* als einen Teil der Normalisierung betrachten [4]. Alle Data-Cleaning-Verfahren setzen voraus, dass die zu säubernde Datenquelle in ein konsistentes, i. d. R. relationales Schema überführt wurde bzw. im Fall mehrerer Datenquellen, dass diese in ein solches zusammengefasst sind oder gefasst werden können[11]. Insbesondere setzen wir im Falle mehrerer Datenquellen voraus, dass die Abbildungsvorschriften für Attribute (nicht jedoch Attributwerte[12]) zwischen den Datenquellen definiert sind. Aufgrund der Komplexität der Schemaintegration wird diese aber i. d. R. als eigenes Problemfeld betrachtet[13].

Zur Umsetzung der beschriebenen Normalisierungen können wir nahezu alle Techniken verwenden, die wir im Zusammenhang mit Matching-Verfahren beschreiben (vgl. Kapitel 4). Dazu zählen u. a. Heuristiken, probabilistische Modelle wie z. B. Hidden Markov Models und Ontologien. Eine umfassende Beschreibung der Umsetzung am Beispiel einer Adressnormalisierung findet sich in [16].

Validierung Die Validierung dient der Korrektur aller semantischer Anomalien (vgl. Abschnitt 3.1) mit Ausnahme von Duplikaten. Dazu zählt vor allem die Erkennung und Behebung folgender Fehler:

- *Typographische Fehler*, z. B. „Hasn“ statt „Hans“
- *Ausreißer*, d. h. solcher Attributwerte, die zwar den Schemarestriktionen entsprechend, aber semantisch mit hoher Wahrscheinlichkeit falsch sind, z. B. Geburtsjahr „1897“ statt „1987“ in einer Personaldatenbank

[10] Dieser Vorgang wird oft auch als „Elementizing“ (dt. etwa Zerlegung) bezeichnet.

[11] Je nach Vefahren betrachten wir die Datensätze in ihren ursprünglichen Schemata (vgl. z. B. Abschnitt 4.4) oder in einem einzigen integrierten Schema (vgl. z. B. Abschnitt 4.2). In jedem Fall setzten wir aber voraus, dass alle Konflikte auf Schemaebene vor Anwendung der Cleaning-Transformationen aufgelöst wurden.

[12] Attributwerte betrachten wir als Instanz-bezogen und daher als Teil des Data-Cleaning-Prozesses. In Abschnitt 4.4 gehen wir auf dieses Problem näher ein.

[13] Einen Überblick über das Problem der Schemaintegration gibt z. B. [15].

– *Inkonsistenzen* zwischen abhängigen Attributen, die nicht von Schemarestriktionen erfasst werden, z. B. widersprüchliche Angaben in den Attributen „PLZ“ und „Ort“

Zumindest für die beiden letztgenannten Fehlerarten benötigen wir dabei offensichtlich über das Schema hinausgehende Metadaten, die wir zuvor in der Analysephase gewonnen haben. Diese können z. B. Histogramme der einzelnen Attributwerte (zur Erkennung von Ausreißern) und Regeln für Attributinterdependenzen (zur Erkennung von Inkonsistenzen) enthalten.

Die Realisierungsmöglichkeiten verhalten sich analog zu den im vorangegangenen Abschnitt beschriebenen, so dass wir an dieser Stelle nicht näher darauf eingehen.

Im nächsten Abschnitt diskutieren wir die Phase des Record Matchings, die auf die Vorverarbeitung folgt.

4 Record Matching

Um den Prozess des Record Matchings zu beschreiben, führen wir in Abschnitt 4.1 zunächst einige grundlegende Begriffe ein und erläutern das Modell von Fellegi und Sunter [1], das der theoretischen Fundierung der nachfolgenden Ausführungen dient. In Abschnitt 4.2 stellen wir prototypisch verschiedene Verfahren zur Durchführung des Record Matchings vor und vergleichen im nachfolgenden Abschnitt eine Reihe von Metriken, die diese Verfahren zur Bewertung der Ähnlichkeit von Datensätzen nutzen können. In Abschnitt 4.4 gehen wir schließlich noch kurz auf die Besonderkeiten nominal skalierter Attributwerte in diesem Kontext ein.

4.1 Grundlagen

In diesem Abschnitt erläutern wir die Grundlagen des Record Matchings und gehen dabei zunächst in Abschnitt 4.1 auf den Begriff des „Duplikats“ und die Äquivalenz von Datensätzen ein. In Abschnitt 4.1 beschreiben wir dann das Modell von Fellegi und Sunter als formale Grundlage der nachfolgenden Ausführungen.

Duplikate Das *Record Matching*[14] verfolgt das Ziel, in der Menge von Datensätzen alle Duplikate zu identifizieren. Dabei bezeichnen wir einen Datensatz R_1 als *Duplikat* eines anderen Datensatzes R_2 (mit $R_1 \neq R_2$), wenn beide Datensätze dasselbe Bezugsobjekt repräsentieren. In diesem Fall sagen wir auch: R_1 ist *äquivalent*[15] zu R_2, kurz: $R_1 \equiv R_2$ [17].

[14] In der Literatur existieren zahlreiche andere Begriffe für diesen Vorgang (etwa Record Linkage, Record Reconciliation, Object Identification, u. v. m. Allerdings werden diese weitgehend synonym gebraucht.

[15] Dabei ist es wichtig anzumerken, dass die beiden Datensätze eines Links keineswegs den gleichen Informationsgehalt haben müssen – obwohl der Begriff der Äquivalenz

Die Duplikat-Beziehung ist symmetrisch (d. h. wenn R_1 ein Duplikat von R_2 ist, so ist R_2 auch ein Duplikat von R_1) und transitiv. Reflexivität haben wir hingegen durch die Definition ausgeschlossen, um die Handhabung der Begriffe zu vereinfachen. (Das heißt ein Datensatz ist kein Duplikat seiner selbst.) Wir unterscheiden zudem exakte Duplikate und nicht-exakte Duplikate: ein Duplikat R_1 ist ein *exaktes Duplikat* von R_2, wenn es die gleiche Menge von Attributen A enthält und jedes der Attribute in R_1 denselben Wert aufweist wie in R_2. Andernfalls bezeichnen wir R_1 als *nicht-exaktes Duplikat* von R_2.

Fellegi-Sunter-Modell Die formalen Grundlagen des Record Matching gehen im Wesentlichen auf Fellegi und Sunter [1] zurück, die im Jahr 1969 die verschiedenen, bis dahin existierenden Ansätze in das im Folgenden beschriebene Modell verallgemeinert haben [18]. Dabei sei betont, dass es sich bei diesem Modell um einen *abstrakten* Ablauf handelt, den wir zur Anwendung auf ein konkretes Problem um konkrete Funktionsdefinitionen bzw. Algorithmen – wie z. B. die später diskutierten Matching-Verfahren – erweitern müssen.

Seien A, B zwei Mengen von Datensätzen, die wir zu einer gemeinsamen Menge $A \times B = \{(a, b); a \in A, b \in B\}$ zusammenführen wollen. Dann existieren zwei Mengen

$$M = \{(a, b); a \in A, b \in B, a \equiv b\} \qquad U = (A \times B) \backslash M$$

wobei wir M als Menge der Matches, und U als Menge der Nicht-Matches bezeichnen. Aufgrund der Symmetrie der Duplikat-Beziehung gilt dabei $(a, b) = (b, a)$, d. h. für zwei Datensätze, die Duplikate voneinander sind, existiert nur *ein* Match.

Zum Matching der Datensätze definieren wir einen *Vergleichsraum* (engl. Comparison Space) Γ, der durch beliebige Kriterien aufgespannt wird. Beispiele für solche Kriterien sind „Vorname stimmt überein“, „Nachname ist ähnlich“ und „Geburtsdatum weicht maximal um 2 Jahre ab“. Eine Vergleichsfunktion $comp : A \times B \longrightarrow \Gamma$ bildet Tupel (a, b) auf einen binären Vektor $\gamma \in \Gamma$ ab, also z. B. auf $(1, 1, 0)$, wenn die ersten beiden Kriterien erfüllt sind, das dritte jedoch nicht.

Im nächsten Schritt bestimmt eine *Entscheidungsfunktion* (engl. Linkage Rule) $dec : \Gamma \longrightarrow \{L, PL, NL\}$, ob das betrachtete Tupel in der Menge der Links L, der Nicht-Links NL oder der möglichen Links PL liegt. Als *Links* bezeichnen wir dabei solche Datensatz-Tupel, welche die Entscheidungsfunktion als Duplikate klassifiziert, als *Nicht-Links* analog solche Tupel, die nicht als Duplikate klassifiziert werden. Tupel, die die Entscheidungsfunktion nicht eindeutig zuordnen kann, bezeichnen wir als *mögliche Links*.

Als Grundlage der Entscheidung dienen dabei im traditionellen Modell die Wahrscheinlichkeiten, dass γ auftritt, unter der Voraussetzung, dass (a, b) in

dies suggeriert –, d. h. insbesondere nicht notwendigerweise dieselbe Menge von Attributen des repräsentierten Miniwelt-Objekts beschreiben. In Kapitel 5 gehen wir näher auf das Problem des Informationsgehalts von Datensätzen ein.

M respektive in U liegt (genauer: das Ereignis „$(a, b) \in M$“ bzw. „$(a, b) \in U$“ auftritt). Formal lässt sich das in der *Agreement Ratio*

$$r(\gamma) = \frac{P(\gamma|(a,b) \in M)}{P(\gamma|(a,b) \in U)}$$

ausdrücken. Die Entscheidungsfunktion für diesen Fall ergibt sich daher zu

$$dec(\gamma) = \begin{cases} L, & \text{falls } r(\gamma) > t_{upper} \\ NL, & \text{falls } r(\gamma) < t_{lower} \\ PL, & \text{sonst} \end{cases} \tag{1}$$

wobei t_{upper}, t_{lower} den Grenzwert für einen Link bzw. Nicht-Link angeben. Aktuelle Record-Matching-Verfahren sind bestrebt t_{upper} und t_{lower} gleichzusetzen, um die Menge PL zu eliminieren[16]. Somit ergibt sich eine binäre Klassifikation

$$dec(\gamma) = \begin{cases} L, & \text{falls } r(\gamma) > t \\ NL, & \text{sonst} \end{cases} \tag{2}$$

mit einem einzigen Grenzwert t. Auf die Problematik der Wahl von t gehen wir in Abschnitt 6.2 ein. Vielen Verfahren nutzen zudem Alternativen zur probabilistischen Einteilung von Datensätzen, die wir in den folgenden Abschnitten diskutieren.

4.2 Matching-Verfahren

Das Matching von Datensätzen ist eine spezielle Form der Join-Operation in Datenbanksystemen: ein Join vergleicht alle Datensätze zweier Datenmengen DS_1, DS_2 anhand eines definierten Operators, eliminiert Duplikate und führt sie in eine einzige Menge zusammen. Im Gegensatz zu einem einfachen Equi-Join, der Datensätze durch Prüfung eines bestimmten Attributs (z. B. eines Schlüssels) auf exakte Übereinstimmung vergleicht, befasst sich das Record Matching speziell mit dem Vergleich von Datensätzen ohne global eindeutige Schlüsselattribute (vgl. Kapitel 1). Zudem betrachtet das Record Matching nur die Identifikation von Duplikaten, da die eigentliche Behandlung im Record Merging erfolgt und über eine bloße Eliminierung von Datensätzen hinausgeht.

Die Verfahren zur Durchführung des Record Matchings leiten sich jedoch aus den gängigen Join-Implementierungen ab. Betrachten wir als Ausgangspunkt den sehr einfach zu implementierenden Nested-Loop-Algorithmus, der beliebige Vergleichsoperatoren unterstützt. Das Verfahren vergleicht zur Identifizierung von Matches jeden Datensatz in DS_1 mit jedem Datensatz in DS_2, so dass es – unter der Annahme, dass $N = |DS_1| \approx |DS_2|$ – offensichtlich in $O(N^2)$ liegt. Da für

[16] Daraus ergibt sich der Vorteil, dass keine manuelle Klassifizierung von Datensätzen mehr erforderlich ist. In Abschnitt 6.1 gehen wir darauf näher ein. (Streng genommen zielen wir nicht auf eine *Gleich*setzung der Grenzwerte ab, sondern auf die Benutzung eines einzigen Grenzwertes, da sonst der Fall $r(\gamma) = t_{upper} = t_{lower}$ offensichtlich nicht eindeutig definiert wäre.)

das Data Cleaning in der Regel enge Zeit- und Hardwarebeschränkungen bestehen (siehe Kapitel 6), ist diese Herangehensweise für praktische Anwendungen nicht geeignet [19,20].

Daher verwenden aktuelle Matching-Verfahren insbesondere *Blocking-Mechanismen*. Diese zielen darauf ab, den teuren Vergleich von Datensätzen auf jeweils möglichst wenige, aussichtsreiche Kandidaten zu beschränken [21]. Dazu unterteilen wir die Datenmenge in Segmente (*Blöcke*), von denen wir annehmen, dass zu jedem Datensatz nur innerhalb seines jeweiligen Blocks potentielle Duplikate existieren. Daher müssen wir jeden Datensatz auch nur mit allen Datensätzen innerhalb desselben Blocks vergleichen, was die Kosten des Verfahrens deutlich reduziert.

Im Folgenden diskutieren wir prototypisch das *Sorted-Neighbourhood-Verfahren*, kurz SNV, um die Idee des Blockings zu veranschaulichen. Das SVN zeichnet sich zudem dadurch aus, dass es den Ausgangspunkt für nahezu alle anderen aktuellen Record-Matching-Verfahren darstellt: Wenn wir ein beliebiges Matching-Verfahren abstrakt als die Kombination eines bestimmten Algorithmus zur Ablaufsteuerung mit einem oder mehreren Vergleichsoperatoren beschreiben, dann optimieren alternative Verfahren entweder Teile des Ablaufs (vgl. z. B. [8]) oder die verwendeten Operatoren (vgl. z. B. [22,20,17]), weichen jedoch nicht grundsätzlich von der Idee des SNV ab.

