Universität Leipzig

Fakultät für Mathematik und Informatik

Institut für Informatik

Übersicht über Crowdsourcing-Ansätze und Plattformen
zur Beurteilung
von Matchergebnissen

Bachelorarbeit

Leipzig, Nov, 2013

vorgelegt von:

Kubitzky, Sven

2226550

Studiengang: polyvalenter Bachelor Lehramt

Fachrichtung: Mathematik, Informatik

Betreuender Hochschullehrer: Prof. E. Rahm

Abteilung: Datenbanken

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Ein wesentlicher Bestandteil unserer heutigen Informationsgesellschaft sind Daten. Darunter ist eine formalisierte Darstellung von Informationen bezüglich realer Objekte zu verstehen. Die eigentlichen Informationen erhält man somit erst durch eine kontextspezifische Interpretation.

Mittlerweile lassen sich Daten auf unterschiedlichen Plattformen finden. Das können beispielsweise Produktdaten, Personen- oder Unternehmensdaten, sowie Geo-Daten sein. Diese liegen zumeist verteilt auf unterschiedlichen (heterogenen) Systemen.

Für eine Vielzahl von Anwendungsgebieten ist es von Bedeutung diese verteilten Daten zusammenzufügen. Für Unternehmen ist es zum Beispiel bedeutsam Daten über Produktbestandteile, Kundenbewertungen, etc., von Lieferanten und Kunden zusammenzutragen und auszuwerten. Für Verbraucher wiederum ist es wichtig, eine gezielte Produktsuche beispielsweise über Schlüsselwörter zu ermöglichen.
Für geographische Informationssysteme ist es unter anderem notwendig Daten über aktuelles Verkehrsaufkommen, Staumeldungen oder Baustellenstandorte zu sammeln und zeitnah auszuwerten, um so eine optimale Routenplanung für Einsatzkräfte, wie Polizei, Notarzt, Feuerwehr, zu gewährleisten.
Für die medizinische Forschung ist es ebenfalls von großer Bedeutung Forschungsergebnisse aus unterschiedlichen Quellen zusammenzutragen und auswerten zu können.

Um diese jeweils recht großen Datenvolumina bearbeiten zu können, ist es unabdingbar die menschliche Arbeit zu minimieren und den gesamten Prozess zumindest teilweise zu automatisieren. In der Vergangenheit wurden dafür bereits unterschiedliche Algorithmen und Verfahren entwickelt. Das Hauptproblem besteht weiterhin darin, zu entscheiden, ob eine Menge von Daten sich auf dasselbe reale Objekt bezieht. Demnach müssen die Ergebnisse der Matching-Verfahren verifiziert werden, um so eine Bewertung des Verfahrens zu erhalten und gegebenenfalls Verbesserungsmöglichkeiten zu finden.

Dieser Verifikationsprozess muss notwendiger Weise von Menschenhand getätigt werden. Hierbei bietet es sich an diese Aufgabe zu zerlegen und von einer großen Community bearbeiten zu lassen, sog. Crowdsourcing.

Diese Arbeit soll dafür einen Überblick der verschiedenen Crowdsourcing-Ansätze liefern und die Anforderungen und Probleme des Crowdsourcing untersuchen.
Dafür wird zunächst das Zusammenführen unterschiedlicher Datenbestände (data matching) betrachtet. Darauf folgt eine allgemeine Vorstellung des Crowdsourcing, um abschließend die unterschiedlichen Ansätze untersuchen zu können.

2 Datenintegration

2.1 Beschreibung Datenintegration

2.1.1 Ziel

Das Ziel der Datenintegration ist es, Daten aus mehreren unterschiedlichen Quellen zusammenzufügen und einheitlich, strukturiert darzustellen, um eine ganzheitliche und effektive Auswertung durch Anfragen zu ermöglichen. Dabei gilt es die Probleme zu lösen, die durch die Verteilung, Autonomie und Heterogenität der Quellen entstehen.

2.1.2 Prozess der Datenintegration

Zunächst soll der Prozess, bzw. der Ablauf, der Datenintegration grob vorgestellt werden.

Zusammengefasst unterteilt sich die Datenintegration in drei Phasen, wie die nebenstehende Abbildung verdeutlicht. In der ersten Phase des Schema-Matchings geht es darum, die verschiedenen Repräsentationen der Daten anzugleichen. Ziel dieses Prozesses ist eine einheitliche Darstellung der Daten. Insbesondere werden in diesem Bereich hauptsächlich die Metadaten betrachtet.

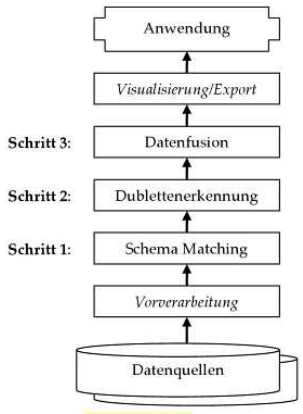

Abbildung 1 - Prozess der Datenintegration

In der darauffolgenden Phase der Dublettenerkennung (auch Object-Matching oder Entity Resulution) sollen unterschiedliche Repräsentationen eines realen Objektes erkannt werden. Das ist notwendig, damit keine redundanten Daten im Anfrageergebnis enthalten sind.

Die identifizierten Dubletten werden im letzten Schritt, der sogenannten Datenfusion, zu einer einzigen Repräsentation zusammengefügt. Dabei gilt es mögliche Datenkonflikte aufzulösen. [7]

In den folgenden Abschnitten werden zunächst verschiedene Probleme, welche den gesamten Prozess der Datenintegration begleiten, dargestellt. Anschließend erfolgt eine detailliertere Betrachtung der einzelnen hier genannten Phasen. Dabei werden auch Ansätze zur Behebung von Integrationsproblemen diskutiert.

Abschließend werden dann verschiedene Konzepte der Datenintegration vorgestellt.

2.2 Verteilung

Bei der Verteilung der Datenbestände unterscheidet man zwischen physikalischer und logischer Verteilung. Diese beiden Arten werden im Folgenden genauer betrachtet.

2.2.1 physikalische Verteilung

Bei der physikalischen Verteilung befinden sich die Daten auf voneinander unabhängigen Servern meistens an unterschiedlichen Orten. Diese Server haben in der Regel keinen gemeinsamen Speicher, sondern sind lediglich über ein Netzwerk miteinander verbunden. Die Gründe einer

physikalischen Verteilung ergeben sich aus einer höheren Ausfallsicherheit, einer verteilten Anfragebearbeitung (Lastbalancierung) und einer geringeren Latenz bzw. einer höheren Bandbreite durch die Nähe des Servers zum Client. Nachteilig ist, dass sich die Optimierung der Anfragen, sowie die Verwaltung der Daten schwieriger gestaltet. Darüber hinaus müssen die Daten über ein Netzwerk transportiert werden, wenn eine Anfrage beispielsweise auf mehrere Datenbestände unterschiedlicher Server zugreift.

2.2.2 logische Verteilung

Bei der logischen Verteilung befinden sich die Daten in unterschiedlichen Tabellen / Schemata mit unterschiedlichen Attributen. Meist werden hier verschiedene Tabellen für semantisch unterschiedliche Daten verwendet, eine Verknüpfung findet dann über Schlüssel statt (intensionale Überlappung). Allerdings kommt es auch vor, dass semantisch gleiche Daten auf verschiedene Tabellen verteilt sind. Dies kann einerseits bewusst eingerichtet sein, um beispielsweise Auswertungen zu Beschleunigen (Partitionierung), oder andererseits sich historisch durch autonome Datenquellen entwickelt haben. Diese redundante Datenhaltung birgt dabei immer die Gefahr von ungewollten und/oder widersprüchlichen Duplikaten.

Weitere Probleme die durch die Verteilung der Datenbestände resultieren sind, dass Daten übersehen werden können und dass sie schwerer verständlich sind. [1, 2]

2.3 Autonomie

Die Verteilung der Datenbestände hat unmittelbar zur Folge, dass auch die Autonomie der Daten zunimmt. Autonomie bedeutet in diesem Zusammenhang, dass die Datenbestände unabhängig voneinander verwaltet, konfiguriert und weiterentwickelt werden.

Man unterscheidet zwischen drei überlappenden Autonomie-Arten: Designautonomie, Kommunikationsautonomie und Ausführungsautonomie, auf die im Folgenden eingegangen wird.

2.3.1 Designautonomie

Designautonomie oder Entwurfsautonomie bezeichnet die freie Wahl, welches Datenmodell (relational, XML, …), welcher Schemaentwurf (Normalisierung, Bezeichnungen, Wahl von Attributen), welches Transaktionsmanagement etc., der Modellierung zugrunde gelegt wird. Ebenfalls findet die Definition der Schnittstellen (Funktionsnamen, Parameter, Rückgabeformate, …) autonom statt. Insbesondere gehört auch die Möglichkeit dies jederzeit ändern zu können mit zur Designautonomie. Findet beispielsweise bei 12 Quellen eine Änderung pro Jahr statt, so bedeutet das für das Integrationssystem im Mittel eine Änderung pro Monat, was ein ernstes Problem darstellt. [1, 2]

2.3.2 Kommunikationsautonomie

Die Kommunikationsautonomie beschreibt die freie Wahl „Was“, „Wann“, „Wie“ und „Mit Wem“ kommuniziert wird.