Sorted-Neighbourhood-Verfahren Das SNV geht zurück auf Hernández und Stolfo [19] und ist eine Erweiterung der Sort-Merge-Implementierung der Join-Operation. Wir unterstellen dabei, dass die zu matchenden Datensätze bereits in einer einzigen Menge vereinigt sind. (Ist dies zu Beginn nicht der Fall, so können wir diesen Zustand durch die Anwendung des Union-Operators auf die beiden Ausgangsmengen erreichen.) Das Verfahren berechnet dann zunächst für jeden Datensatz einen (nicht notwendigerweise eindeutigen) Schlüssel, wobei i. d. R. die am stärksten diskriminierenden Attribute zur Berechnung herangezogen werden. Ein typischer Schlüssel könnte z. B. für einen Datensatz „Person“ aus den ersten drei Buchstaben des Nachnamens konkateniert mit den ersten drei Stellen der Personalnummer bestehen. Anschließend sortiert das Verfahren alle Datensätze anhand ihrer Schlüssel. Dabei liegt die Annahme zugrunde, dass Duplikate jeweils ähnliche Attributwerte (und somit ähnliche Schlüssel) aufweisen und sich daher nach der Sortierung in direkter oder naher Nachbarschaft zueinander befinden.

In einer zweiten Phase wird iterativ ein „Fenster“ der Größe k über die Datensätze bewegt, das in der ersten Iteration die Datensätze $R_0, \ldots, R_{k-1}$ betrachtet, in der zweiten Iteration $R_1, \ldots, R_k$, usw. Das Fenster stellt den Block dar, innerhalb dessen der Algorithmus jedes Paar von Datensätzen vergleicht. Die algorithmische Komplexität des gesamten Verfahrens ergibt sich daher für N Datensätze und $k < \lceil logN \rceil$ als $O(NlogN)$, liegt also deutlich unter der der Nested-Loop-Variante. Eine gängige Optimierung ist die Verwendung von Clustering-Algorithmen alternativ zu der hier verwendeten Sortierung der Datensätze, die eine Verbesserung auf $O(NlogN/C)$ ermöglicht. Da sich das Ver-

fahren ansonsten identisch verhält, gehen wir an dieser Stelle jedoch nicht näher darauf ein.

Nachteilig an allen Blocking-basierten Verfahren ist die potentielle Verschlechterung der Ergebnisqualität, da ggf. außerhalb des betrachteten Blocks existierende Matches offensichtlich nicht erkannt werden können. Bei dem SNV ist dieses Problem besonders gravierend, da die Zuordnung eines Datensatzes zu einem Block ausschließlich aufgrund der Sortierung erfolgt und damit nur von dem für den Datensatz berechneten Schlüssel abhängt. Dieser weist jedoch nur eine geringe Fehlertoleranz auf: lautet in obigem Beispiel der Nachname eines Personendatensatz etwa fälschlicherweise „üller“ statt „Müller“, so positionieren wir den betreffenden Datensatz (mit hoher Wahrscheinlichkeit) nicht in der Nachbarschaft äquivalenter Datensätze [19].

Multi-Pass-Sorted-Neighbourhood-Verfahren Eine mögliche Optimierung des SNV ist das *Multi-Pass-Sorted-Neighbourhood-Verfahren*, kurz MP-SNV [19]. Es zielt darauf ab, die Genauigkeit zu steigern und trotzdem mit möglichst kleinen Fenstergrößen – und damit einer akzeptablen Laufzeitkomplexität – auszukommen. Statt eines Durchlaufs mit einem einzigen berechneten Schlüssel führt das MP-SNV mehrere, unabhängige Läufe des SNV mit kleinen Fenstergrößen k und unterschiedlichen Schlüsseln durch. Durch das kleine k, sind die einzelnen Läufe effizient; zudem lassen sie sich durch ihre Unabhängigkeit voneinander parallelisieren und so der Gesamtzeitaufwand reduzieren.

In einem zweiten Schritt bildet das MP-SNV dann die transitive Hülle der Ergebnisse des ersten Schritts, also der einzelnen SNV-Ausführungen. Da alle Record-Matching-Verfahren als Ergebnis eine Menge von Links liefern, hat die transitive Hülle die Form:

$$L^\star := \{(R_1, R_2) \mid R_1, R_2 \in D, R_1 \equiv R_2\}$$

D ist dabei die Menge aller Datensätze. Anschaulich enthält $L^\star$ also alle Links in der Grundmenge D, die in mindestens einem SNV-Lauf identifiziert wurden, sowie alle sich daraus durch Transitivität ergebenden Links. Das Bilden der transitive Hülle bewirkt eine weitgehende Unabhängigkeit der Erkennung von Duplikaten von den Sortierungsreihenfolgen in den einzelnen SNV-Läufen. Insbesondere können auch solche Duplikate zuverlässig erkannt werden, deren berechneter Schlüssel in einem einzelnen Lauf aufgrund eines Fehlers in den zur Schlüsselberechnung verwendeten Attributen Abweichungen aufweist [19].

Unter der Voraussetzung, dass die Mengen der in den einzelnen SNV-Läufen zur Schlüsselberechnung herangezogenen Attribute (weitgehend) disjunkt sind und ein Datensatz nicht in *allen* Attributwerten Fehler enthält, bietet das MP-SNV somit eine gute Genauigkeit bei vertretbarer Laufzeit. Ein mögliches Problem des Verfahrens ist die mögliche Fortpflanzung von Fehlern in der Duplikat-Erkennung durch die Verwendung der transitiven Hülle. Mong [17] führt allerdings an, dass i. d. R. nur wenige, über den Suchraum verteilte Duplikate eines

Datensatzes existieren und verschiedenen Tests daher ergeben haben, dass nur in seltenen Fällen eine Fehlerfortpflanzung auftritt.

Inkrementelles Sorted-Neighbourhood-Verfahren Eine weitere Variante des SNV ist das *Inkrementelle Sorted-Neighbourhood-Verfahren*, kurz I-SNV [8]. Es zielt darauf ab, dass Record Matching für den häufigen Anwendungsfall zu optimieren, dass ein Datenbestand kontinuierlichen Erweiterungen (Insert) unterliegt und daher regelmäßig gesäubert wird (z. B. in einem Data Warehouse). In vielen Fällen stellen die neu hinzugefügten Datensätze (z. B. neu erfasste Kunden) nur einen sehr geringen Anteil des Gesamtdatenbestandes dar, so dass ein wiederholtes Ausführen des SNV auf allen Datensätzen in hohem Maße ineffizient ist.

Das I-SNV betrachtet äquivalente Datensätze als einen Cluster (s. o.) und wählt zu jedem Cluster jeweils einen oder mehrere Datensätze als *primäre Repräsentanten* (engl. Prime Representative), die stellvertretend für ihren Cluster später bei der Vergleichsoperation eingesetzt werden. (Für einen primären Repräsentanten schreiben wir im Folgenden kurz R_p.) Eine beliebige Menge neu eingefügter Datensätze vereinigen wir mit der Menge aller R_p und wenden auf die Vereinigung das MP-SNV an. Damit ordnen wir jeden neuen Datensatz R entweder einem bestehenden Cluster zu (wenn einer seiner R_p äquivalent zu R ist) oder er bildet einen neuen Cluster. Die teure Matching-Operation müssen wir somit für jeden Datensatz (mit Ausnahme der R_p) nur einziges Mal ausführen, so dass sich die Laufzeit des Matching-Prozesses bei der wiederholten Anwendung auf einen (inkrementell erweiterten) Datenbestand deutlich reduziert.

Allerdings steigt die kumulierte Laufzeit aller Matching-Prozesse gegenüber dem MP-SNV leicht an und der Anteil korrekt erkannter Links sinkt geringfügig. Ein weiterer Nachteil des Verfahrens ist der Umgang mit Modifikationen an vorhandenen Datensätzen (Update, Delete): Insbesondere müssen wir alle Modifikationsoperationen erfassen (Verwaltungsoverhead!) und jeweils auf alle Datensätze in den betroffenen Clustern erneut das MP-SNV anwenden. Gravierendster Nachteil des I-SNV ist allerdings die Voraussetzung, dass die zugrunde liegenden Ähnlichkeitsmetriken invariant für alle Läufe des Verfahrens sind. Ändert sich eine Metrik, so müssen wir das Clustering für die *gesamte* Datenbasis erneut ausführen, um eine konsistente Clusterzuordnung neuer Datensätze zu gewährleisten[17].

Im Folgenden betrachten wir diejenigen Optimierungen, die nicht den Ablauf des Verfahrens, sondern die verwendeten Operatoren betreffen.

[17] Im Vergleich zu den anderen Verfahren ist dieser Nachteil natürlich zu relativieren, da diese ohnehin bei jedem Cleaning-Lauf auf der gesamten Datenbasis arbeiten. Da der Vorteil des I-SNV – also gerade der Wegfall dieser Eigenschaft – damit jedoch zumindest einmalig neutralisiert wird, stellt die Invarianz der Metrik durchaus einen Nachteil des Verfahrens dar.

4.3 Ähnlichkeitsmetriken

Wie im vorangegangenen Abschnitt diskutiert, benötigen wir für das Record Matching einen im Vergleich zu einem Equi-Join komplexeren Vergleichsoperator, der – entsprechend dem Modell von Fellegi und Sunter – beliebiger Natur sein kann. Als Vergleichskriterium ziehen wir die *Ähnlichkeit* (engl. Similarity) zweier Datensätze R_1, R_2 heran. Im Gegensatz zur Äquivalenz, die uns die semantische Übereinstimmung der Datensätze anzeigt (vgl. Abschnitt 4.1), definieren wir Ähnlichkeit als den Grad der syntaktischen und lexikalischen Übereinstimmung zwischen R_1 und R_2.

Zur Vereinfachung der Notation fassen wir eine Ähnlichkeitsmetrik als eine Funktion $sim : DS \times DS \longrightarrow [0, \ldots, 1] \subset \mathbb{R}$ auf, die für zwei beliebige R_1, R_2 eine reelle Zahl zwischen Null (einschließlich) und Eins (einschließlich) liefert. (Das Intervall $[0, \ldots, 1] \subset \mathbb{R}$ schreiben wir im Folgenden kurz als I_{01}.) Der Wert 1 stehe dabei die völlige Übereinstimmung der Datensätze (d. h. R_1 und R_2 sind identisch), 0 entsprechend für das Fehlen jeglicher Ähnlichkeit.

Zur Bestimmung der Ähnlichkeit existieren zahlreiche Metriken, die unabhängig von dem verwendeten Matching-Verfahren sind. Viele dieser Metriken bestimmen die Ähnlichkeit der Datensätze aus der gewichteten[18] Ähnlichkeit ihrer Attributwerte. Definieren wir die Ähnlichkeit zweier Attributwerte v_1, v_2 analog zur Ähnlichkeit von Datensätzen und $sim(v_1, v_2)$ analog als Funktion zu ihrer Bestimmung, so ergibt sich die Metrik zu

$$sim(R_1, R_2) = \sum_{i=0}^{|A|} w_i \cdot sim(v_i^{R_1}, v_i^{R_2})$$

wobei $w_i \in I_{01}$ der Gewichtungsfaktor des i-ten Attributs ist.

Im Folgenden betrachten wir daher – sofern nicht explizit angegeben – Metriken für Attribut-Distanzen. Dabei ist anzumerken, dass diese, bis auf wenige Ausnahmen (vgl. z. B. Abschnitt 4.4), auf den Vergleich String-wertiger Attribute ausgerichtet sind. Für kardinal skalierte Attribute (etwa „Alter“) lassen sich sehr einfache Metriken bilden, so dass diese keiner weiteren Betrachtung bedürfen. (Denkbar sind insbes. die (optional gewichtete) Differenz zweier Werte sowie die Interpretation der Werte als Strings, so dass die darauf ausgerichteten Verfahren anwendbar sind.) Eine gesonderte Betrachtung erfordern hingegen nominal skalierte Werte („Enums“), auf die wir in Abschnitt 4.4 gesondert eingehen.

Mit den im Folgenden diskutierten String-Metriken wollen das große Spektrum möglicher Ansätze illustrieren. Wir erheben allerdings keinerlei Anspruch auf Vollständigkeit.

[18] Die Gewichtung bestimmt, wie stark die Ähnlichkeit eines einzelnen Attributs die Ähnlichkeit des gesamten Datensatzes beeinflusst. So sind Namen und insbesondere Id-Strings (wie etwa Personalnummern) i. d. R. deutlich stärkere Indikatoren für eine Äquivalenz zweier Datensätze als bspw. Alter oder Gehalt.

Editierabstand Der *Editierabstand* (engl. Edit Distance) zweier String-Werte s_1, s_2 ist die minimale Anzahl an Operationen, die notwendig sind, um s_1 in s_2 zu überführen. Zur Berechnung des Abstandes existieren verschiedene Verfahren, von denen wir beispielhaft einige gebräuchliche kurz erläutern.

Ein sehr einfaches Beispiel eines Editierabstandes ist die *Hamming-Distanz*. Sie definiert als Operationen ausschließlich Zeichenersetzungen und ist daher nur für Strings gleicher Länge anwendbar. Aufgrund dieser Einschränkung scheidet sie für die Verwendung in Record-Matching-Verfahren praktisch aus.

Eine Verallgemeinerung der Hamming-Distanz ist die *Levenshtein-Distanz*, die durchaus Anwendung im Kontext von Data-Cleaning-Anwendungen findet. Sie definiert als Operationen neben dem Ersetzen auch das Einfügen und Löschen von Zeichen, so dass sie für beliebige Strings berechnet werden kann. Mögliche Erweiterungen umfassen das Betrachten der Vertauschung von Zeichen als eine weitere (einzelne) Operation, um häufige Eingabefehler besser abzubilden, und das unterschiedliche Gewichten der Operationen [23]. Nachteilig an der Levenshtein-Distanz ist jedoch, dass Abkürzungen (z. B. „Dr." vs. „Doktor") zu sehr großen Distanzwerten führen und daher nicht zuverlässig erkannt werden können.

Metriken zur Berücksichtigung von Abkürzungen Ein einfacher Algorithmus, der das Matching von Strings trotz des Vorhandenseins von Abkürzungen erlaubt, ist der *Rekursive Field-Matching-Algorithmus* [22]. Er basiert auf der Berechnung einer *Matching Score* S, die sich durch folgende Funktion ergibt:

$$score(s_1, s_2) := \begin{cases} 1, & s_1 = s_2 \vee s_1 \subset s_2 \\ 0, & \text{sonst} \end{cases}$$

Dabei bezeichne $=$ die Beziehung „ist identisch mit" und $\subset$ „ist Abkürzung von".

Die Prüfung der Identität ($=$) für zwei beliebige Strings ist offensichtlich trivial. Die Erkennung der $\subset$-Beziehung erfolgt durch reguläre Ausdrücke, die folgende Fälle abdecken:

1. s_1 ist ein Präfix von s_2 (z. B. „Fr" statt „Frau")
2. s_1 ist ein Präfix von s_2 konkateniert mit einem Suffix von s_2 (z. B. „Fa" statt „Firma")
3. s_1 ist eine Konkatenation mehrerer Präfixe beliebiger Teilstrings (getrennt durch Leerzeichen) von s_2 (z. B. „IEEE" statt „Institute of Electrical and Electronics Engineers")

Abkürzungen, die nicht aus Prä- bzw. Suffixen bestehen, unterstützt dieser Ansatz nicht, d. h. er könnte bspw. „TU" zwar mit „Technischer Universität" identifizieren, nicht jedoch „DBIS" mit „Datenbanken und Informationssysteme". Außerdem versagt der Ansatz bei Vertauschungen in der Reihenfolge von Worten. Zudem liegt der Algorithmus in $O(N^2)$, da jedes Zeichen in s_1 mit jedem Zeichen in s_2 verglichen wird.

Als alternativen Ansatz schlagen Lee et al. [20] vor, ein Verzeichnis mit bekannten Abkürzungen zu verwenden. Bei jedem Vergleich von Strings wird

in diesem Verzeichnis „nachgeschlagen“, ob einer der Strings (bzw. ein darin enthaltener Teilstring) eine Abkürzung des anderen ist. Beschränkt man die Betrachtung von Strings auf ganze Wörter, so ließe sich dieser Vergleich effizient z. B. mittels Hashes implementieren. Die Erstellung und Pflege eines Abkürzungsverzeichnisses ist jedoch mit hohem Aufwand verbunden und in hohem Maße abhängig von der Anwendungsdomäne, so dass diese Lösung für praktische Verwendungen eine untergeordnete Rolle spielt[19].

Phonetische Metriken Die in Abschnitt 4.3 erläuterten Editierabstände betrachten Distanzen in der syntaktischen Repräsentation von Attributwerten. Daneben existieren auch phonetische Metriken, die versuchen, Distanzen in der Aussprache der Werte bestimmen. Ein prominentes Beispiel dieser Kategorie ist der Soundex-Algorithmus [25], der die englische Aussprache beliebiger Strings abzubilden versucht. Der Algorithmus berechnet für einen beliebigen String einen Code bestehend aus einem Buchstaben und drei Zahlen, wobei Strings gleicher Aussprache auf den gleichen Code abgebildet werden[20]. Bspw. ergibt sich für die Strings „Dickson“ und „Dixon“ der gleiche Code „D-250“.

Aufgrund seiner Einfachheit wird der Algorithmus in vielen Systemen eingesetzt. Gleichzeitig bietet er jedoch nur eine sehr grobe Approximation, die insbesondere viele ähnliche klingenden Begriffe nicht als solche erkennt. Ein grundlegendes Problem aller phonetischen Metriken ist darüber hinaus, dass die Aussprache eines Strings in hohem Maße von der zugrunde gelegten natürlichen Sprache abhängt. Im Fall des Soundex-Algorithmus, der auf die englische Sprache ausgelegt ist, werden z. B. deutsche Begriffe nur schlecht abgebildet. Daher existieren zahlreiche alternative phonetische Metriken. Einen Überblick gibt z. B. [26].

WHIRL Einen Information-Retrieval-basierten Ansatz zum Matching von Datensätzen stellt WHIRL (für „Word-based Heterogeneous Information Retrieval Logic“) dar [2]. WHIRL betrachtet einen Datensatz nicht als eine Menge von Attributwerten (vgl. Abschnitt 2.3), sondern als Dokumentenvektor im Sinne des Vektorraummodells. Sei $T = \{t_0, \ldots, t_N\}$ die Menge aller atomaren Terme[21], die in Attributwerten der betrachteten Datensätze vorkommen. Dann ist ein *Dokumentenvektor* ein Vektor reeller Zahlen $v \in \mathbb{R}^{|T|} = \big(w_0, w_1, ..., w_{|T|}\big)$, wobei w_i das Gewicht des i-ten Terms in v bezeichnet.

[19] Die Verwendung von Abkürzungsverzeichnissen ist allerdings sehr wohl praktikabel, wenn für die konkrete Anwendungsdomäne bereits ein solches Verzeichnis existiert, das eingesetzt werden kann. Dies ist z. B. für medizinische Anwendungen der Fall.

[20] Eine ausführliche Beschreibung des Soundex-Algorithmus findet sich z. B. im Anhang von [25].

[21] Zur besseren Anschaulichkeit verstehen wir unter Termen im Folgenden Wörter. Cohen [2] schlägt die Verwendung von Wortstämmen unter Einsatz von Porters Stemming-Algorithmus vor, um die Matching-Toleranz zu erhöhen. Alternativ kann als Term auch jede andere Aufteilung eines Strings herangezogen werden, z. B. Satzteile oder zusammenhängende Begriffe.

Das Gewicht berechnet sich nach dem in Information-Retrieval-Anwendungen verbreiteten TF-IDF-Gewichtungsschema aus der *Termhäufigkeit* (engl. Term Frequency) TF und der *Inversen Dokumenthäufigkeit* (engl. Inverse Document Frequency) IDF. Die Termhäufigkeit beschreibt, wie oft ein Term in einem Dokument auftritt. Aus der inversen Dokumenthäufigkeit ergibt sich die Seltenheit eines Terms t bezogen auf alle Dokumente, d. h. sie ist um so größer, je weniger Dokumente t enthalten. Für eine Menge von Dokumenten C (d. h. in diesem Kontext die Menge der betrachteten Datensätze) und die Teilmenge der t enthaltenden Dokumente $C_t \subset C$ berechnet sie sich als $IDF_t = \frac{|C|}{|C_t|}$. Die (nicht-normalisierte) Gewichtung eines Terms t ergibt sich dann zu

$$\hat{w}_t := \begin{cases} ((log(TF_{v,t}) + 1) \cdot log(IDF_t), & \text{falls } t \text{ im Dokument} \\ & \text{von } v \text{ enthalten ist} \\ 0, & \text{sonst} \end{cases} \tag{3}$$

Diese Gewichtung normalisieren wir auf das Intervall I_{01} und verwenden als Ähnlichkeitsmaß die Kosinusähnlichkeit. Diese ergibt sich aus dem Skalarprodukt der beiden Dokumentenvektoren normalisiert mit den Vektorlängen, entspricht geometrisch betrachtet also dem Kosinus des Winkels α der beiden Vektoren:

$$sim(v_1, v_2) = cos(\alpha(v_1, v_2))$$

Primärer Vorteil dieses Verfahrens ist, das die Reihenfolge der Terme keine Rolle spielt. Die Attributwerte „Müller, Hans“ und „Hans Müller“ hätten somit eine Ähnlichkeit von 1 (wenn wir die Interpunktion ignorieren). Allerdings berücksichtigt das Verfahren keine Abkürzungen. Eine Kombination des Konzepts mit anderen bereits diskutierten Metriken wäre daher vielversprechend.

4.4 Ähnlichkeit nominal skalierter Werte

Wie bereits angedeutet, ist der Vergleich nominal skalierter Werte wie etwa „Geschlecht“ deutlich komplexer als der Vergleich von Zeichenketten oder Kardinalwerten. Ist das Geschlecht bspw. durch String-wertige Konstanten {„männlich“; „weiblich“} repräsentiert, lassen sich u. U. Metriken für String-wertige Attribute anwenden. Dies funktioniert auch dann noch, wenn die Attributwerte in zwei Relationen unterschiedlich gewählt sind, solange die korrespondierenden Wert-Repräsentationen noch eine gewisse Ähnlichkeit aufweisen (z. B. {„männlich“; „weiblich“} vs. {„m“; „w“}). Spätestens, wenn die Werte keinerlei syntaktische oder lexikalische Ähnlichkeit mehr aufweisen, versagen diese Methoden jedoch. In obigem Beispiel könnte das Geschlecht etwa durch die Werte {0; 1} kodiert sein.

Für diesen Fall existieren daher spezielle Verfahren, die wir in diesem Abschnitt kurz erläutern. Dazu betrachten wir zunächst Ontologie-basierte Metriken und im Anschluss das Value Mapping als Prototyp eines auf statistischen Merkmalen beruhenden Verfahrens.

Ontologie-basierte Metriken Eine Möglichkeit, beim Vergleich von Datensätzen ihre Semantik mit einzubeziehen, besteht im Einsatz von Ontologien. Eine *Ontologie* ist ein formales Modell, welches die Objekte einer bestimmten Domäne und die Beziehungen zwischen diesen Objekten beschreibt.

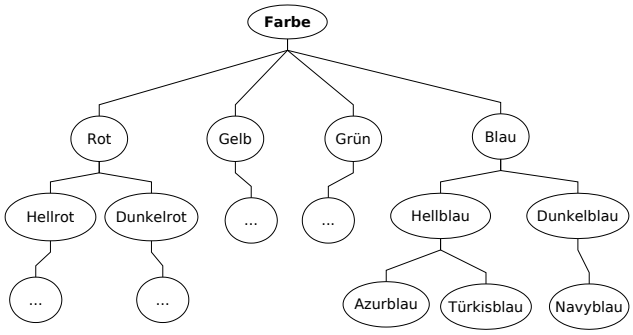

Abbildung 2. Ontologie am Beispiel von Farben (in Anlehnung an [27])

Kedad und Métais [27] beschreiben den Gebrauch der Subtyp-Beziehung („ist-ein“) und der sich daraus ergebenden Hierarchie von Begriffen, um Aussagen über die semantische Näher zweier Begriffe abzuleiten. Wie in Abb. 2 gezeigt, können wir die Begriffshierarchie als einen Baum auffassen, wobei Begriffe Knoten und Subtyp-Beziehungen Kanten darstellen. Im Beispiel ist also Azurblau ein Subtyp von Hellblau, Hellblau von Blau, Blau von Farbe, usw.

Wie nah zwei Begriffe verwandt sind, ist allerdings in hohem Maß vom jeweiligen Kontext abhängig. So kann in einer Anwendung beim Vergleich von Farbwerten die Unterscheidung nach Rot, Gelb und Grün ausreichen, während in einer anderen eine detaillierte Differenzierung einzelner Farbnuancen wichtig ist. Davon ausgehend schlagen [27] die Verwendung eines *Genauigkeitsgrads* (engl. Level of Accuracy) LA vor: LA ist eine beliebige, nicht leere Menge[22] disjunkter Klassen von Begriffen. Dabei verstehen wir unter einer *Klasse* T eines Begriffs t die Menge aller seiner (direkten und indirekten) Kindknoten vereinigt mit $\{t\}$. So enthält die Klasse des Begriffs Blau u. a. die Knoten Blau, Hellblau und Azurblau, nicht jedoch den Knoten Rot. Die Menge aller Begriffe (d. h. die Klasse des Wurzelknotens) bezeichnen wir als T_0.

Damit können wir die Ähnlichkeit zweier Begriffe t_1, t_2 als Zugehörigkeit zur selben Klasse definieren. Formal definieren wir also eine Ähnlichkeitsfunktion $sim : T_0 \times T_0 \longrightarrow I_{01}$ mit

[22] Eine einfachere Definition des Genauigkeitsgrads wäre die Angabe einer Ebene im Begriffsbaum, d. h. es würden alle Klassen auf dieser Ebene betrachtet (bspw. die Klassen Rot, Gelb und Blau). Durch die allgemeinere Definition als beliebige Menge besteht jedoch die Möglichkeit, bestimmte, im Kontext wichtige Äste stärker zu differenzieren, etwa Rot, Gelb, Hellblau, Dunkelblau.

$$sim(t_1, t_2) = \begin{cases} 1, \text{ falls } \exists T \in LA : t_1 \in T \wedge t_2 \in T \\ 0, \text{ sonst} \end{cases} \quad (4)$$

Um eine graduelle Beschreibung der Ähnlichkeit von Begriffen (im Sinne einer semantischen Distanz) vorzunehmen, kann die beschriebene Metrik um eine differenziertere Betrachtung der Beziehungen zwischen Begriffen erweitert werden. Hierfür existieren zahlreiche Ansätze (vgl. z. B. [27]), die jedoch außerhalb des Rahmens dieser Arbeit liegen.

Das beschriebene Ähnlichkeitsmaß ermöglicht den Ähnlichkeitsvergleich von Begriffen ausschließlich anhand ihrer Semantik, d. h. ohne auf eine Ähnlichkeit ihrer syntaktischen Repräsentation angewiesen zu sein. Allerdings ergeben sich verschiedene Probleme: Ontologien in der beschriebenen Form ermöglichen zwar das Matching zwischen Datensätzen, unterstützen jedoch nicht den Merging-Prozess (vgl. Kapitel 5). So gestaltet sich z. B. die Zusammenführung zweier äquivalenter Datensätze mit Farbangaben „Blau“ und „Azur“ schwierig.