Es wird also unabhängig voneinander festgelegt, welche Informationen preisgegeben werden und ob dies jederzeit oder nur zu definierten „ruhigen“ Zeiten geschieht, bzw. ob eine Priorisierung der Zugriffe erfolgt, d.h. ob Anfragen von bestimmten Clients bevorzugt bearbeitet werden.

Das „Wie“ gibt darüber Aufschluss, welche Anfragemöglichkeiten (Sprache, Prädikate, Sortierung,…) unterstützt werden. Das „Mit Wem“ legt weiterhin fest, ob Clients bei zu vielen Zugriffen gesperrt werden, oder ob von Anfang an nur bestimmte IP-Räume zugelassen sind.

Zur Verdeutlichung kann man beispielsweise den Zugriff auf eine Buchdatenbank einmal über JDBC und einmal über ein HTML Formular (Bsp. Amazon) vergleichen. Man sieht leicht, dass man einmal als ausgewählter Nutzer (Login auf DB), alle verfügbaren Informationen, jederzeit im vollen SQL Umfang abfragen kann, oder als beliebiger Nutzer nur bestimmte Teilinformationen durch ausfüllen vordefinierter Suchfelder erhält, wie man bei Abbildung 2 sieht.

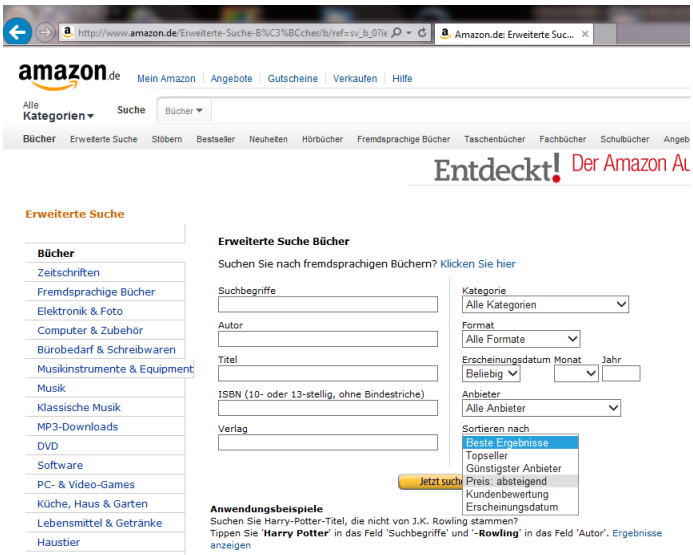

Abbildung 2 - Erweiterte Büchersuche bei Amazon

Dies entspricht den unterschiedlichen Aspekten der Kommunikationsautonomie. [1, 2]

2.3.3 Ausführungsautonomie

Die letzte Autonomie-Art ist die sogenannte Ausführungsautonomie. Das bedeutet, dass bei unterschiedlichen Datenquellen unabhängig voneinander festgelegt wird, wie Anfragen bearbeitet werden, ob also eine Optimierung stattfindet und auf welchem Synchronisationslevel diese Anfragen ablaufen. Weiterhin ist auch die zeitliche Ablaufsteuerung (Scheduling), wann also Anfragen ausgeführt werden, autonom festgelegt. So können einzelne Quellen bestimmte Antwortzeiten garantieren und andere nicht, was wiederum ein Problem für das Integrationssystem darstellt. [1, 2]

2.4 Heterogenität

Die autonom getroffenen Entscheidungen bezüglich Design, Kommunikation und Ausführung haben zur Folge, dass sich Informationssysteme technisch, syntaktisch, strukturell oder semantisch voneinander unterschieden. [1]

Das Integrationssystem muss diese Unterschiede überbrücken. Dabei treten unterschiedliche Probleme auf.

2.4.1 Technische Heterogenität

Aus technischer Sicht ist problematisch, dass unterschiedliche Informationssysteme unterschiedliche Anfragesprachen, unterschiedliche Kommunikationsprotokolle und unterschiedliche Austauschformate verwenden.
Bei drei verschiedenen Quellen könnte dies beispielsweise so aussehen, dass es sich bei der Ersten um eine Oracle-Datenbank handelt, die einen Zugriff über JDBC ermöglicht. Die Zweite könnte eine XML-Datenbank sein, welche Zugriffe mittels XPath bzw. XQuery unterstützt. Die dritte Quelle wiederum könnte ein Angebot von statischen HTML-Seiten zu Verfügung stellen, auf die über HTTP-Protokolle zugegriffen wird. [3] Es muss somit eine Übersetzung einer Anfragesprache in eine andere stattfinden.

Darüber hinaus treten Probleme auf, wenn die jeweils verwendeten Anfragesprachen nur bestimmte Operatoren (Vergleiche, Konjunktion, Disjunktion, Negation) zulassen, oder verschiedene Variablen einmal als obligatorisch (gebunden) und einmal als fakultativ (frei) angesehen werden. Kommt es hierbei zu Unstimmigkeiten zwischen den Informationssystemen und dem Integrationssystem, so können Anfragen unter Umständen nicht ausgeführt werden. Dies tritt vor allem dann auf, wenn die globale Anfragesprache, das heißt die Anfragesprache des Integrationssystems, mehr Operationen zulässt (mächtiger ist), als die Anfragesprache des einzelnen Informationssystems (lokale Anfragesprache).
Beispielsweise unterstützt die erweiterte Büchersuche von Amazon keine gleichzeitige Auswahl von Verlag und Preis eines Buches (vgl. Abbildung 2). Somit kann das einfache SQL-Statement:

```
SELECT *
FROM Buch
WHERE Verlag = “Springer“
AND Preis = 49.95
```

nicht direkt übersetzt werden. Weitere Einschränkungen können zusätzlich gegeben sein, indem beispielsweise nur ein Verbund (Join) über eine festgelegte Anzahl von Relationen zulässig ist. [1]

2.4.2 Syntaktische Heterogenität

Probleme, die aus syntaktischen Unterschieden resultieren, sind unterschiedliche Darstellungen des gleichen Sachverhaltes. So können beispielsweise Dezimalzahlen mit Punkt- oder Kommanotation geschrieben, Währungseinheiten durch Symbole (€, $) oder Worte (EURO, DOLLAR) oder Abkürzungen (EUR, USD) dargestellt, oder unterschiedliche Datumsformate (1.2.1999, 2/1/1999, 1999/2/1, 1. Febr. 99, usw.) verwendet werden, um nur einige Beispiele zu nennen. Für standardisierte Darstellungen existieren Übersetzungstabellen, was eine Integration nicht weiter problematisch macht. Für Sachverhalte, zu denen keine standardisierten Darstellungen vorliegen, wie Konferenznahmen, Veranstaltungsbezeichnungen, Buchtitel (Titel oder Titel + Untertitel), gestaltet sich eine Integrationssicht weitaus schwieriger. [1]

2.4.3 Strukturelle Heterogenität

Unter struktureller Heterogenität, oder strukturellen Unterschieden, versteht man allgemein die Tatsache, dass gleiche Datenbestände von verschiedenen Quellen durch unterschiedlichen Schemata abgebildet werden. So finden sich beispielsweise unterschiedliche Aufteilungen von Attributen auf Relationen. Dies geschieht, indem ein reales Objekt, oder ein Objektorientiertes-Modell auf unterschiedliche Art und Weise auf ein relationales Modell abgebildet wird. Insbesondere kann es hierbei zu Konflikten in der Modellierung von Relationen, Attributen und Werten kommen (Schematische Heterogenität), wie folgendes Beispiel (entnommen aus [1] und [2]) verdeutlicht:

Relation vs. Attribut:

maenner(Id, vorname, nachname) frauen (Id, vorname, nachname)	vs.	person (Id, vorname, nachname, maennlich?, weiblich?)

Relation vs. Wert:

maenner(Id, vorname, nachname) frauen (Id, vorname, nachname)	vs.	person (Id, vorname, nachname, geschlecht)

Attribut vs. Wert:

person (Id, vorname, nachname, maennlich?, weiblich?)	vs.	person (Id, vorname, nachname, geschlecht)

2.4.4 Semantische Heterogenität

Es bleiben abschließend noch die semantischen Unterschiede, bzw. die semantische Heterogenität, zu klären. Darunter werden Unterschiede in der Bedeutung und Interpretation von Schemaelementen und Attributen verstanden. So findet man häufig Synonyme, also verschiedene Bezeichnungen für dieselbe Informationsmenge. Als Beispiel betrachte man folgende Relationen: [3]

Q1: **Angestellter** (Id, Vorname, **Name**, männlich, weiblich)
Q2: **Person** (Id, Vorname, **Nachname**, Geschlecht)

Hierbei handelt es sich bei den Begriffen <Angestellter, Person> und <Name, Nachname> um Synonyme.