Zudem bietet das Verfahren keine Toleranz gegenüber syntaktischen Fehlern oder fehlerhaften Zuordnungen in der Hierarchie. [27] führen in diesem Zusammenhang z. B. die Problematik des Zeitbezuges in sich schnell verändernden Anwendungsdomänen an. Auch der enge Bezug zu den jeweiligen Anwendungsdomänen allgemein und die Voraussetzung, dass eine Ontologie für die jeweilige Domäne existiert (sowie ggf. der Aufwand für deren Pflege und Wartung) stellen Nachteile dar. Ein Lösungsansatz für die letztgenannten Probleme stellt z. B. das Value Mapping dar, auf das wir im Folgenden eingehen.

Value Mapping Kang et al. [28] schlagen mit dem *Value Mapping* ein statistisches Verfahren vor, dessen Fokus auf der Abbildung von Attributwerten zwischen Relationen liegt. Im Kontext des Record Matchings können wir das Value Mapping allerdings als eine spezielle Distanz-Metrik betrachten, die sehr nützliche Eigenschaften ausweist.

Das Verfahren liefert eine Abbildungsvorschrift, die für einen beliebigen Attributwert in einer Relation seine Entsprechung in einer anderen Relation angibt und uns daher den Vergleichen der Attributwerte ermöglicht. Die *Erstellung* dieser Abbildungsvorschrift müssen wir in einem Cleaning-Prozess allerdings nur ein einziges Mal ausführen, so dass sie streng genommen in die Analysephase fällt. Zur besseren Vergleichbarkeit mit den anderen Verfahren zur Abbildung nominal skalierter Werte, diskutieren wir sie allerdings in diesem Kontext.

Interessant ist das Value Mapping, da es – im Gegensatz zu (nahezu) allen anderen Metriken und Matching-Verfahren allgemein – weder syntaktische noch lexikalische Ähnlichkeiten der betrachteten Datensätze voraussetzt und gleichzeitig nicht das Vorhandensein semantischer Modelle (wie etwa Ontologie-basierte Metriken) erfordert. Die Grundidee des Verfahrens ist es, anstatt Distanzen zwischen den konkreten Repräsentationen von Attributwerten (also z. B. Strings oder Zahlen) die Distanzen zwischen statistischen Merkmalen der Attributwerte zu betrachten. Dabei liegen die folgenden Annahmen zugrunde:

1. Es soll ein Record Matching (bzw. Matching von Attributwerten) auf Datensätzen vorgenommen werden, die sich in zwei Relationen T_1 und T_2 befinden.
2. Bezeichne A_T die Menge der Attribute einer Relation T; dann sei (z. B. durch ein vorausgegangenes Schema Matching) für A_{T_1}, A_{T_2} eine bijektive Abbildung m_A gegeben, die allen Attributen in A_{T_1} ein Attribut in A_{T_2} zuordnet. Anschaulich gesprochen wissen wir also, welche Spalten der beiden Tabellen sich jeweils entsprechen.
3. Für alle $a \in A_{T_1}, b \in A_{T_2}$ mit $b = m_A(a)$ existiere eine zweite bijektive Abbildung $m_v : F(a) \longrightarrow F(b)$, die jedem Wert im Wertebereich von a einen Wert im Wertebereich von b zuordnet. Die Menge dieser Abbildungen sei jedoch nicht bekannt. (Anschaulich: wir wissen, dass für jeden Attributwert in T_1 ein Attributwert in T_2 existiert, der diesem entspricht, kennen aber die Abbildung zwischen den Attributwerten nicht.)

Der Value-Mapping-Algorithmus errechnet nun in einer ersten Phase separat für T_1 und T_2 die Häufigkeit des Zusammentreffens[23] (engl. Co-Occurrence Frequency) von Attributwerten in einem Datensatz. Unter Zusammentreffen verstehen wir dabei, dass zwei Attributwerte v_0 und v_1 im selben Datensatz auftreten. Daraus ergibt sich die sog. *Co-Occurrence-Frequency-Matrix* $C = V \times V$, wobei V die Vereinigung der Wertebereiche aller Attribute einer Relation bezeichnet.

In einer zweiten Phase errechnet der Algorithmus ein Matching für die Attributwerte von T_1 und T_2. Eine einfache Möglichkeit, diese Berechnung durchzuführen, besteht darin, das Matching als ein Optimierungsproblem aufzufassen. Dabei minimieren wir die euklidische Distanz zwischen den Einträgen jeder Zeile von C. Anschaulich legen wir also solche Werte als äquivalent fest, deren Einträge in den beiden Matrizen sich am ehesten entsprechen. Das Ergebnis der zweiten Phase ist eine Abbildungsvorschrift, die für jeden Attributwert einer Relation beschreibt, welchem Wert in der jeweils anderen Relation dieser (wahrscheinlich) entspricht.

Nachteilig an diesem Verfahren ist die mangelnde Berücksichtigung fehlerhafter Daten. So könnte ein einzelner inkorrekter Datensatz ggf. zur falschen Abbildung eines oder mehrerer Attributwerte führen. Problematisch ist dabei insbes. die Fehlerfortpflanzung, da eine inkorrekte Abbildung das Matching zahlreicher Datensätze verfälschen kann.

5 Record Merging

Die Aufgabe des Record Mergings besteht in der Behandlung von Datensätzen, welche die Matching-Phase als äquivalent identifiziert hat. Wir diskutieren

[23] Die einfachste Möglichkeit, die Abbildung der Attributwerte vorzunehmen, wäre offensichtlich, ihre Häufigkeitsverteilung zugrunde zu legen. Diese Lösung würde allerdings nur dann zuverlässig funktionieren, wenn die Häufigkeiten der Attributwerte sich ausreichend stark unterscheiden, d. h. insbesondere nicht gleichverteilt sind. Um dieses Problem zu umgehen, betrachtet der Value-Mapping-Algorithmus paarweise Häufigkeiten. [29]

zunächst im folgenden Abschnitt die sich dabei ergebenden Probleme und klassifizieren mögliche Lösungsansätze. In den Abschnitten 5.2 und 5.3 gehen wir anschließend auf konkrete Verfahren zur Durchführung des Mergings ein.

5.1 Grundlagen

Das Ziel des Record Mergings lässt sich in zweierlei Weise interpretieren: Eliminierung von Duplikaten und Zusammenführung partieller Informationsquellen [30]. Traditionell ist es üblich, den Data-Cleaning-Prozess im Allgemeinen und insbesondere das Record Merging – in diesem Kontext daher auch oft einfach als Deduplication, Duplicate Elimination oder Purging bezeichnet – primär als die Entfernung (exakter oder nicht-exakter) Duplikate aus einer gegebenen Menge von Datensätzen zu betrachten (vgl. z. B. [31,19]). In der Regel erfolgt bei dieser Betrachtungsweise auch keine klare Differenzierung zwischen der Duplikat-Eliminierung und dem vorausgegangen Record Matching; vielmehr wird sie als das (triviale) Ende des Matching-Algorithmus angesehen.

Dieser Betrachtungsweise liegt die Annahme zugrunde, dass Duplikate nicht nur dasselbe Bezugsobjekt repräsentieren, sondern auch weitgehend den gleichen Informationsgehalt aufweisen, d. h. vor allem die gleiche Menge von Attributen enthalten. Insbesondere im Kontext von Data-Warehousing-Applikationen, die ganz gezielt versuchen, Informationen aus komplementären Quellen zusammenzuführen, ist diese Annahme jedoch fraglich. Betrachten wir bspw. „Person“ als ein Bezugsobjekt in unserer Miniwelt, dann könnten zwei imaginäre Datenquellen „Personaldatenbank“ und „Telefonverzeichnis“ die in Tabelle 1 und Tabelle 2 dargestellten Datensätze enthalten. Durch eine im Vorfeld durchgeführte Integration der beiden Schemata ergäben sich dann die in Tabelle 3 dargestellten Datensätze.

Personalnummer	*Nachname*	*Vorname*	*Gehalt*
0815	Maier	Hans	42.000

Tabelle 1. Personaldatenbank

Nachname	*Straße*	*Ort*	*Telefonnummer*
Maier	Lange Straße 42	Musterstadt	0123 45678

Tabelle 2. Telefonverzeichnis

	Pnr.	*Nachname*	*Vorname*	*Gehalt*	*Straße*	*Ort*	*Telefonnr.*
R_1	0815	Maier	Hans	42.000	—	—	—
R_2	—	Maier	—	—	Lange Straße 42	Musterstadt	0123 45678

Tabelle 3. Integrierte Datenbasis vor der Matching-Phase

Nehmen wir an, die Matching-Phase identifiziert die beiden Datensätze R_1 und R_2 als Duplikate (d. h. $R_1 \equiv R_2$). Dann ist die traditionelle Lösung, also eines der beiden Duplikate einfach zu entfernen, in diesem Szenario offensichtlich nicht angebracht, da ein Informationsverlust die Folge wäre. Betrachten wir

R_1 und R_2 hingegen als partielle Informationsquellen, so besteht die Aufgabe des Record Mergings darin, diese zu einem gemeinsamen Datensatz zusammenzuführen (engl. merge) und dabei – idealerweise – den Informationsgehalt der beiden einzelnen Quellen zu erhalten.

Eine zusätzliche Schwierigkeit ergibt sich in dem Fall, dass die Datensätze nicht konfliktfrei zusammengeführt werden können (wie dies in obigen Beispiel der Fall war), sondern sich teilweise widersprechen. Dieser Fall tritt dann auf, wenn beide Datensätze mindestens ein gemeinsames Attribut enthalten, dessen Wert sich in R_1 und R_2 unterscheidet. Wir gehen dabei davon aus, dass die Vorverarbeitungsphase (vgl. Abschnitt 3.2) die einzelnen Attributwerte auf der syntaktischen Ebene normalisiert hat, also z. B. die Strings „Lange Str.“ und „Lange Straße“ in ein einheitliches Format überführt wurden. Daher verstehen wir unter einem Widerspruch nur solche Attributwerte, die zwar auf der syntaktischen Ebene korrekt sind, jedoch inhaltlich in Konflikt zueinander stehen. Widersprüche existieren somit ausschließlich auf der *semantischen* Ebene. Ein möglicher Widerspruch in obigem Beispiel könnten die Angaben „Lange Straße 41“ (in R_1) und „Lange Straße 42“ (in R_2) sein.

Zusammenfassend unterscheiden wir daher drei mögliche Beziehungen zwischen äquivalenten Datensätzen:

1. *Identität*: R_1 und R_2 haben die gleiche Attributmenge und stimmen in allen Attributwerten überein.
2. *Komplementarität*: Die Attributmengen von R_1 und R_2 unterscheiden sich, d. h. es existiert mindesten ein nicht-gemeinsames Attribut.
3. *Konflikt*: R_1 und R_2 haben mindestens ein gemeinsames Attribut, dessen Wert sich unterscheidet.

Identität und Komplementärität sowie Identität und Konflikt schließen sich gegenseitig aus, Komplementärität und Konflikt können hingegen zusammen auftreten.

Die Behandlung identischer Datensätze ist trivial und erfolgt wie oben beschrieben durch das Entfernen jeweils aller zueinander äquivalenter Datensätze bis auf einen einzigen aus der Datenmenge. Auch Datensätze, die zwar komplementär aber nicht konfliktär sind, können wir sehr einfach zusammenführen, indem wir alle NULL-Werte[24] durch den Wert des entsprechenden Attributs eines jeweils äquivalenten Datensatzes ersetzen und die betroffenen Datensätze anschließend als identische Datensätze behandeln.

Im Folgenden konzentrieren wir uns daher auf die Behandlung konfliktärer Daten, die im Gegensatz zu den beiden beschriebenen Fällen nicht trivial ist. Wir unterscheiden dabei nach Bleiholder [32] drei Arten des Umgangs mit konfliktären Daten:

[24] Dieses Verfahren setzt voraus, dass wir NULL nicht als Wert sondern vielmehr als „fehlende Information“ behandeln. Ist dies nicht erwünscht, so sind alle komplementären Datensätze auch konfliktär und erfordern somit eine gesonderte Behandlung. In Abschnitt 5.3 gehen wir auf diesen Aspekt näher ein.

1. *Ignoranz*: Konflikte werden einfach ignoriert, d. h. konfliktäre Datensätze bleiben in der Zielrelation unverändert erhalten. Dieser Ansatz bietet offensichtlich keine befriedigende Lösung für Data-Cleaning-Probleme und ist daher uninteressant.
2. *Vermeidung*: Konflikte werden umgangen, indem konfliktäre Datensätze zwar zusammengeführt, die enthaltenen konfliktären Attribute jedoch entweder entfernt oder als solche gekennzeichnet werden. Verfahren dieser Kategorie diskutieren wir im folgenden Abschnitt.
3. *Lösung*: Konflikte werden gelöst, indem aufgrund der vorhandenen Attributwerte die Bestimmung eines semantisch sinnvollen, ggf. von den vorhandenen Attributwerten abweichenden Wertes versucht wird. Diese Variante bietet – zumindest theoretisch – die ideale Lösung des Merging-Problems. Allerdings ist ihre praktische Umsetzung schwierig, wie wir in Abschnitt 5.3 näher erläutern.

5.2 Konfliktvermeidende Verfahren

Im Folgenden diskutieren wir als beispielhafte Möglichkeiten des konfliktvermeidenden Mergings das Masking und die mengenbasierte Zusammenführung.