Weiterhin können bei der Überschreitung von Domänengrenzen auch Homonyme, also gleiche Bezeichnungen für verschiedene Informationsmengen auftreten. So hat das Attribut „Funktion“ in der Relation Angestellter eine andere Bedeutung, als das Attribut „Funktion“ in der Relation Protein. [3] Die Semantik einer Bezeichnung ist somit immer kontextabhängig, was eine automatische Zusammenfügung zu einem Integrationssystem erheblich erschwert.

Im Folgenden sollen nun die einzelnen Phasen der Datenintegration genauer beleuchtet werden.

2.5 Schemaintegration

Unter Schemaintegration versteht man allgemein eine Zusammenfassung mehrerer (Export-) Schemata zu einem globalen konzeptionellen Schema, wie es unter anderem für Mediator-Wrapper-Basierte Systeme (2.9.2) notwendig ist.

Zum besseren Gesamtverständnis wird vorab die sog. 5-Schichten-Architektur vorgestellt. Schematisch setzt diese sich wie folgt zusammen:

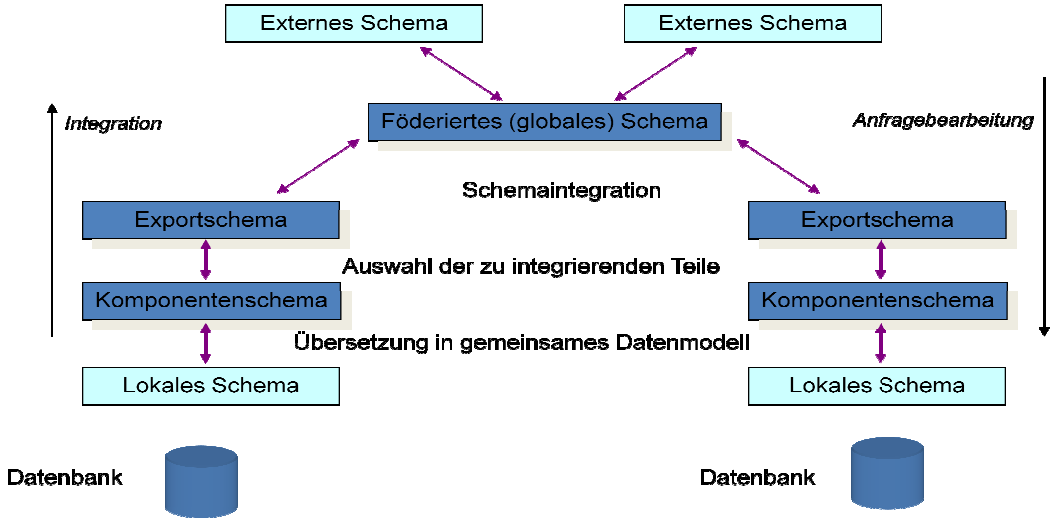

Abbildung 3 - 5 Schichten Architektur

Zunächst wird das lokale Schema in ein Komponentenschema überführt, indem fehlende Semantik hinzugefügt wird. Der Übergang zwischen lokalen Schemata und Komponentenschemata findet über Mappings statt. Das Exportschema stellt jene Teilmenge des Komponentenschemas dar, auf die von außen zugegriffen werden darf. Es ist demzufolge überflüssig, sofern das komplette Komponentenschema exportiert wird. Das föderierte Schema integriert sämtliche Exportschemata und kennt demzufolge die Datenverteilung. Die externen Schemata sind anwendungsabhängig und ermöglichen eine kontrollierte Sicht auf das Gesamtsystem. [1]

Der Prozess der Schemaintegration umfasst die folgenden 3 Phasen: (a) Vorintegration (2.5.1), (b) Erkennung und Behebung von Konflikten (2.5.2) und (c) Mischung und Restrukturierung der Schemaangaben (2.5.3).

2.5.1 Vorintegration

Im Bereich der Vorintegration muss zunächst bei mehreren Schemata festgelegt werden, in welcher Reihenfolge und nach welcher Strategie die einzelnen Schemata integriert werden sollen. Bei den Integrationsstrategien gibt es allgemein 2 unterschiedliche Herangehensweisen.

Zum einen die binäre Vorgehensweise, bei welcher in mehreren Schritten je zwei Schemata integriert werden.
Zum anderen die n-ären Vorgehensweise. Hier wird versucht möglichst viele Schemata über wenige Zwischenschritte zu integrieren. [5]

Folgende Abbildung dient zur Veranschaulichung:

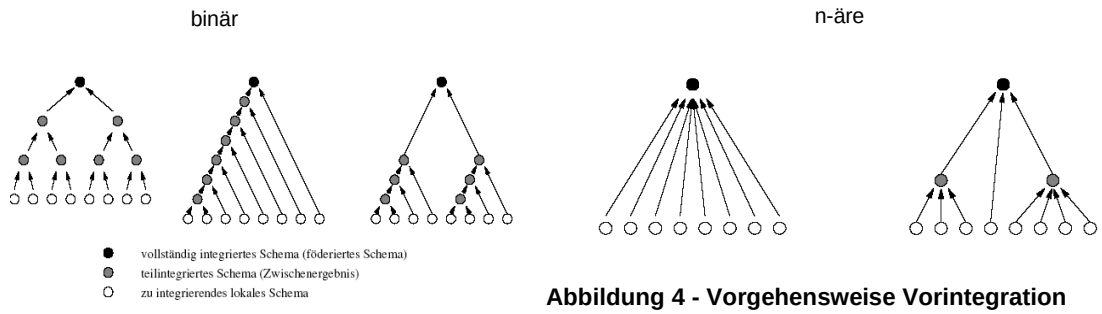

Abbildung 4 - Vorgehensweise Vorintegration

Weiterhin werden im Rahmen der Vorintegration die Schlüsselkandidaten der beteiligten Schemata bestimmt, wodurch Abhängigkeiten zwischen den Schemata erkannt werden können. Des Weiteren gilt es Transformationsabbildungen zur Beseitigung von Datenkonflikten (Datentypen: int vs. string, unterschiedliche Währungen, Skalierungen, etc.) zu definieren.

Als Beispiel (entnommen aus [5] sowie [3]) betrachten wir folgende Schemata:

Quelle 1 - UNIBIB:

PUBLIKATION (Pubnr, Titel, Typcode)
BUCHPUB (Pubnr, Verlag, Ejahr, Exemplare, ISBN)
VERFASSER (Pubnr, Vname)
SCHLAGWORT (Pubnr, Sname)

Quelle 2 - STADTBIB:

BUCH (ISBN, Titel, Autor, Vnr, Jahr, Preis, Standort)
VERLAG (Vnr, Vname, Adresse)

Das Attribut ISBN in der Relation BUCHPUB sei dabei ein Schlüsselkanditat, wodurch wir eine Abhängigkeit zwischen den Relationen BUCHPUB und BUCH feststellen.
Da wir lediglich zwei Schemata integrieren wollen, entfällt die Wahl einer Integrationsstrategie an dieser Stelle.

2.5.2 Erkennen und Beheben von Konflikten

In der folgenden Phase der Erkennung und Behebung von Konflikten sind die Probleme festzustellen und zu beheben, die aufgrund der semantischen Heterogenität entstehen. Zur Beseitigung von Namenskonflikten reicht es dabei aus, die betroffenen Attribute umzubenennen. Dies kann automatisch geschehen. Für die Beseitigung struktureller Probleme, also Probleme, die aus unterschiedlichen Modellierungen resultieren, existiert keine allgemeine Vorgehensweise. Dieser Prozess muss somit manuell, unter Berücksichtigung der Anwendungssemantik, geschehen.

Für unser Beispiel treten die folgenden Namenskonflikte auf:

- Verlag in BUCHPUB vs. Vname in VERLAG
- Ejahr in BUCHPUB vs. Jahr in BUCH
- Vname in VERFASSER (vs. Vname in VERLAG) vs. Autor in BUCH

Zur Beseitigung nehmen wir folgende Änderungen im Schema UNIBIB vor:

Quelle 1 - UNIBIB:

PUBLIKATION (Pubnr, Titel, Typcode)
BUCHPUB (Pubnr, *Vname*, *Jahr*, Exemplare, ISBN)
VERFASSER (Pubnr, *Autor*)
SCHLAGWORT (Pubnr, Sname)

2.5.3 Mischung und Restrukturierung

In der letzten Phase geht es darum, die Angaben der einzelnen Schemata so zusammenzuführen, dass keine Informationen verloren gehen. Weiterhin soll das erzeugte Schema möglichst keine Redundanz aufweisen, also minimal sein. Diese Forderung ist dadurch begründet, dass Änderungsoperationen durch automatische Anwendung auf alle betroffenen Kopien die globale Konsistenz der Daten gewährleistet.