Masking Eine sehr einfache Möglichkeit zur Vermeidung eines Konflikts zwischen Attributen besteht offensichtlich darin, alle konfliktären Attributwerte zu entfernen, d. h. durch einen NULL-Wert zu ersetzen. Damit ist offensichtlich ein Informationsverlust verbunden, der qualitativ mit dem in Abschnitt 5.1 beschriebenen Purging-Verfahren vergleichbar ist. Zusätzlich verfälscht dieser Ansatz die Semantik des Attributs, da ein „tatsächlicher“ NULL-Wert (im Sinne von „Es wurde kein Wert angegeben“ bzw. „Es ist kein Wert verfügbar“) nicht zu unterscheiden ist von einem fehlerhaften und daher entfernten Wert.

Diese Verfälschung löst die *Maskierung* (engl. Masking) von Werten. Dabei werden konfliktäre Attribute mit einem speziellen Flag versehen („maskiert“) [6]. Zur Ermittlung des tatsächlicher Wert des Feldes können entweder konfliktlösende Verfahren (vgl. Abschnitt 5.3) oder mengenbasierte Merging-Verfahren (vgl. Abschnitt 5.2) genutzt werden. Bei Anfragen auf den Daten hat der Benutzer dann die Möglichkeit, maskierte Werte entweder zu ignorieren oder diese bewusst mit einzubeziehen. Allerdings verlagern wir damit das Problem z. T. nur aus dem Data-Cleaning-Prozess in die nachfolgende produktive Nutzungsphase des Datenbestandes, so dass diese Lösung unbefriedigend ist.

Mengenbasierte Zusammenführung Eine weitere Alternative zum Löschen oder (potentiell falschen) Zusammenführen von Datensätzen bzw. Attributwerten besteht darin, einen Datensatz nicht als einfaches Tupel, sondern selbst als Menge von Tupeln zu betrachten [6]. Im Unterschied zu konfliktignorierenden Verfahren (vgl. Abschnitt 5.1) halten wir dabei im System zusätzlich die Information vor, welche Datensätze äquivalent sind, d. h. eine solche Tupel-Menge bilden.

Da immer nur ein Teil der Attribute eines Datensatzes konfliktär sind, reicht es aus, nur Attributwerte (statt ganzer Datensätze) als Mengen zu betrachten. Einen solchen Ansatz beschreiben z. B. Menestrina et al. [33]. Bspw. könnte ein Attribut „Name“ statt eines atomaren Wertes „Maier“ eine Wertemenge {„Maier“, „Mayer“, „Meier“} enthalten, so dass ein Informationsverlust – zumindest bei der Speicherung der Daten – ausgeschlossen ist.

Zur Unterstützung dieser Ansätze muss die jeweilige Datenquelle (z. B. ein RDBMS) in beiden Fällen eine sehr komplexe Verwaltungslogik implementieren [6]. Das wesentliche Problem besteht allerdings darin, dass die Behandlung von Konflikten nicht während des Data Cleanings erfolgt, sondern – wie bei dem Masking-Verfahren – (explizit oder implizit) bei jeder einzelnen Anfrage, die nach der Säuberung auf den Daten ausgeführt wird. Dies erhöht in drastischer Weise die Komplexität und Fehleranfälligkeit von Anfragen und erreicht damit nicht die Ziele des Data Cleanings.

Der Attribut-basierte Ansatz birgt zusätzlich die Gefahr, dass implizit vorhandene Informationen verloren geht. Dies ist z. B. der Fall, wenn zwischen einzelnen Attributwerten Beziehungen bestehen. Verfügen bspw. zwei äquivalente Datensätze über die Attribute „Straße“ und „Hausnummer“ mit den Werten („Kurze Straße“, 1) bzw. („Lange Straße“, 4711), so könnte eine spätere Abfrage auf dem Ergebnisdatensatz aufgrund der fehlenden Beziehung der beiden resultierenden Attributmengen das Tupel („Kurze Straße“, 4711) zurückliefern. Dieses Ergebnis wäre völlig nutzlos, gleichzeitig aber nicht als semantisch inkorrekt zu erkennen.

5.3 Konfliktlösende Verfahren

Neben den beschriebenen konfliktvermeidenden Verfahren existieren auch eine Reihe von Ansätzen, die tatsächlich eine Lösung von Konflikten anstreben. Auf einige dieser Ansätze gehen wir in diesem Abschnitt ein.

Selektionsverfahren Selektionsverfahren versuchen, aus einer gegebenen Menge vorhandener Attributwerte (bzw. ganzer Datensätze) einen einzigen auszuwählen, der dann als korrekt angenommen wird. Eine einfache Realisierung besteht in einem Mehrheitsabstimmungsverfahren (engl. Voting), bei dem derjenige Attributwert als korrekt angenommen wird, der in der Mehrheit der beteiligten Duplikate auftritt [34]. Dies erfordert insbesondere, dass mindestens jeweils drei äquivalente Datensätze vorhanden sind, was die Verwendung dieses Ansatzes i. d. R. für praktische Anwendungen nahezu ausschließt.

Alternativ können wir statistische Methoden verwenden, um z. B. anhand eines Histogramms denjenigen Wert auszuwählen, der mit der höchsten Wahrscheinlichkeit auftritt oder die geringste Abweichung zu anderen Werten dieses Attributs aufweist (vgl. Abschnitt 3.2). Ein erweiterter Ansatz ist die gezielte Gewichtung von Datensätzen z. B. aufgrund von Fehlerwahrscheinlichkeiten. Einen vielversprechenden Ansatz dieser Art diskutieren wir am Ende dieses Abschnitts.

Aggregatfunktionen Aggregatfunktionen sind eine weitere Variante der Konfliktlösung. Im Gegensatz zu Selektionsverfahren wählen wir dabei nicht einen der in den Duplikaten auftretenden Attributwerte als den Korrekten aus, sondern errechnen anhand der Menge konfliktärer Werte einen neuen Wert. Dabei können wir sowohl generische Aggregatfunktionen (wie z. B. den Durchschnitt) anwenden [33] oder domänen- und anwendungsspezifische Aggregatfunktionen implementieren [6].

Generische Funktionen erfordern keinen zusätzlichen Implementierungsaufwand und sind unabhängig von konkreten Anwendungskontexten. Allerdings ist fraglich, inwiefern z. B. der (ganzzahlige) Durchschnitt eines Attributs „Alter“ einen sinnvollen Wert ergibt; ganz im Gegenteil dürfte die Anwendung solcher Aggregatfunktionen sogar in vielen Fällen zu definitiv inkorrekten Werten führen. Zudem ist die Anwendung bei String-wertigen Attributen problematisch.

Individuell für einen Anwendungskontext implementierte Aggregatfunktionen können beliebige Logik zur Bestimmung eines im Kontext sinnvollen bzw. wahrscheinlich korrekten Wertes enthalten. Allerdings ist ihre Umsetzung aufwendig und schränkt (offensichtlich) die allgemeine Verwendbarkeit des Verfahrens ein. Allgemein haben beide Verfahren den inhärenten Nachteil, dass die Entstehung von Werten nur schwer nachvollziehbar ist und fehlerhafte „Korrekturen“ daher ggf. nur schwer zu erkennen sind (vgl. Abschnitt 6.1).

Konfidenz-basierte Verfahren Menestrina et al. [33] schlagen einen Merging-Ansatz vor, der auf *Konfidenzen* (engl. Confidences), d. h. dem Vertrauen in die Korrektheit von Datensätzen, beruht. Die Konfidenz eines Datensatzes R schreiben wir als $R.c \in I_{01}$. Der Wert $R.c = 1$ drückt aus, dass wir die Korrektheit eines Datensatzes als sicher ansehen, während ein Datensatz mit $R.c = 0$ wertlos ist.

Die Konfidenzwerte von Datensätzen ergeben sich aus drei Faktoren: dem in der jeweiligen Datenquelle hinterlegten Wert, Anpassungen von Attributwerten während der Vorverarbeitung und beim Matching von Datensätzen. Der erste Faktor ist offensichtlich. Die Vorverarbeitung kann zu Konfidenzänderungen führen, da z. B. die Korrektur eines Ausreißers in den Attributwerten durch einen Durchschnittswert zur Verringerung der Konfidenz des betroffenen Datensatzes führt. Die Matching-Phase beeinflusst die Konfidenz, da die Duplikatidentifikation – außer im Fall einer Clerical Review – immer mit Grenzwerten arbeitet, bei deren Überschreiten wir zwei Datensätze als äquivalent annehmen. Dabei können wir z. B. der Ähnlichkeitsgrad der betroffenen Datensätze zur Berechnung der neuen Konfidenz heranziehen.

Das Modell von Menestrina et al. [33] beschränkt Konfidenzen auf Datensatzebene, um die Komplexität der dafür notwendigen Verwaltungsfunktionen in der Datenquelle zu reduzieren. Daraus ergibt sich aber ein gravierender Nachteil, der an folgendem Beispiel deutlich wird: Nehmen wir an, R_1 und R_2 sind die komplementären Datensätze aus Tabelle 3 und R_3 ist der sich aus der Zusam-

menführung von R_1 und R_2 ergebende Datensatz[25]. Weiter seien die folgenden Konfidenzwerte bekannt: $R_1.c = 0.8, R_2.c = 0.7$. Ziehen wir den ungewichteten Durchschnitt als Berechnungsvorschrift[26] heran, dann ergibt sich der Konfidenzwert von R_3 zu 0.56, d. h. $R_3.c < R_1.c$ und $R_3.c < R_2.c$. Wir verlieren folglich durch das Zusammenführen die Information, dass wir mit einer Sicherheit von 80% von der Richtigkeit z. B. des Vornamens überzeugt sind [33].

Zur Lösung dieses Problems schlagen Menestrina et al. vor, die ursprünglichen Datensätze (mit ihren jeweiligen Konfidenzwerten) beizubehalten und *zusätzlich* den zusammengeführten Datensatz in das Data Set aufzunehmen. Daraus ergeben sich wiederum eine ganze Reihe von Folgeproblemen bei Verwaltung der Äquivalenzinformation, der Behandlung von späteren Anfragen und der effizienten Durchführung der Merging-Prozesses, die [33] durch die Einführung verschiedener Hilfskonstrukte wie der sog. Dominierung von Datensätzen u. ä. zu beherrschen versuchen. Dadurch nimmt jedoch die Komplexität der nötigen Verwaltungslogik wieder drastisch zu, welche der Ansatz gerade zu reduzieren versucht (s. o.), so dass diese Lösung ungeeignet erscheint.

Eine alternative, grundsätzlichere Lösung des Problems besteht in der Einführung von Konfidenzen auf der Attributebene. Ein Datensatz nimmt also die Form $R = ((v_1; c_1), (v2; c2), \ldots, (v_n; c_n))$ an. Bei identischen Attributwerten können wir dann das Maximum der Konfidenzen der beiden ursprünglichen Werte in den Zusammengeführten Datensatz übernehmen, in obigem Beispiel also „Maier“.$c = 0.8$. Komplexer ist hingegen die Behandlung nicht-identischer Attribute (z. B. Telefonnummer „12345“ vs. „54321“). In diesem Fall können wir entweder den Wert mit der höheren Konfidenz (bzw. bei gleicher Konfidenz zufällig einen der Werte) übernehmen oder auf eine mengenbasierte Attributverwaltung (vgl. Abschnitt 5.2) zurückgreifen. Die Selektion eines Wertes anhand der Konfidenz ist bereits ein weitreichender Ansatz, der eine genaue und gut nachvollziehbare Zusammenführung ermöglicht. Mengenbasierte Attribute werden durch das zusätzliche Ordnungskriterium in Form der Konfidenz ein Lösungsansatz noch größerer Mächtigkeit für das Record Merging, der jedoch auch eine komplexere Verwaltungslogik bedingt, wie bereits in Abschnitt 5.2 diskutiert.

Ein Sonderfall bei Konfidenzen auf Attributebene stellen NULL-Werte dar. Behandeln wir sie wie „normale“ Attributwerte, so sind NULL-Werte und Nicht-NULL-Werte konfliktär. Es ergäbe sich also in obigem Beispiel als Wert mit der höchsten Konfidenz für den Vornamen der Wert NULL, obwohl ein definierter Wert („Hans“) verfügbar wäre. In dem Fall, dass ein NULL-Wert die Semantik „Nicht zutreffend“ hat (etwa bei der Telefonnummer, falls die betreffende Person

[25] Für dieses Beispiel unterstellen wir, dass bei der Zusammenführung NULL-Werte durch den jeweils definierten Wert ersetzt werden, also R_3 als Vorname z. B. den Wert „Hans“ enthält.

[26] Das Problem sich ändernder Attribut-Konfidenzen aufgrund der zu geringen Granularität tritt auch bei jeder anderen Berechnungsvorschrift für c auf. Wir verwenden daher zur besseren Anschaulichkeit eine sehr einfache Variante.

keinen Anschluss besitzt), ist dies sinnvoll, d. h. es sollten auch NULL-Werte als „Wert“ mit einer individuellen Konfidenz verwaltet werden. Für die alternative Semantik „Wert nicht vorhanden“ besteht eine mögliche Lösung darin, NULL-Werte per se mit der Konfidenz 0 zu versehen, d. h. NULL- und Nicht-NULL-Werte als komplementär zu betrachten. Um beide Fälle zu behandeln, ist bspw. eine Verwaltung der NULL-Semantik als Teil der Schema-Information der Datenquelle denkbar.

6 Qualitätskriterien und -messung

Die Qualität von Data-Cleaning-Verfahren hängt von einer Vielzahl von Kriterien ab. In vielen Fällen (vgl. z. B. [19,2,35]) wird die Qualität eines Verfahrens (ausschließlich) an Qualität der resultierenden Ergebnismenge festgemacht. Obwohl diese offensichtlich von hoher Bedeutung ist, sind für die praktische Eignung von Data-Cleaning-Prozesse noch eine Reihe weiterer Faktoren wichtig, welche den Transformations*prozess* als solchen betreffen. Daher gehen wir im folgenden Abschnitt zunächst auf die wesentlichen prozessbezogenen Qualitätskriterien ein und diskutieren dann in Abschnitt 6.2 ergebnisbezogene Kriterien.