Für unser Beispiel besteht nunmehr der Konflikt, dass beide Schemata unterschiedliche Primär- bzw. Fremdschlüssel verwenden, welche dem jeweils anderen Schema unbekannt sind. Ferner ist die Bedeutung der Attribute “Typcode“, “Exemplare“ und “Standort“ lokal begrenzt.

Zur Beseitigung dieser Konflikte müssen sog. Transformationsfunktionen eingerichtet werden, die die Möglichkeit bieten zu einer ISBN den entsprechenden Pubnr-Eintrag, sowie zu einem Vname-Eintrag den entsprechenden Vnr-Wert zu ermitteln. Liegt also der ISBN Wert (BUCH), sowie der Vname Wert (VERLAG) in einem Eintrag der BUCHPUB Relation vor, so ergibt sich eine entsprechende Zuordnung der Einträge. Ist dies nicht der Fall, so müssen neue noch nicht verwendete Werte generiert und vergeben werden. [5, 3]

Darüber hinaus ist eine Transformationsfunktion notwendig, die aus dem Attribut Autor (BUCH) die Namen der einzelnen Verfasser bestimmt und über die entsprechenden Schlüsselbeziehungen (ISBN -> Pubnr) eine Repräsentation für die Relation VERFASSER erstellt. Das daraus resultierende Schema könnte somit folgende Gestalt haben:

PUBLIKATION (Pubnr, Titel, Typcode)
BUCHP (Pubnr, Vnr, Jahr, Preis, Standort-STADT, Ex-UNI, ISBN)
VERFASSER (Pubnr, Autor)
SCHLAGWORT (Pubnr, Sname)
VERLAG (Vnr, Vname, Adresse)

Hierbei wurden die Attribute der BUCH Relation auf die neue Relation BUCHP, PUBLIKATION, VERFASSER abgebildet. Die Angaben aus der Relation BUCHPUB befinden sich weiterhin in BUCHP, lediglich der Verlagsname wurde in VERLAG aufgenommen.

Das Hauptproblem bei der Schemaintegration liegt darin, dass semantische Konflikte weitestgehend nur manuell behoben werden können. Dies setzt jedoch eine genaue Kenntnis über den Aufbau aller beteiligten Datenbanken voraus. Da die Offenlegung dieses Wissens jedoch eine Einschränkung der Autonomie bedeutet, gestaltet sich der Entwurf eines integrierten Schemas im Allgemeinen schwierig.

Ebenfalls problematisch ist, dass Änderungen an den lokalen Schemata, welche aufgrund der Autonomie jederzeit möglich sind, Rückwirkungen auf das globale Schema haben, sofern diese Änderungen das Export-Schema betreffen. In diesem Fall ist eine zumindest teilweise Wiederholung der Schemaintegration vorzunehmen, was wiederum mit einem erheblichen Aufwand verbunden ist. Somit ist dieser manuelle Ansatz bei einer größeren Anzahl von Datenbanken nicht praktikabel. [5]

Neuere Forschungen gehen in die Richtung, für das globale Schema ein objektorientiertes Datenmodell zu verwenden. Verglichen mit dem Relationenmodell findet sich hier eine größere Modellierungsmächtigkeit, was die Behandlung von Schema- und Datenkonflikten erleichtert. Ebenfalls stehen durch Definition von objektspezifischen Funktionen mächtige Transformationsmöglichkeiten zur Verfügung. Ein Problem, welches mit der Wahl eines objektorientierten Datenmodells verbunden ist, besteht darin, dass Operationen, welche vom globalen Schema unterstützt werden, sich nur mit Einschränkungen auf die lokalen Schemata übertragen lassen, wodurch in der Regel eine eingeschränktere Funktionalität in Kauf genommen werden muss. [5]

2.6 Schema Mapping / Schema Matching

Der wesentliche Bestandteil der Schemaintegration ist das sogenannte Schema Mapping, beziehungsweise das (semi) automatisierte Schema Matching. Dabei werden durch eine Liste von Korrespondenzen die äquivalenten Bestandteile zweier heterogener Schemata in Beziehung gesetzt.

Kam es bei der Schemaintegration insgesamt darauf an, die Export-Schemata mehrerer heterogener Datenquellen in einem globalen Schema zu vereinen, so ist es beim Schema Mapping das Ziel eine Abbildung eines Quellschemas, respektive mehrerer Quellschemata, auf ein anderes Zielschema zu erhalten. Dafür gilt es Wertkorrespondenzen der Daten zu identifizieren, um so ein eine Abbildungsvorschrift (Schema Mapping) zwischen Quellschemata und Zielschema zu erhalten. Aus diesem Mapping sollen wiederum Transformationsanfragen generiert werden können, um die Quelldaten in die Struktur der Zieldaten zu überführen. [6]

Die beiden folgenden Abbildungen dienen zur Veranschaulichung des Schema-Mapping-Prozesses:

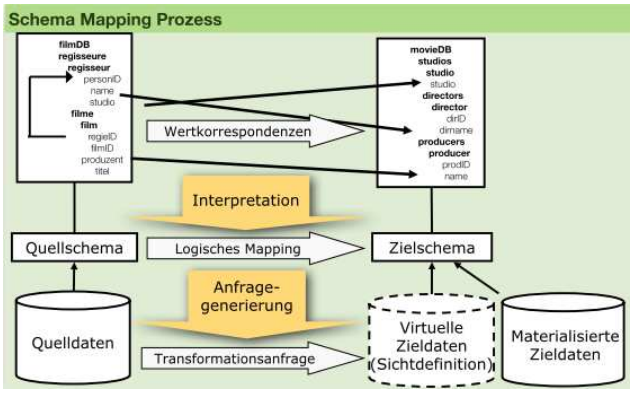

Abbildung 6 - Schema Mapping Prozess

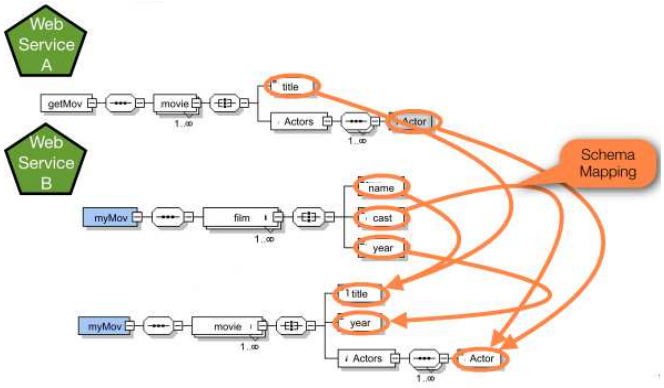

Abbildung 5 - Beispiel Schema-Mapping

Beim Schema Matching besteht insbesondere das Ziel, diese Abbildung automatisch zu generieren. Das bedeutet die Beziehungen der Schemata sollen aus den Schemainformationen selbst, ohne explizite Betrachtung der Daten, abgeleitet werden.

Folgende Grafik soll diese Unterschiede verdeutlichen:

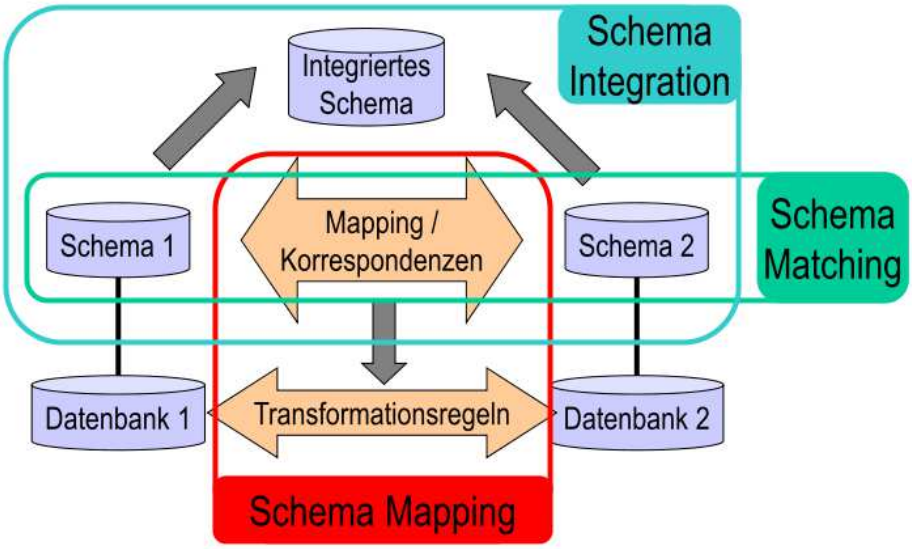

Abbildung 7 - Unterschiede Schema-Integration / ~Mapping / ~Matching

Für ein automatisches Mapping gilt es unterschiedliche Probleme zu überwinden. Diese Probleme werden einerseits durch große und somit unübersichtliche Schemata verursacht. Darunter werden Schemata mit über 100 Tabellen und entsprechend vielen Attributen, Schemata mit einer tiefen Schachtelung sowie mit einer Vielzahl von Fremdschlüssel-Beziehungen verstanden. Andererseits müssen auch Schemata, in denen unbekannte Synonyme / Homonyme, sowie fremdsprachliche oder kryptische Bezeichnungen verwendet sind, bearbeitet werden. Kryptische Bezeichnungen sind dabei Tabellen- oder Attributnamen, welche hinreichend kurz sind, so dass die Bedeutung der Daten nicht mehr aus der Bezeichnung hervorgeht. [1]

Um dennoch möglichst alle Korrespondenzen zu identifizieren und falsche Korrespondenzen zu vermeiden, existieren drei allgemeine Ansätze: [2]

- die beschriftungsbasierte Übereinstimmungssuche (label-based matching),
- die instanzbasierte Übereinstimmungssuche (instance-based matching),
- sowie die strukturbasierte Übereinstimmungssuche (structure-based matching).