6.1 Prozessbezogene Kriterien

Wichtige prozessbezogene Kriterien sind u. a. die Laufzeitkomplexität und Laufzeit, der Grad der benötigten Benutzerinteraktion, die Abhängigkeit von einer bestimmten Anwendungsdomäne und die Verfolgbarkeit von Transformationen. Daneben ist auch die Wahrung der Privatsphäre ein wichtiges Kriterium, das keinen technischen, sodern vielmehr einen ethischen Bezug hat. Im Folgenden gehen wir auf diese Kriterien kurz ein.

Algorithmische Komplexität Data-Cleaning-Prozesse säubern i. d. R. sehr große Datenmengen (in der Größenordnung von $> 10^9$ Datensätzen), wobei die Ressourcen Zeit und Rechnerkapazität nur in klar begrenztem Maße zur Verfügung stehen. Zudem nehmen die Datenmengen (z. B. im Kontext von Data-Warehousing-Anwendungen) kontinuierlich zu, so dass die Algorithmen in ausreichendem Maße skalieren müssen.

Die Vorverarbeitung und das Merging können wir in annähernd linearer Zeit, d. h. in $O(N)$, durchführen (vgl. Abschnitt 3.2), wobei N die Anzahl der zu säubernden Datensätze beschreibt. Problematisch ist hingegen insbesondere die Matching-Phase, die im einfachsten Fall (Vergleich aller Datensätze) $\frac{n^2-n}{2}$ Vergleiche erfordert, also in $O(N^2)$ liegt. Selbst bei optimierten Verfahren, die z. B. den Suchraum einschränken (vgl. Abschnitt 4.2), liegt das Matching noch in $O(NlogN)$ und dominiert damit den gesamten Cleaning-Prozess. Vor diesem Hintergrund sind inkrementelle Verfahren von besonderer Bedeutung, da sie eine wiederholte Säuberung von Datensätzen zu vermeiden suchen.

Um die Kosten eines Verfahrens exakt zu bestimmen, müssten wir neben der algorithmischen Komplexität auch die Zugriffskosten auf Speicherungseinheiten (Hauptspeicher, Externspeicher) berücksichtigen, was im Umfang dieser Arbeit jedoch nicht möglich ist. Ansätze zur differenzierten Kostenbetrachtung (bezogen auf Matching-Verfahren) finden sich z. B. in [8].

Laufzeit Neben der Laufzeitkomplexität ist auch die tatsächliche Laufzeit relevant, insbesondere wenn harte zeitliche Schranken bestehen. Dies ist z. B. der Fall, wenn für den Versand von Werbebriefen durch ein Unternehmen zu einem im Voraus definierten Zeitpunkt eine gesäuberte Datenbasis benötigt wird. Manche Algorithmen (vgl. z. B. Abschnitt 4.2) bieten bei durchschnittlicher Komplexität einen hohen Spielraum für Parallelisierung und somit Senkung der Laufzeit bei Inkaufnahme eines höheren Hardwarebedarfs. Ein anderer Ansatz besteht in diesem Zusammenhang in der Wiederverwendung von Teilergebnissen, um die zu verarbeitende Datenmenge zu reduzieren und so die Laufzeit zu reduzieren (vgl. Abschnitt 4.2).

Benutzerinteraktion Eine Interaktion mit dem Benutzer erfordern Data-Cleaning-Verfahren traditionell u. a. bei der Definition des Workflows, bei der Abbildung von Attributwerten zwischen Datenquellen und bei der Entscheidung über Datensätze in der Menge möglicher Links PL („Clerical Review“). Alle genannten Bereiche sind mit hohem Zeitaufwand und/oder fachlichen Anforderungen an die Benutzer verbunden.

Besonders problematisch ist der Fall der Clerical Review, da der Aufwand hierbei nicht konstant ist, sondern von der Anzahl der Datensätze in PL abhängt. Aufgrund der großen Datenmengen und der sich daraus ergebenden großen Mächtigkeit von PL ist eine manuelle Bearbeitung dieser Datensätze nicht zumutbar bzw. schlicht unmöglich [2]. Record-Matching-Verfahren müssen daher zwingend die Benutzerinteraktion auf ein Minimum beschränken, um praktisch anwendbar zu sein. Nicht zu vermeidende Interaktionen sollten von entsprechenden Werkzeugen unterstützt werden, um effizientes Arbeiten zu ermöglichen.

Domänenbezug Viele Aspekte des Data Cleanings hängen in hohem Maße von der jeweiligen Anwendungsdomäne ab. Dabei existiert ein Trade-Off zwischen einer möglichst hohen Effektivität und Effizienz des Verfahrens einerseits (welche eine starke Ausrichtung auf eine Anwendungsdomäne begünstigt) und einer möglichst hohen Wiederverwendbarkeit und geringem Customizing-Aufwand (idealerweise also keinem Anwendungsbezug) andererseits.

Grundsätzlich können wir einen Domänenbezug auf der Implementierungs- und auf der Konfigurationsebene unterscheiden. In ersterem Fall implementieren wir den Workflow bzw. die darin verwendeten Verfahren spezifisch für einen konkreten Anwendungskontext. Dies ermöglicht einerseits eine effektivere und

effizientere Implementierung des Cleaning-Workflows, reduziert aber Offensichtlich die Wiederverwendbarkeit auf ein Minimum. Ein Beispiel für einen solchen Ansatz ist die Säuberung von bibliographischen Daten in [34].

Liegt der Anwendungsbezug auf der Konfigurationsebene, sind die verwendeten Verfahren generisch, werden aber über Parameter auf die jeweilige Anwendung justiert. Beispiele für solche Parameter sind die Grenzwerte für Matches und Nicht-Matches (vgl. Abschnitt 6.2) und Regelwerke für die Distanzberechnung zwischen Datensätzen. Insbesondere die manuelle Erstellung dieser Regelwerke erfordert ebenfalls einen hohen personellen Aufwand und zudem fachlich qualifiziertes Personal. Aktuelle Verfahren nutzen daher verstärkt Ansätze des Maschinenlernens, die aus einer Menge von Trainingsdaten automatisch Regelwerke generieren.

Data Lineage Ein weiteres wichtiges Kriterium für Cleaning-Prozesse ist *Data Lineage* (dt. etwa Abstammung, Herkunft von Daten). Darunter verstehen wir die Historie der Transformationsoperationen, die zu dem aktuellen Zustand eines Datensatzes geführt haben. Zum einen ist Lineage wichtig, um die Verlässlichkeit von Daten zu beurteilen. Dabei besteht ein direkter Zusammenhang mit den in Abschnitt 5.3 diskutierten Konfidenzen von Datensätzen.

Zum anderen ist es nur mit Lineage-Informationen möglich, die Qualität des Transformationsprozesses wirksam zu verbessern. Ein wichtiger Aspekt der Qualitätsverbesserung ist insbesondere auch die Analyse ggf. durch den Cleaning-Prozess verursachter Fehler und ihrer Ursachen, die nur mit Informationen über die Transformationshistorie möglich ist.

Eine ausführliche Diskussion des Lineage-Problems und mögliche Techniken zum Verfolgen von Änderungen durch die verschiedenen Phasen finden sich in [36].

Wahrung der Privatsphäre Ein weiteres wichtiges Kriterium – das im Gegensatz zu den bisher genannten nicht primär technischer Natur ist, sondern vielmehr eine ethische Dimension beschreibt – betrifft die Säuberung personenbezogener Daten: Wir müssen berücksichtigen, inwiefern die Privatsphäre der durch die Datensätze repräsentierten Personen durch den Data-Cleaning-Prozess verletzt wird. Besonders problematisch ist dieser Aspekt in den Phasen des Matchings und Mergings von Datensätzen. Führen wir bspw. eine Mitarbeiterdatenbank mit einem anonymisierten Bestand medizinischer Datensätze zusammen, so werden Datensätze ggf. „deanonymisiert“, was eine schwere Beeinträchtigung der Privatsphäre der betroffenen Personen bedeutet.

Dabei müssen wir je nach Anwendungskontext unterscheiden, ob eine Wahrung der Privatsphäre gewünscht ist (z. B. bei der Säuberung medizinischer Daten), oder ob dies gerade nicht der Fall ist (etwa wenn eine Finanzbehörde oder eine Versicherung eine Datensäuberung im Rahmen von Fahndungsaktivitäten durchführt). Zudem ist es schwierig, Metriken zur Messung dieses Kriteriums zu finden. Allerdings existieren verschiedene Ansätze, wie wir eine Säuberung von

Daten ohne Deanonymisierung technisch umsetzen können. Diese Ansätze werden oft auch als *blinde Säuberung* bzw. blindes Matching bezeichnet und basieren i. d. R. auf entsprechenden Verfahren zum Datenschutz in statistischen Datenbeständen. Die Erläuterung konkreter Ansätze liegt außerhalb des Umfangs dieser Arbeit; beispielhaft sei jedoch auf die Beschreibung solchen Verfahrens in [37] verwiesen. Eine umfassende Diskussion der allgemeinen Problemstellung findet sich z. B. in [38].

6.2 Ergebnisbezogene Kriterien

In diesem Abschnitt diskutieren wir verschiedene Kriterien zur Bewertung der Qualität der aus dem Data-Cleaning-Prozess resultierenden Datenmenge. Der Schwerpunkt liegt dabei auf der Bewertung des durchgeführten Record Matching, da diesem i. d. R. die höchste Bedeutung zukommt. Damit bewerten wir implizit auch die Qualität der vorverarbeitenden Schritte, die indirekt (z. B. durch Behebung typographischer Fehler) den Matching-Prozess beeinflussen, für die jedoch keine keine dedizierten Metriken existieren. Für die Zusammenführung von Datensätzen existieren bisher nur sehr rudimentäre Ansätze wie die Übereinstimmungsrate, auf die wir zuerst eingehen.

Übereinstimmungsrate Ein einfaches Maß für die Qualität eines zusammengeführten Datensatzes R_U ist die *Übereinstimmungsrate* mit den Ursprungsdatensätzen $R_1, \ldots, R_n$ (engl. Source Match Ratio) SMR, die sich aufgrund der Zahl jeweils übereinstimmender und nicht-übereinstimmender Attributwerte errechnet [34]. Stimmt R_U bspw. in vier von sieben Attributen mit R_1 überein, so ergibt sich die Übereinstimmungsrate als $SMR = \frac{a_m}{a_{ges}} = \frac{4}{7}$, wobei a_m die Anzahl übereinstimmender Attribute und a_{ges} die Gesamtzahl der Attribut in R_U ist.

Allerdings lässt dieses Maß nur indirekt auf die Qualität der Zusammenführung schließen und beschreibt vielmehr deren Schwierigkeit. Selbst bezogen auf alle jeweils in R_U zusammengeführten Quelldatensätze ließe sich anhand der Übereinstimmung lediglich ein Durchschnittswert errechnen, der keine sinnvollen Aussagen über deren Qualität erlaubt.

Genauigkeit Die *Genauigkeit* (engl. Accuracy) gibt an, wie zuverlässig ein Verfahren Duplikate erkennt. Sie errechnet sich aus der Anzahl fälschlich erkannter Links („False Positives“) $|FP|$ und der Anzahl fälschlich erkannter Nicht-Links („False Negatives“) $|FN|$, die wir in Relation zur Anzahl tatsächlich korrekter Klassifizierungen – also der Summe von „True Positives“ und „True Negatives“ $|TP| + |TN|$ – setzen [35]:

$$acc = \frac{|TP| + |TN|}{|TP| + |FP| + |TN| + |FN|}$$

Diese Berechnung unterstellt eine Gleichwertigkeit von False Positives und False Negatives, die in vielen Anwendungskontexten jedoch nicht gegeben ist[27]. Daher können wir die Formel optional um eine relative Gewichtung („Badness“) von $|FP|$ bzw. $|FN|$ erweitern, um diese kontextspezifisch unterschiedlich zu bewerten [39].

Problematisch bei dieser Kennzahl (und nahezu allen anderen) ist die Voraussetzung, dass wir die Anzahl korrekter Links in der Ausgangsdatenmenge kennen müssen. Denn nur eine Datenmenge, in der $|L|$ exakt bestimmt ist – auch als „Gold Standard Set“ bezeichnet –, lässt präzise Aussagen über Abweichungen zu [39]. Da eine manuelle Klassifizierung von Datensatzpaaren als Link bzw. Non-Link i. d. R. nicht möglich ist (vgl. Abschnitt 6.1), müssen wir Testdaten mit einer vorgegebenen Menge an Links generieren.

Durch die Notwendigkeit von Testdaten scheiden auf $|L|$ basierende Metriken für die Bewertung von Data-Cleaning-Prozessen auf realen Daten allerdings aus. Zudem besteht bei Testdaten die Gefahr, dass diese entweder nicht repräsentativ für die jeweilige Anwendung sind oder (bewusst oder unbewusst) speziell auf ein bestimmtes Verfahren oder Werkzeug hin optimiert werden. Auf Möglichkeiten zur Generierung geeigneter Testdaten gehen z. B. Christen et al. [40] ein.

Ein weiterer Nachteil der Genauigkeit als Qualitätskriterium ist die Dominanz der True Negatives bei der Berechnung: in praktischen Anwendungen hat $|TN|$ i. d. R. einen sehr hohen Wert, so dass sich selbst dann ein hoher Genauigkeitswert ergibt, wenn die Anzahl korrekt erkannter Links ($|TP|$) gegen Null konvergiert [35].