Algorithmen und Verfahren zum Schema Matching kombinieren üblicher Weise mehrere dieser Ansätze. Dabei können mehrere Verfahren angewendet (hybride) oder die Ergebnisse mehrerer Verfahren kombiniert (composite) werden.

Im Folgenden sollen die oben genannten Ansätze erläutert werden.

2.6.1 Label-Based Matching

Bei diesem Verfahren werden die Attributmengen A und B zweier Schemata betrachtet. Zunächst wird das Kreuzprodukt der Mengen A und B gebildet, um anschließend jedem Tupel (a,b) aus A×B ein Ähnlichkeitsmaß bezüglich der Attributnamen zuzuordnen. [2]

Als Maß für die Ähnlichkeit zweier Attributnamen kann beispielsweise die Levenshtein-Distanz (Editier-Distanz) verwendet werden (vgl. B.3). Die ähnlichsten Paare, d.h. die Paare von Attributnamen mit der geringsten Editier-Distanz, werden dann als Match betrachtet. [6]

Dieser Ansatz gestaltet sich als problematisch, wenn, wie bereits angesprochen, kryptische Schemata vorliegen. Weiterhin ist das Verfahren nicht besonders effizient, denn wenn hinreichend viele Attribute vorliegen, wird das Kreuzprodukt dementsprechend mächtig. Ferner werden bei diesem Ansatz keine linguistischen Probleme (Synonyme, Homonyme, Fremdsprache) berücksichtigt.

Darüber hinaus bereitet die Wahl geeigneter Schwellwerte, ab denen ein möglicher Match akzeptiert wird, große Schwierigkeiten.

2.6.2 Instance-Based Matching

Bei diesem Verfahren sind zu den Attributmengen A und B zweier Schemata zusätzlich noch die jeweils darunter liegenden Daten gegeben. Die Kernidee besteht nun darin, für jedes Attribut interessante Eigenschaften, wie Buchstabenverteilung, Länge der Daten, zu extrahieren. Jedes Tupel des Kreuzproduktes aus A und B, wird nun bezüglich der Ähnlichkeit dieser Eigenschaften bewertet. [2]

Auf eine genauere Vorstellung der verwendeten Techniken wird an dieser Stelle verzichtet. Das instance-based matching ist selbst Gegenstand zahlreicher Forschungsarbeiten.

Es sei lediglich erwähnt, dass die Auswahl der zu vergleichenden Eigenschaften, sowie die Wahl einer geeigneten Vergleichsmethode Probleme verursacht. Ebenfalls ist es von Bedeutung, eine günstige Gewichtung der einzelnen Eigenschaften vorzunehmen. Hier geht die Entwicklung hin zum maschinellen Lernen.

2.6.3 Structure-Based Matching

Bei diesem Ansatz werden nicht nur zwei Attributmengen betrachtet, sondern allgemeiner zwei Elementmengen A und B zweier Schemata. Elemente können hierbei Attribute, Relationen, etc. sein. Die Kernidee ist es nun, die komplexe Struktur der Schemata auszunutzen, um Ähnlichkeiten zu finden. Hierbei können sowohl die Hierarchieebene, als auch die Nachbarschaftsbeziehungen betrachtet werden. [2] Ein Algorithmus hierfür ist das „Similarity Flooding“ nach [14], welcher unter B.1 näher beschrieben ist.

2.7 Object Matching

Durch das vorangegangene Schema-Matching wurde eine einheitliche Darstellung der Objekte erreicht. Das bedeutet, dass nun die Datenbestände mehrerer Quellen einheitlich in einem Schema betrachtet werden können. Der Einfachheit halber gehen wir davon aus, dass wir sämtliche Daten in einer Tabelle zusammenführen. Damit entsteht unweigerlich das Problem, dass mit hoher Wahrscheinlichkeit Datensätze vorhanden sind, die ein und dasselbe reale Objekt repräsentieren. [7]

Diese Tatsache scheint auf dem ersten Blick nicht dramatisch. Bei näherer Betrachtung fällt jedoch auf, dass bei Änderungsoperationen auf den Daten beispielsweise nur eine Repräsentation betroffen ist. Das hat zur Folge, dass in den Dubletten noch die alten Daten stehen, was eine Inkonsistenz des gesamten Systems nach sich zieht. Um dies zu verhindern, ist es unabdingbar diese redundanten Datensätze zu identifizieren und im nächsten Schritt („Datenfusion“) zu einer Repräsentation zusammenzufügen.

Das entscheidende Problem dieser Dubletten besteht darin, dass sich die Datensätze in genau den Attributen unterscheiden, die notwendig sind, um ein Objekt eindeutig zu beschreiben. Wäre dies nicht der Fall, könnten die Dubletten durch entsprechende UNIQUE-Klauseln für die betroffenen Attribute eliminiert werden. [7]

Klassische Beispiele, wie Dubletten entstehen, sind:

- das Hinzufügen / Weglassen von Zusatzinformationen:
    - “Frankfurt” vs. “Frankfurt am Main” vs. “Frankfurt a. M.”
    - “A Mapping-based Object Matching System” vs. “MOMA – A Mapping-based …”
- das Vertauschen der Reihenfolge
    - “Hartmetall-Anbohrer” vs. “Anbohrer, Hartmetall”
- Tippfehler, etc.

2.7.1 Erkennen von Dubletten

Zur Identifikation redundanter Datensätze werden wieder Ähnlichkeitsmaße eingeführt, anhand derer wieder über definierte Schwellwerte entschieden wird, ob Dubletten vorliegen oder nicht.

Neben dem bereits beschrieben Verfahren der Editier-Distanz existieren weitere Ansätze zur Definition von Ähnlichkeitsmaßen, beispielsweise das SOUNDEX-Maß (vgl. B.4) oder die Jaccard-Ähnlichkeit (vgl. B.5).

Ferner bietet es sich an sogenannte Partitionierungsstrategien anzuwenden. Das bedeutet es soll nicht das komplette Kreuzprodukt über alle Datensätze ausgewertet werden. Vielmehr sollen die Daten vorher gruppiert werden, was eine effizientere Auswertung ermöglicht. Beispielsweise ist es unnötig den „Max Mustermann“ aus Hamburg mit dem „Max Mustermann“ aus München zu vergleichen.

In [15] werden hierfür beispielhaft zwei Verfahren genannt. Einmal das sog. „sorted neighborhood“ und einmal das sog. „canopy clustering“.

Bei der ersten Methode werden zunächst Schlüssel (Zeichenketten) aufgrund von relevanten Merkmalen der Elemente gebildet. Diese Schlüssel und damit die Elemente selbst, werden anschließend mit einem Sortieralgorithmus lexikografisch sortiert. Abschließend werden durch ein sliding window die Gruppen (Cluster) gebildet. [22]

Das Canopy Clustering als ein weiteres sehr effizientes Verfahren zur Gruppierung von Datensätzen wird unter B.6 erklärt.

Insgesamt bietet es sich auch hier an, mehrere Verfahren zu kombinieren, sowie verschiedene Strategien des maschinellen Lernens einzubeziehen.

Die Resultate des Object-Matching-Prozesses lassen sich gemäß folgender Taxonomie einteilen.

Abbildung 8 - Taxonomie Object-Matching-Resultate

Es wird deutlich, dass auch hier nicht auf die manuelle Arbeit verzichtet werden kann. Um „False-Positives“ und „False-Negatives“ zu erkennen, ist das menschliche Interpretationsvermögen von großer Wichtigkeit. Aber selbst für den Menschen stellt dies keine triviale Aufgabe dar.

Beispielsweise ist die Entscheidung, ob es sich bei den Datensätzen „Patrick Meier, Hamburg“ und „Petrick Maier, Hamburg“ um ein und dieselbe Person (wegen Tippfehler), oder um zwei Personen handelt, nicht gerade einfach.

2.8 Datenfusion

Im zuvor stattgefunden Prozess der Dublettenerkennung wurden anhand der beschrieben Verfahren und durch manuelle Nachbearbeitung die entsprechenden Dubletten identifiziert. Nun gilt es zu diskutieren, wie diese zusammengeführt werden können, um letztlich alle Daten in einer Tabelle, genauer in einem Datenbestand, darzustellen. Dabei treten wieder Konflikte, sog. Datenkonflikte, auf.

Zum Beispiel wurden folgende Repräsentationen als Dublette eines Objektes identifiziert.

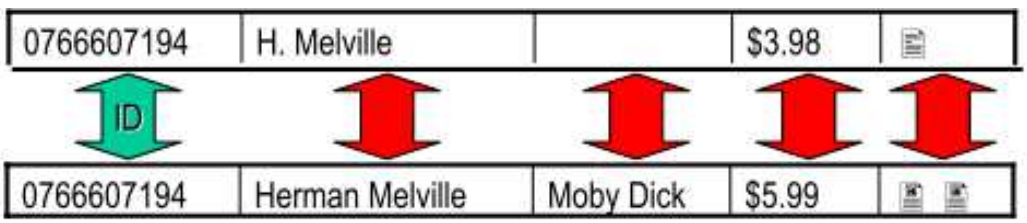

Abbildung 9 - Beispiel Duplikaterkennung

Es gilt hierbei die Fragen zu klären, wie zusätzliche Informationen („Moby Dick“ vs. NULL), bzw. unterschiedliche Informationen („$3.98 vs. $5.99“) gehandhabt werden. Dafür müssen unterschiedliche Gesichtspunkte berücksichtigt werden. Einerseits können die Aktualität der Quelle, sowie das (persönliche) Vertrauen, andererseits kann bei mehr als zwei Quellen eine Mehrheitsentscheidung (Majority Voting) betrachtet werden. [1]

Operationen zur Datenfusion sind zum einen der MINIMUM-UNION-Operator und zum anderen der MERGE-Operator.

Beim MINIMUM-UNION werden die Daten zunächst durch einen „outer union“ zusammengelegt, um anschließend die Tupel auszuwählen, welche die meisten Informationen beinhalten. Da an dieser Stelle allerdings keine Kombination einzelner Attributwerte stattfindet, ist dieses Verfahren unbefriedigend. [1]

Der MERGE-Operator stellt hingegen eine Kombination einer JOIN- und einer UNION-Operation dar. Die identifizierten Dubletten dafür seien über einen eindeutigen Bezeichner identifiziert. Es findet ein JOIN bezüglich dieses Bezeichners statt, wobei Datensätze, welche keinen Partner haben mittels UNION dennoch zur Gesamttabelle hinzugefügt werden. Bei Attributen, die einmal einen Nullwert und einmal einen von Null verschiedenen Wert aufweisen, wird anschließend der von Null verschiedene Wert verwendet. Sollten jeweils unterschiedliche Werte vorliegen, gilt es über eine Konfliktlösungsfunktion zu entscheiden welcher Wert verwendet wird. [1]

Eine genauere Erläuterung dieser Operationen findet sich bei [1].

2.9 Konzepte der Datenintegration

Im Folgenden sollen unterschiedliche Gesamtkonzepte der Datenintegration, wie sie auch in der Praxis Verwendung finden, vorgestellt werden.

2.9.1 Data Warehouse

Zur Integration von Daten in ein Data Warehouse, werden die Daten aus in der Regel verschiedenen Datenstrukturen zunächst in die Zielstruktur des Data Warehouses transformiert und anschließend in die zentrale zugrunde liegende Datenbasis kopiert. Da hier also eine Arbeit direkt auf den Daten stattfindet, spricht man beim Data Warehouse von einem materialisierten Integrationssystem. Anfragen an das Integrationssystem werden nun gegen die kopierten, materialisierten Daten gestellt.

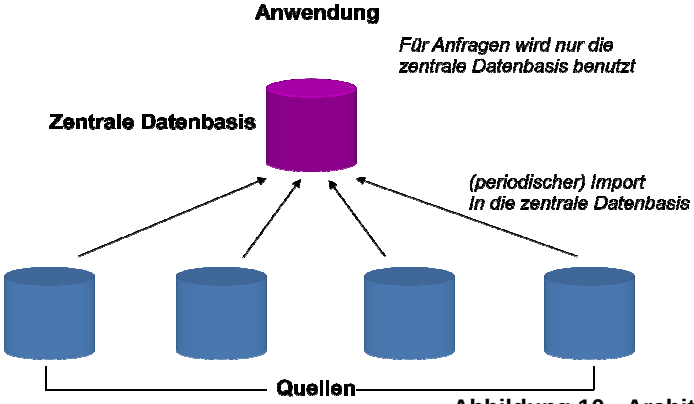

Abbildung 10 - Architektur eines Data-Warehouse

Der Vorteil, der beim Data Warehouse entsteht ist, dass klar definiert werden kann, welche Daten in die zentrale Datenbasis übernommen werden und welche nicht. Es kann also eine klare Trennung von wichtigen und unwichtigen Daten stattfinden.

Darüber hinaus ist das Data Warehouse sehr stabil, da Änderungen auf den Quelldaten keine unmittelbare Auswirkung auf die zentrale Datenbasis haben. Dies kann sich für bestimmte Auswertungen allerdings auch Nachteilig gestalten. [1]

2.9.2 Mediator-Wrapper-Basierte Informationssysteme

Anders als beim Data Warehouse handelt es sich hierbei um ein virtuelles Integrationssystem. Es findet also keine redundante Datenhaltung in einer zentralen Datenbasis statt, sondern die Anfragen werden von sogenannten Mediatoren in Teilanfragen zerlegt und für das jeweilige Schema der Quelle übersetzt. Die Daten werden somit nur bei Bedarf übertragen und temporär gespeichert. [1, 3] Wrapper sind dafür spezielle Softwarekomponenten, welche die Kommunikation und den Datenverkehr zwischen Mediatoren und Datenquellen gewährleisten. [1]

Als Beispiel [3] nehmen wir an, ein Nutzer möchte Informationen über Bohrmaschinen zum Preis unter 250€ erhalten. Er stellt an die Anwendung also folgende Anfrage:

```
SELECT Name, Preis, Bewertung
WHERE Preis < 250
 AND Kategorie = “Bohrmaschine“
```

Die Anwendung entscheidet zunächst, welche Quellen (Handwerkermarkt, Verbraucherportal) für diese Anfrage relevant sind und leitet sie an den entsprechenden Mediator weiter. Hier wird die Anfrage einmal für den Handwerkermarkt

```
SELECT Artikelname, Einzelpreis
WHERE Artikelkategorie = “Bohrmaschine“
 AND Preis < 250
```

und einmal für das Verbraucherportal

```
SELECT Hersteller, Artikel, Rating
WHERE Artikelkategorie = “Bohrmaschine“
```

zerlegt und an sogenannte Wrapper weitergeleitet, welche die Anfragen in eine passende Quellenanfrage übersetzen. So generiert der Wrapper für den Handwerkermarkt beispielsweise eine SQL-Anfrage über JDBC und der Wrapper für das Verbraucherportal eine HTTP-Anfrage.

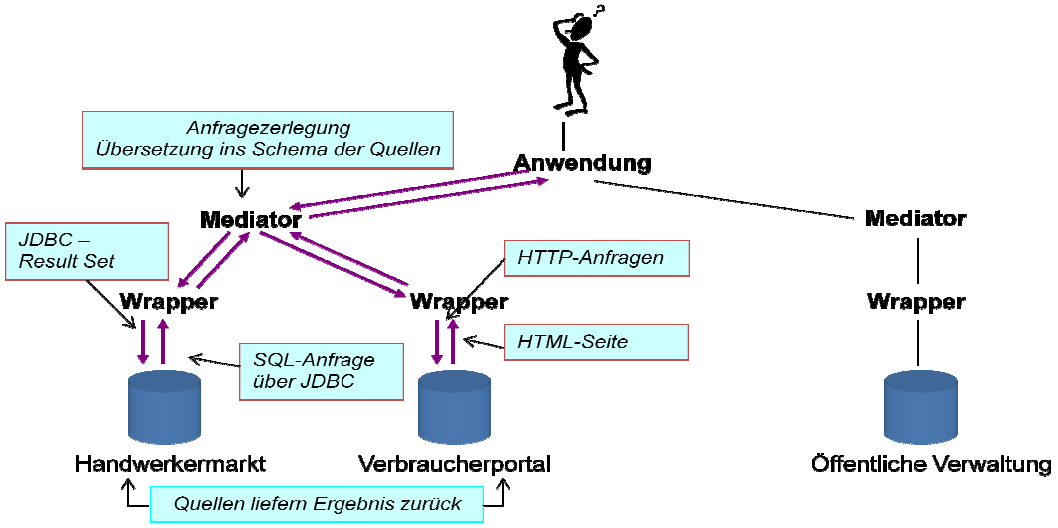

Abbildung 11 - Workflow Mediator-Wrapper-System

Die erhaltenen Anfrageergebnisse (einmal ein ResultSet und einmal eine HTML-Seite) werden anschließend von den jeweiligen Wrappern zusammengeführt, gefiltert und in das gemeinsame Datenmodell rücktransformiert. Der Mediator verschmilzt die Ergebnismengen und leitet diese an die Anwendung weiter, welche die Ergebnisse sammelt und dem Nutzer darstellt. Die Mediatoren überwinden somit die strukturelle und semantische Heterogenität, die quellenspezifischen Wrapper die technische Heterogenität der Datenquellen. [1]

Bei diesem System wird die 5-Schichten-Architektur besonders deutlich. Eine Anwendung (externes Schema) kommuniziert mit dem Mediator (föderiertes Schema). Der Mediator wiederum leitet die entsprechenden Teilanfragen an den Wrapper weiter. Der Mediator kommuniziert also mit dem Exportschema des Wrappers. Der Wrapper (Komponentenschema) wiederum kommuniziert mit den Quelldatenbanken (lokale Schemata).