Precision und Recall Neben der Genauigkeit finden Information-Retrieval-basierte Qualitätsmetriken häufig Anwendung (vgl. z. B. [22,2]). Die beiden wesentlichen Kennzahlen hierbei sind *Precision* und *Recall*. Im ursprünglichen Kontext beschreibt Recall die „Vollständigkeit“ eines Suchergebnisses, d. h. den Anteil existierender relevanter Dokumente, der im Suchergebnis enthalten ist, und Precision den Anteil relevanter Dokumente in der Ergebnismenge.

Bezogen auf das Record Matching beschreibt die Precision *prec* den Anteil tatsächlicher Links (Matches) an allen als Link klassifizierten Datensatzpaaren und Recall den Anteil korrekt klassifizierter Links an der Gesamtheit der vorhandenen Links [35]:

$$prec = \frac{|TP|}{|TP| + |FP|} \qquad rec = \frac{|TP|}{|TP| + |FN|}$$

Die Precision entspricht dabei der Genauigkeit (vgl. Abschnitt 6.2), wenn die False Negatives mit 0 gewichtet werden. Allerdings besteht durch das Fehlen

[27] So spielt im Beispiel der Werbesendungen die Vermeidung unnötiger Kosten durch mehrfache Zustellung (d. h. eines hohen $|FN|$) vermutlich eine höhere Rolle, als der Verlust einzelner Datensätze. Im Zusammenhang mit medizinischen Daten ist hingegen das Nicht-Erkennen eines Duplikats wahrscheinlich eher tolerabel, als das fehlerhafte Zusammenführen zweier Nicht-Duplikate.

von $|TN|$ in beiden Formeln, bei Precision und Recall nicht das Problem der Verfälschung durch eine Dominanz der True Negatives.

Zwischen den beiden Kriterien besteht ein gegenläufiger Zusammenhang, d. h. es existiert ein Trade-Off zwischen Precision und Recall [35]. Dieser Zusammenhang ist nützlich, da wir mit seiner Hilfe die Qualität eines Matching-Verfahrens in Abhängigkeit von dem gewählten Grenzwert t zur Unterscheidung von Links und Non-Links (vgl. Abschnitt 4.1) modellieren können[28]. Als Werkzeug dienen uns dabei die im Information Retrieval verbreiteten *Precision Recall Curves*, welche für regelmäßige Intervalle von Recall-Werten (z. B. $0; 0.1; 0.2; \ldots; 1$) die jeweilige Präzision eines Verfahrens approximieren. Ersetzen wir den Recall-Wert durch t, so erhalten wir eine Kurve, welche die Präzision eines Verfahrens in Abhängigkeit von dem verwendeten Grenzwert beschreibt [39].

Weitere Kriterien Neben den diskutierten Kriterien der Ergebnisqualität existieren zahlreiche weitere, die jeweils bestimmte Aspekte des Data-Cleaning-Prozesses (bzw. des Record Matchings) abbilden. Dazu zählen z. B. das *F-Measure* (das harmonische Mittel von Precision und Recall, das auf die Ermittlung eines guten Grenzwertes t abzielt), die *Reduktionsrate* (engl. Reduction Ratio, zur Bewertung von Blocking-Mechanismen) und die *Spezifizität* (engl. Specificity, analog zur Präzision nur bezogen auf TN). Einen guten Überblick gibt [35].

7 Frameworks und Werkzeuge

Im Umfeld des Data Cleanings existieren zahlreiche Werkzeuge und verschiedene Frameworks zur Entwicklung von Data-Cleaning-Lösungen sowohl im wissenschaftlichen als auch im kommerziellen Umfeld. In diesem Kapitel gehen wir beispielhaft auf einige dieser Lösungen ein, wobei wir zunächst in Abschnitt 7.1 allgemeine Cleaning-Frameworks und frei verfügbare Werkzeuge diskutieren. In Abschnitt 7.2 gehen wir dann auf kommerzielle Data-Cleaning-Werkzeuge ein und betrachten kurz die wirtschaftliche Bedeutung des Data Cleanings.

7.1 Frameworks und freie Werkzeuge

In diesem Abschnitt geben wir einen kurzen Überblick über Frameworks und frei verfügbare Softwaresysteme zum Data Cleaning. Unter einem Framework[29]

[28] In vielen Fällen werden bei der Bewertung eines Verfahrens nur die Werte für ein *optimales* t angegeben, so dass faktisch eine vergleichbare Bewertung nicht möglich ist. Zudem wird dabei i. d. R. unterstellt, dass ein optimales t für die gegebene Datenmenge bekannt ist, was jedoch in den wenigsten Kontexten der Fall sein dürfte [39].

[29] Allerdings verzichten im Folgenden auf eine klare Abgrenzung zwischen den Begriffen Framework und Architektur, da sie für die dargestellten Zusammenhänge nicht

verstehen wir ein erweiterbares und anpassbares System kollaborierender Softwareeinheiten, das für eine allgemeine, übergeordnete Aufgabenstellung Kernfunktionalitäten mit entsprechenden Bausteinen bereitstellt [41]. Diese gehen in den meisten Fällen auf wissenschaftliche Arbeiten und Projekte zurück und haben eine Reihe interessanter Impulse gegeben. Allerdings haben diese Projekte darüber hinaus bisher nur in wenigen Fällen praktisch verwertbare Systeme hervorgebracht.

AJAX AJAX[30] [9] ist ein Framework zur Entwicklung von Data-Cleaning-Lösungen. Der Fokus liegt dabei auf der klaren Trennung zwischen einer logischen Ebene (Workflow-Spezifikation) und einer physischen Ebene (Implementierung), um die Austauschbarkeit von Algorithmen zu gewährleisten und die Ergebnisqualität von der Qualität des Prozesses (insbes. seiner Performanz) zu entkoppeln [36].

Das Framework definiert vier logische Grundtransformationen („Mapping“, „Clustering“, „Matching“, „Merging“), welche den gesamten Data-Cleaning-Prozess abdecken und in einer eigenen Sprache spezifiziert werden. Die Sprache ist eine SQL99-Erweiterung, die weitgehend deklarativ aufgebaut ist. Die Implementierung (physische Ebene) der zugrunde liegenden Verfahren (z. B. Normalisierungsalgorithmen oder Ähnlichkeitsmetriken) erfolgt allerdings programmatisch in Benutzer-definierten SQL-Funktionen (User-defined Functions).

AJAX bietet zudem einen Exception-Mechanismus, der an Programmiersprachen wie z. B. Java angelehnt ist. Damit ermöglicht das Framework, dass der Transformationsprozess auch fortgesetzt werden kann, wenn bei der Transformation einzelner Datensätze Fehler auftreten. Vorhergesehene Exceptions können programmatisch behandelt werden („catch“-Semantik), während nicht-vorhergesehene Exceptions zur einer Clerical Review der betroffenen Datensätze nach Abschluss des Transformationsprozesses führen.

Potter's Wheel Potter's Wheel [10] – auch bekannt als „Berkeley A-B-C“ – ist der Forschungsprototyp einer Data-Cleaning-Software, deren Schwerpunkt auf der interaktiven Spezifikation des Transformations-Workflows liegt. Das primäre Ziel ist es, die Spezifikation möglichst einfach, d. h. insbesondere ohne explizite Programmierung oder die Angabe regulärer Ausdrücke, zu ermöglichen.

Das System verwendet eine an Tabellenkalkulationen angelehnte Oberfläche, die dem Benutzer einen beispielhaften Ausschnitt aus dem zu reinigenden Datenbestand zeigt. Der Benutzer kann auf die verschiedenen Spalten vordefinierte Operationen (z. B. Split) anwenden, deren Ergebnisse direkt visualisiert werden („immediate Feedback“). Liefert eine Operation unerwünschte Resultate, so

wesentlich ist. Der Vollständigkeit halber sei aber darauf hingewiesen, dass es sich zumindest bei dem in Abschnitt 7.1 beschriebenen Ansatz streng genommen um eine Architektur im Sinne der Definition in [41] und nicht um ein Framework handelt.

[30] Nicht zu verwechseln mit der bei Web-Anwendungen verbreiteten „Asynchronous-JavaScript-and-XML“-Technologie, die ebenfalls mit AJAX abgekürzt wird.

kann der Benutzer sie über eine „Undo“-Funktion rückgängig machen. Darüber hinaus ermöglicht Potter's Wheel die Angabe individueller Formatierungen (sog. Domänen) für Attribute. Hierfür gibt der Benutzer dem System Beispiele vor, aus denen es entsprechende Pattern ableitet. Datensätze, die nicht den Formatierungen entsprechen, markiert das System, um dem Benutzer die Spezifikation zu erleichtern.

Unter der Oberfläche stellt Potter's Wheel im Wesentlichen ein Framework dar, das keine konkreten (bzw. nur beispielhafte Implementierungen) der eigentlichen Data-Cleaning-Phasen liefert, sondern lediglich Schnittstellen definiert. Für die praktische Verwendbarkeit müssten daher die eigentlichen Matching-, Analyse- und Merging-Operationen implementiert werden. Nachteilig dabei ist, das die Oberfläche dem Benutzer keine Hilfestellung bei der Auswahl ggf. vorhandener, alternativer Algorithmen z. B. zur Durchführung des Matchings gibt. Fraglich ist zudem, ob die Mächtigkeit des Spezifikationsansatzes auch für komplexe Cleaning-Prozesse ausreicht. Die Kombination der Techniken immediate Feedback, Undo und Spezifikation durch Beispiel stellt jedoch eine interessante Variante der Workflow-Spezifikation dar, die in anderen Systemen weitgehend vernachlässigt wird.

Weitere Frameworks Neben den beiden beschriebenen Ansätzen existieren eine Reihe weiterer frei verfügbarer Frameworks und Systeme, die zumeist ebenfalls aus wissenschaftlichen Arbeiten entstanden sind. Interessant ist z. B. TAILOR [42], das von den Autoren auch als „Record Linkage Toolbox“ bezeichnet wird . Es vereinigt eine generische Schichtenarchitektur für Record-Matching-Systeme mit einer Bibliothek verschiedener Implementierungen (u. a. von Ähnlichkeitsmetriken, Blockingverfahren und Entscheidungsmodellen). Allerdings wird das Projekt schon seit einigen Jahren nicht mehr aktiv betrieben.

Eines der wenigen aktiven Projekte hingegen ist Febrl („Freely extensible biomedical record linkage“) [18], das die Entwicklung einer Open-Source-Lösung für das Data Cleaning verfolgt. Das System unterstützt in der aktuellen Version u. a. eine auf Hidden-Markov-Models basierende Normalisierung und Sorted-Neighbourhood-Matching (vgl. Abschnitt 4.2). Ein besonderes Augenmerk liegt zudem auf der Parallelisierbarkeit des Systems.

Einen umfassenderen Überblick über Data-Cleaning-Frameworks und insbes. einen Vergleich ihrer Fähigkeiten gibt z. B. [6].

7.2 Kommerzielle Werkzeuge

Neben den diskutierten Frameworks existieren auch zahlreiche kommerzielle Data-Cleaning-Lösungen. Viele dieser Lösungen umfassen nur Teilaspekte des Cleaning-Prozesses, wobei üblicherweise eine der drei folgenden Dimensionen im Fokus liegt:

– *Anwendungskontext*: Data Cleaning ist ein Problem das in vielen Anwendungsbereichen auftritt (vgl. Kapitel 1). Für nahezu jeden dieser Bereiche

existieren Speziallösungen, die genau auf die jeweiligen Anforderungen zugeschnitten sind, jedoch im Vergleich zu allgemeinen Cleaning-Werkzeugen eine geringere Mächtigkeit aufweisen. Beispiele für ein solche Werkzeuge sind die CRM-Module von Trillium Software und Omikron Data Quality, die speziell auf die Säuberung von Customer-Relationship-Management-Datenbanken im Marketingumfeld ausgelegt sind.
- *Datenart*: Die Säuberung einer definierten Art von Daten ermöglicht die Verwendung von Domänenwissen z. B. in Form von Ontologien, Häufigkeitsverteilungen oder der Struktur von Datensätzen. Sehr verbreitet sind insbesondere Werkzeuge zur Säuberung von Adressdaten, z. B. QuickAddress (QAS), AddressDoctor (Platon Data Technology) oder Data Quality Server (GlobalAddress).
- *Phase des Cleaning-Prozesses*: Verbreitet in dieser Kategorie sind insbesondere Werkzeuge für Teilbereiche der Datenanalyse wie z. B. WizRule (WizSoft) zur Ableitung von Regeln (vgl. Abschnitt 3.1) und Erkennung von Outliern.

Neben diesen speziellen Lösungen haben sich in den vergangen Jahren zahlreiche Komplettlösungen für das Data Cleaning herausgebildet. Insbesondere haben zahlreiche Zusammenschlüsse und Übernahmen von Unternehmen eine Konsolidierung vieler Speziallösungen zu umfassenden Produkt-Suiten bewirkt[31]. Auffällig ist dabei, dass insbesondere die führenden Unternehmen im Datenbankbereich (etwa IBM, Oracle, Microsoft) ihr Produktportfolio im Data-Cleaning- bzw. allgemein im Information-Integration-Sektor stark ausgebaut haben.

Ein weiterer wesentlicher Einflussfaktor sind Werkzeuge für ETL-Prozesse[32]. Traditionell bieten sie nativ nur sehr beschränkte Data-Cleaning-Möglichkeiten (z. B. durch String-Matching). Interessant sind ETL-Werkzeuge für Data-Cleaning-Anwendungen allerdings insofern, als sie i. d. R. umfangreiche Engines zur Unterstützung von Transformations-Workflows bereitstellen, welchen den Aufruf dedizierter Werkzeuge oder Bibliotheken im Rahmen komplexer Transformations-Workflows ermöglichen [4]. Sie lassen sich daher mit spezialisierten Cleaning-Werkzeugen zu Mächtigen Data-Cleaning-Lösungen kombinieren. Zudem erweitern die führenden Hersteller ihre Produkte inzwischen zunehmend zu vollwertigen Data-Cleaning-Lösungen, so dass die Abgrenzung zwischen ETL- und Data-Cleaning-Werkzeugen verschwimmt.