Dieser Aufbau kann darüber hinaus so verallgemeinert werden, dass mehrere Mediatoren angelegt werden, die ebenfalls untereinander kommunizieren können. Ferner besteht auch die Möglichkeit, dass eine Anwendung direkt mit dem Wrapper kommuniziert.

Die Vorteile dieser Architektur liegen darin, dass stets mit den aktuellen Daten gearbeitet wird. Weiterhin hat das System nur sehr geringe Anforderungen an die Quellen, wodurch eine hohe Autonomie derselben zugelassen werden kann. Zusätzlich ist das Integrationssystem durch neue Wrapper, neue Mediatoren leicht erweiterbar.

Nachteilig ist, dass durch die Wrapper ein regelmäßiger und hoher Wartungs- bzw. Anpassungsaufwand entsteht. Weiterhin bleiben die Anfragemöglichkeiten durch die Quellenautonomie gegebenenfalls beschränkt. Darüber hinaus ist mit einer schlechteren Performanz bei Anfragen zu rechnen, weil beispielsweise Duplikate erst zur Laufzeit erkannt werden können. [1]

2.9.3 Peer-Data-Management-Systeme

Ein anderer Ansatz, um Daten aus unterschiedlichen Quellen in ein System zu integrieren, ist die Erstellung eines strukturierten Peer-to-Peer Netzwerkes. Es liegt hierbei keine Verknüpfung zwischen lokalen und globalem Schema mehr vor. Vielmehr findet ein Mapping zwischen Paaren von Peer-Schemata statt. Dabei kann jeder Peer Anfragen anderer Peers entgegennehmen, Sichten auf seine Daten zur Verfügung stellen, Daten exportieren und auch selbst Anfragen stellen. Ein Peer vereint

somit Mediator, Wrapper, Datenquelle und Benutzer. Dabei kann ein Peer auch selbst ein integriertes Informationssystem sein. [1]

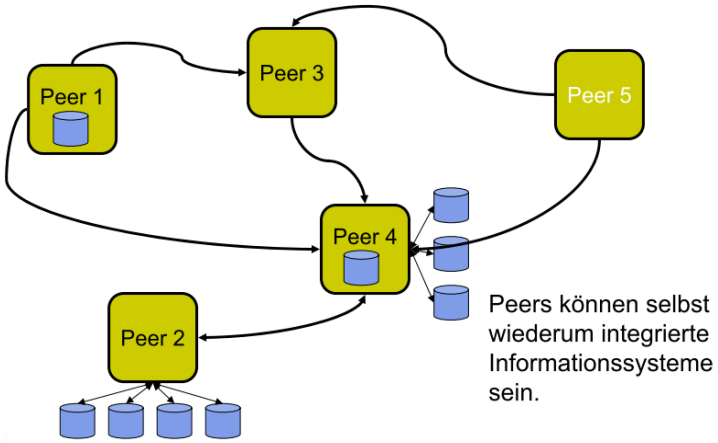

Abbildung 12 - Architektur eines PDMS

Ein solches Peer-Data-Management-System (PDMS) bietet sich vor allem dann an, wenn die Bildung eines globalen Schemas wenn überhaupt nur kompliziert möglich ist. Anwendungsgebiete sind beispielsweise die medizinische Forschung oder Gesundheitsinformationssysteme, wo jeweils verschiedene komplexe Schemata vorliegen. [1]

Vorteile dieses Systems sind, dass leicht neue Peers in das System hinzugefügt werden können, da kein globales Schemawissen notwendig ist. Weiterhin ist nur ein Mapping zum ähnlichsten Schema notwendig, da alle Daten über die transitive Hülle der Mappings zur Verfügung stehen. Solche Mapping können gegebenenfalls automatisch abgeleitet (Schema Matching) werden. Darüber hinaus müssen die Nutzer, also wieder die Peers, nur das eigene Schema kennen. [4]

Probleme, die ein PDMS mit sich bringt, sind, dass sehr schnell sehr komplexe Systeme entstehen können und somit die Gefahr von zyklischen Mappings (Redundanz) zunimmt. Weiterhin gestaltet es sich als schwierig, Anfragen bei vielen Zwischenstationen effizient zu bearbeiten. Darüber hinaus besteht bei jedem Zwischenschritt, also bei jedem Mapping, grundsätzlich die Gefahr, dass Informationen verloren gehen. [4]

Dieser Informationsverlust ist dadurch bedingt, dass die Mappings der einzelnen Schemata zum Teil unvollständig sind. Dieser Informationsverlust muss bei der Anfragebearbeitung berücksichtigt werden. Ein möglicher Ansatz, wie bei [13] beschrieben, ist, bei den einzelnen Mappings die Informationsqualität (IQ) anzugeben und durch Auswertung dieser Werte den günstigsten Pfad im PDMS zu wählen.

2.10 Matching Frameworks

Es wurde festgestellt, dass beim Schema-Matching zahlreiche unterschiedliche Probleme auftreten, welche nicht voll automatisch gelöst werden können. Vielmehr erleichtern zahlreiche Matching-Verfahren und Ansätze die Arbeit. Viele relevante Entscheidungen während des Matchings können

bislang nur manuell getroffen werden. In diesem Zusammenhang sprechen wir darum von semi automatischen Verfahren.

Aus diesem Grund ist es wichtig und notwendig, geeignete Anwendungen zur Verfügung zu stellen, die den Nutzer bei diesen Entscheidungen unterstützen. Diesbezüglich wurden und werden zahlreiche Prototypen entwickelt, die dies leisten sollen.

Einige dieser Forschungsprototypen sind:

- Cupid – University of Washington
- Clio – IBM Almaden + University of Toronto
- COMA / COMA++ / COMA 3.0 – Universität Leipzig

Exemplarisch soll im Folgenden das Matching-Framework COMA++ vorgestellt werden. Dies soll insbesondere dazu dienen, die Architektur, die Arbeitsweise, sowie die Anforderungen an ein Matching-Framework zu verdeutlichen.

2.10.1 COMA++[1]

COMA++ ist eine generische Match-Plattform, welche neben relationalen Schemata auch XML Schemas und OWL-Ontologien gleichermaßen unterstützt. COMA++ ist die Erweiterung des vorherigen Prototyps COMA und unterstützt eine Kombination unterschiedlicher Match-Algorithmen. Der Name COMA beruht dabei auf *flexible Combination of Schema Matching Approaches*.

Neben den verschiedenen Match-Strategien, welche beliebig kombiniert und konfiguriert werden können, besteht weiterhin die Möglichkeit bereits vorhandene Match-Ergebnisse wiederverwenden zu können. Darüber hinaus werden auch Ansätze unterstützt, welche große Match-Probleme in kleinere Probleme zerlegen, sog. fragment-based match approach.

Weiterhin können mit COMA nicht nur Match-Probleme bearbeitet werden. Das System bietet weiterhin die Möglichkeit die Effektivität verschiedener Algorithmen und Strategien zu evaluieren. [16]

Die Weiterentwicklung COMA 3.0 bietet neben einem erweiterten Arbeitsmanagement auch die Möglichkeit mehrere Ontologien zusammenzufügen (engl. merging). Zu diesem Zweck wurde das parallele Projekt ATOM (automatic target-driven ontology merging) in COMA 3.0 integriert.