Beispiele für umfassende Data-Cleaning-Produkte, die überwiegend auf ETL-Prozessen basieren, sind u. a.

- WebSphere Information Integration (IBM)

[31] So entstand bspw. die derzeitige IBM-Lösung durch Übernahme der Firma Ascential Software, die ihrerseits zuvor einen der ehemaligen Marktführer im Data-Cleaning-Sektor, Vality Technology, übernommen hatte.

[32] ETL (Extraktion, Transformation, Laden) ist einer der Kernprozesse von Data Warehouses, der die Gewinnung von Daten aus verschiedenen Quellen (Extraktion), ihre Überführung in die benötigte Form (Transformation) und das Einbringen der Daten in das Data Warehouse (Laden) umfasst.

– Warehouse Builder (Oracle)
– Highquality Suite (Human Inference)
– ChoiceMaker Suite (ChoiceMaker)

Daneben existieren auch zahlreiche kleinere Lösungen, die insbesondere keine Unterstützung für komplexe Cleaning-Workflows bieten und i. d. R. auf den Workstation-Betrieb ausgelegt sind. Dazu zählen z. B.:

– MatchIT (HelpIT Systems)
– Clean & Match (WinPure)
– LinkageWiz (LinkageWiz)

Während vor einigen Jahren der Fokus kommerzieller Data-Cleaning-Angebote noch klar auf Werkzeugen lag (vgl. z. B. [3]), vollzieht sich derzeit eine Trendwende zum Angebot von Data-Cleaning-bezogenen *Dienstleistungen*. Diese umfassen neben Beratungsangeboten und der Entwicklung individueller Cleaning-Lösungen insbesondere die Durchführung von Säuberungen großer Datenbestände und der Überwachung der Qualität von Datenbeständen. Nahezu alle der genannten Unternehmen bieten inzwischen derartige Dienstleistungen an und vermarkten diese als Alternative zum Erwerb der jeweiligen Produkte.

8 Zusammenfassung und Ausblick

Das Data Cleaning ist der Prozess der Identifikation und Korrektur von Anomalien in einer gegebenen Datenmenge. Es ist einer Teilgebiet der Informationsintegration, das von zentraler Bedeutung für den Umgang mit großen Datenbeständen und deren Pflege ist und in vielen Bereichen Anwendung findet.

Der Data-Cleaning-Prozess umfasst sechs Phasen:

1. Die *Datenanalyse* dient der Gewinnung von Metadaten und der Identifikation potentiell fehlerhafter Datensätze. Zur Umsetzung existieren statistische, musterbasierte, distanzbasierte und regelbasierte Verfahren.
2. Die *Workflow-Definition* legt die konkrete Abfolge von Transformationsoperationen für den gegebenen Kontext fest. Sie erfordert eine intensive Unterstützung des Anwenders durch Werkzeuge.
3. Die *Workflow-Verifikation* dient der Überprüfung der Korrektheit des spezifizierten Workflows vor dessen Ausführung.
4. Die *Transformation* umfasst drei Teilprozesse:
   (a) Die *Vorverarbeitung* dient der Überführung in ein syntaktisch und lexikalisch einheitliches Format (Normalisierung) und der Beseitigung semantischer Anomalien (Validierung).
   (b) Das *Record Matching* ist die Identifikation von Datensätzen, die sich auf dasselbe Objekt in der Miniwelt beziehen (Duplikate). Record-Matching-Verfahren bestehen aus einem Algorithmus zur Ablaufsteuerung und einer Menge von Ähnlichkeitsmetriken und weisen eine hohe algorithmische Komplexität – im Worst Case $O(N^2)$ – auf.

   (c) Das *Record Merging* führt Duplikate in einen einzigen Datensatz zusammen. Wichtigstes Kriterium ist dabei die Erhaltung des Informationsgehalts.
5. Die *Evaluation* dient der Bewertung der durchgeführten Transformationen. Wir unterscheiden dabei prozessbezogene Qualitätskriterien (z. B. Laufzeitkomplexität) und ergebnisbezogene Qualitätskriterien (z. B. Genauigkeit des Matchings). Metriken zur Messung der Qualität betrachten primär das Record Matching.
6. Der *Rückfluss der gereinigten Daten* in die ursprünglichen Datenquellen stellt den Abschluss des Cleaning-Prozesses dar. Er ist optional, aber sinnvoll zur permanenten Steigerung der Datenqualität und der Effizienz nachfolgender Säuberungen.

Zur Unterstützung des Anwenders bei der Durchführung des Data-Cleaning-Prozesses existieren zahlreiche Frameworks und Werkzeuge. Erstere sind in den meisten Fällen Ergebnisse wissenschaftlicher Arbeiten und zeigen Lösungsansätze auf. Werkzeuge sind vornehmlich kommerziell und sowohl für kleine Anwendungskontexte als auch in Form umfassender Anwendungssuiten zur Datenintegration verfügbar. In den letzten Jahren entwickelt sich auch ein Markt für das Data Cleaning als Dienstleistung.

Einige wesentliche Probleme des Data Cleanings sind das Fehlen von Standards bei Benennungen, die unzureichende Vernetzung verschiedener Wissenschaftsgebiete in der Theorie (Informatik, Mathematik, Informationswissenschaften) und der Austausch mit den Anwendungsgebieten (Medizin, Biologie, Wirtschaftswissenschaften, etc.). Zudem betrachten viele Arbeiten nur Teilaspekte des Data Cleanings, ohne sich um die Integrierbarkeit in einen umfassenden Prozess zu bemühen. Ebenfalls ein Schwachpunkt ist das Fehlen einheitlicher und vergleichbarer Qualitätsmetriken, mit dem sich jüngere Publikationen – zumindest im Bereich des Record Matchings – jedoch zunehmend auseinandersetzen. Ebenfalls zu erwarten ist eine Zunahme der Forschungsaktivitäten bei dem Cleaning nicht-relationaler Daten (z. B. XML) und vor allem nicht-textueller Daten (z. B. Bildern und Videos).

Literatur

1. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association **64**(328) (1969) 1183–1210
2. Cohen, W.W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. (1998) 201–212
3. Kimball, R.: Dealing with Dirty Data. DBMS Magazine **9**(10) (1996) 55
4. Rahm, E., Do, H.H.: Data Cleaning: Problems and Current Approaches. IEEE Data Engineering Bulletin **23**(4) (2000) 3–13
5. Milano, D., Scannapieco, M., Catarci, T.: Using Ontologies for XML Data Cleaning. In Meersman, R., Tari, Z., eds.: OTM Workshops 2005. Number 3762 in LNCS (2005) 562–571

6. Müller, H., Freytag, J.C.: Problems, Methods, and Challenges in Comprehensive Data Cleansing. Technical Report HUB-IB-164, Humboldt-Universität zu Berlin, Institut für Informatik (2003)
7. Sattler, K., Conrad, S.: Konfliktbehandlung in einer Anfragesprache für Datenbankföderationen. In Kutsche, R.D., Leser, U., Freytag, J., eds.: 4. Workshop "Föderierte Datenbanken" (Proceedings). (1999) 144–157
8. Hernandez, M.A., Stolfo, S.J.: Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. Data Mining and Knowledge Discovery **2**(1) (1998) 9–37
9. Galhardas, H., Florescu, D., Shasha, D., Simon, E.: Declaratively Cleaning your Data using AJAX. Journ. Bases de Donnees Avancees (2000)
10. Raman, V., Hellerstein, J.M.: Potter's Wheel: An Interactive Data Cleaning System. In: The VLDB Journal. (2001) 381–390
11. Maletic, J.I., Marcus, A.: Data Cleansing: Beyond Integrity Analysis. In: Proceedings of The Conference on Information Quality (IQ2000), Massachusetts Institute of Technology (2000) 200–209
12. Lu, R., Lee, M.L., Hsu, W.: Using Interval Association Rules to Identify Dubious Data Values. In Li, Q., Wang, G., Feng, L., eds.: Advances in Web-Age Information Management: 5th International Conference (Proceedings). Volume 3129 of Lecture Notes in Computer Science., Springer (2004) 528–538
13. Marcus, A., Maletic, J.I.: Utilizing Association Rules for the Identification of Errors in Data. Technical Report TR-14-2000, University of Memphis, Division of Computer Science (2000)
14. Perner, P. In: Data Mining on Multimedia Data. Volume 2558 of Lecture Notes in Computer Science. Springer (2002) 13–22
15. Conrad, S.: Schemaintegration – Integrationskonflikte, Lösungsansaätze, aktuelle Herausforderungen. Informatik – Forschung und Entwicklung **17**(3) (2002) 101–111
16. Christen, P., Belacic, D.: Automated Probabilistic Address Standardisation and Verification. [43] 53–68
17. Monge, A.E.: Matching Algorithms within a Duplicate Detection System. In: IEEE Data Engineering Bulletin. Volume 23. (2000) 14–20
18. Christen, P., Churches, T., Hegland, M.: Febrl – A Parallel Open Source Data Linkage System. In Dai, H., Srikant, R., Zhang, C., eds.: Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Proceedings. Volume 3056 of Lecture Notes in Computer Science., Springer-Verlag (2004) 638–647
19. Hernandez, M.A., Stolfo, S.J.: The Merge/Purge Problem for Large Databases. In: SIGMOD Conference. (1995) 127–138
20. Lee, M.L., Ling, T.W., Lu, H., Ko, Y.T.: Cleansing Data for Mining and Warehousing. In: Database and Expert Systems Applications. (1999) 751–760
21. Christen, P., Goiser, K.: Assessing Deduplication and Data Linkage Quality: What to Measure? [43] 37–52
22. Monge, A.E., Elkan, C.: The Field Matching Problem: Algorithms and Applications. In: Knowledge Discovery and Data Mining. (1996) 267–270
23. Wikipedia: Levenshtein Distance — Wikipedia, The Free Encyclopedia (2006) [Online; Abgerufen am 21. Juni 2006].
24. Hall, P.A.V., Dowling, G.R.: Approximate String Matching. ACM Comput. Surv. **12**(4) (1980) 381–402
25. Newcombe, H.B.: Record Linking: The Design of Efficient Systems for Linking Records into Individual and Family histories. American Journal of Human Genetics **19**(3) (1967)

26. Zobel, J., Dart, P.: Phonetic String Matching: Lessons from Information Retrieval. In: SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (1996) 166–172
27. Kedad, Z., Métais, E.: Ontology-Based Data Cleaning. In Andersson, B., Bergholtz, M., Johannesson, P., eds.: NLDB. Volume 2553 of Lecture Notes in Computer Science., Springer (2002) 137–149
28. Kang, J., Han, T.S., Lee, D., Mitra, P.: Establishing value mappings using statistical models and user feedback. In: CIKM. (2005) 68–75
29. Kang, J., Lee, D., Mitra, P.: Identifying Value Mappings for Data Integration: An Unsupervised Approach. In: WISE. (2005) 544–551
30. Monge, A.E., Elkan, C.: An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. In: Research Issues on Data Mining and Knowledge Discovery. (1997) 23–29
31. Bitton, D., DeWitt, D.J.: Duplicate record elimination in large data files. ACM Trans. Database Systems **8**(2) (1983) 255–265
32. Bleiholder, J.: Techniken des Data Merging in Integrationssystemen. In Samia, M., Conrad, S., eds.: Tagungsband zum 16. GI-Workshop Grundlagen von Datenbanken. (2004) 23–27
33. Menestrina, D., Benjelloun, O., Garcia-Molina, H.: Generic Entity Resolution with Data Confidences. Technical Report Stanford Infolab Publication Number 2005-35, Stanford University (2005)
34. Hylton, J.A.: Identifying and Merging Related Bibliographic Records. Technical Report MIT/LCS/TR-678, Massachusetts Institute of Technology (1996)
35. Christen, P., Goiser, K.: Quality and Complexity Measures for Data Linkage and Deduplication. In Guillet, F., Hamilton, H.J., eds.: Quality Measures in Data Mining. Studies in Computational Intelligence. Springer Verlag (2006)
36. Galhardas, H., Florescu, D., Shasha, D., Simon, E., Saita, C.A.: Improving Data Cleaning Quality Using a Data Lineage Facility. In: Design and Management of Data Warehouses. (2001) 3
37. Churches, T., Christen, P.: Blind Data Linkage Using n-gram Similarity Comparisons (2004)
38. Scheuren, F.: Linking Health Records: Human Rights Concerns. In Chapman, A.R., ed.: Health Care and Information Ethics: Protecting Fundamental Human Rights. Sheed and Ward (1997)
39. Bilenko, M., Mooney, R.J.: On evaluation and training-set construction for duplicate detection. In: Proceedings of KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation. (2003) 7–12
40. Christen, P.: Probabilistic Data Generation for Deduplication and Data Linkage. In Gallagher, M., Hogan, J., Maire, F., eds.: Intelligent Data Engineering and Automated Learning – IDEAL 2005: 6th International Conference (Proceedings). Volume 3578 of LNCS. (2005) 109–116
41. Hartel, C.R.: Architekturen und Frameworks für zuverlässige und adaptive Informationssysteme. Seminar Dependable Adaptive Information Systems, Lehrgebiet Datenverwaltungssysteme, Technische Universität Kaiserslautern (2006)
42. Elfeky, M.G., Verykios, V.S., Elmagarmid, A.K.: TAILOR: A Record Linkage Toolbox. In: ICDE. (2002)
43. Simoff, S.J., Williams, G.J., Galloway, J., Kolyshkina, I., eds.: Proceedings of the fourth Australasian Data Mining Conference. In Simoff, S.J., Williams, G.J., Galloway, J., Kolyshkina, I., eds.: Proceedings of the fourth Australasian Data Mining Conference, Sydney, Australia (2005)