COMA++ setzt sich grob aus 5 Teilen zusammen. Einem Archiv (*Repository*), einem *Model-* und einem *Mapping-Pool*, sowie einem *Match Cusomizer* und einer Ausführungsplattform (*Execution Engine*), wie in folgender Abbildung dargestellt:

[1] http://dbs.uni-leipzig.de/Research/coma.html

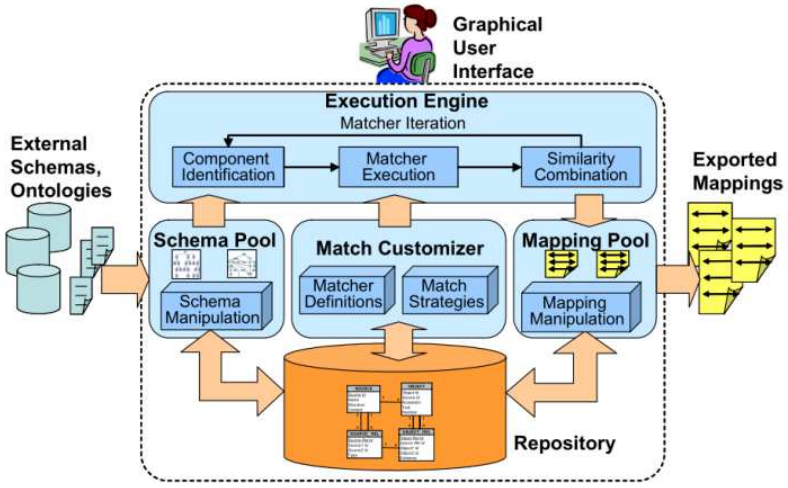

Abbildung 13 - Architektur COMA++

Im Repository werden alle relevanten Match-Daten persistent abgelegt. Dieses „Wissen“ wird dann bei der Modell- und Mapping-Erzeugung verwendet. Im Model-Pool werden alle Schemata und im Matching-Pool entsprechend alle berechneten Mappings gespeichert. Das eigentliche Matching zweier Schemata läuft in der Execution Engine ab. Dabei werden iterativ mehrere Phasen durchlaufen. Zunächst werden die Bestandteile des Schemas ausgewählt, welche dem Matching-Prozess unterzogen werden sollen (*component identification*). Auf diese Komponenten werden dann verschiedene Match-Strategien, sowie verschiedene Kombinationen derer, angewendet (*matcher execution*).

Das Ergebnis dieser Phase, also die berechneten Ähnlichkeiten, wird dann in einem sog. *Similarity Cube* dargestellt. Abschließend wird dann eine spezielle Kombination ausgewählt (*similarity combination*), auf Grundlage derer die Korrespondenz zwischen den betrachteten Komponenten abgeleitet wird. Das so erhaltene Mapping kann dann als Ausgangsbasis für eine neue Iteration verwendet werden. [16]

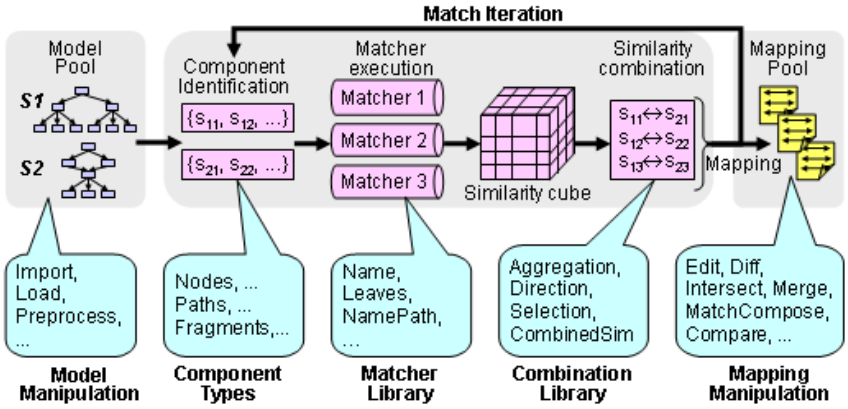

Abbildung 14 - Workflow COMA++

Durch den *Match Cusomizer* ist dabei dem Nutzer die Möglichkeit gegeben jeden Iterationsschritt individuell zu konfigurieren. Dafür steht insbesondere eine umfangreiche und nutzerfreundliche GUI (Graphical User Interface) zur Verfügung, wie in Abbildung 15 dargestellt.

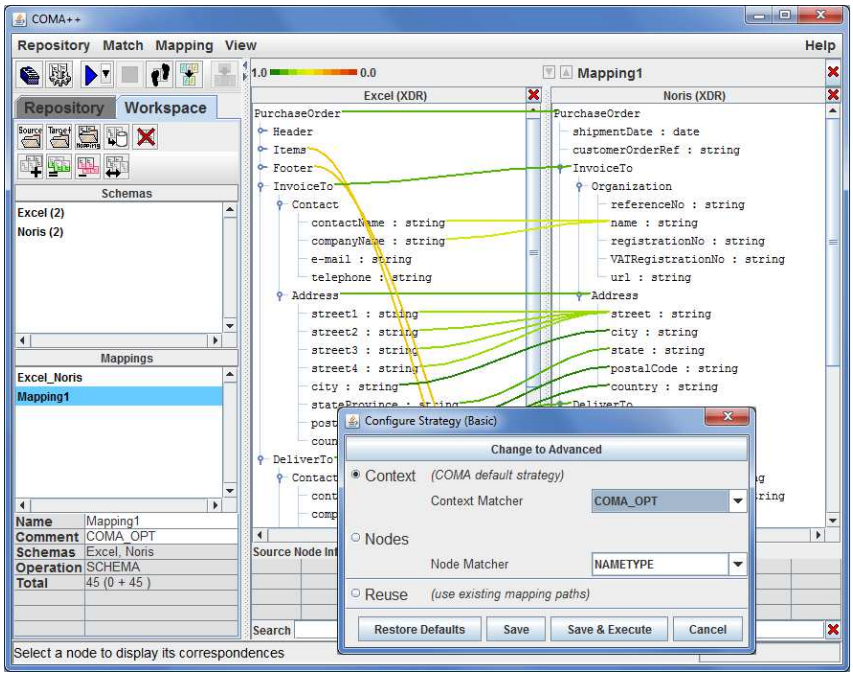

Abbildung 15 - User Interface von Coma++ (Version 2009)

Somit ist dem Nutzer insbesondere die Möglichkeit gegeben jedes Mapping zu editieren, um falsche Korrespondenzen zu entfernen und fehlende problemlos zu ergänzen.

Weiterhin bietet COMA++ die Möglichkeit zwei Schemata, bzw. zwei Ontologien, mit Hilfe einer Taxonomie zu matchen. Diese Taxonomie wird zwischen Quellschema und Zielschema gelegt, um Beziehungen zwischen Quell- und Zieldaten besser zu finden.

Wie in Abbildung 16 dargestellt, sind die Elemente Weizen und Kölsch Vertreter von obergärigem Bier (top-fermented beer). Mittels dieser Taxonomie ist somit eine Ähnlichkeit der beiden Elemente ableitbar, beispielsweise durch die Entfernung innerhalb der Taxonomie.

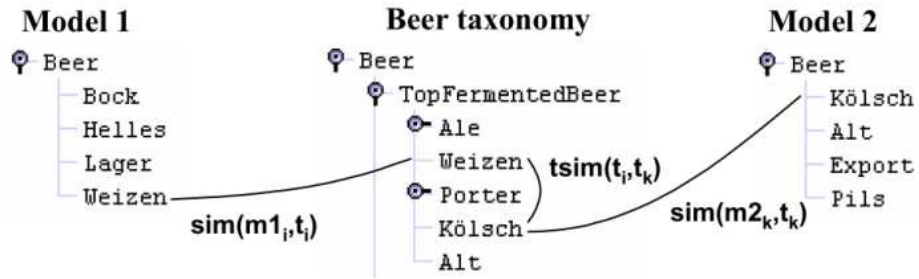

Abbildung 16 - Taxonomy-Based Matching

Die Wiederverwendung bereits berechneter Matchings wird insbesondere bei der Verwendung von sogenannten *Pivot-Schemas* deutlich. Das sind Standard-Schemata oder Ontologien innerhalb eines Geltungsbereiches, gegen welche alle neuen Schemata zuerst gematcht werden. Über dieses Matching wird das neue Schema dann (transitiv) auf das eigentliche Zielschema abgebildet. Mit einem solchen Mapping-Pfad, der maximal die Länge 2 hat, kann für das Mapping von zwei Schemata eine erheblich bessere Performance erzielt werden. [16]

Die Auswertung der gelieferten Matchergebnisse ist bei hinreichend großen Schemata allerdings nach wie vor mit einem hohen manuellen Aufwand, sowie unter Umständen mit domänenspezifischem

Expertenwissen verbunden. Da sich dieses Problem allem Anschein nach nicht durch Algorithmen lösen lässt, ist eine Idee dieses Problem in sehr viele, sehr kleine Einzelaufgaben zu zerlegen. Diese Einzelaufgaben könnten dann von einer großen Community bearbeitet werden, um so gemäß dem Divide-and-Conquer Prinzip eine effiziente und stimmige Lösung des Gesamtproblems zu erhalten.

Somit liegt es nahe Crowdsourcing-Ansätze zu untersuchen. Crowdsourcing verspricht nämlich genau das zu leisten.

2.11 Zusammenfassung Datenintegration

Bei der Datenintegration unterscheidet man allgemein zwischen materialisierter und virtueller Datenintegration.
Bei der materialisierten Datenintegration, werden die Daten mehrerer Quellen in eine zentrale, globale Datenbank integriert. Das Konzept, welches darauf aufbaut ist das Data Warehouse.
Bei der virtuellen Datenintegration hingegen werden Anfragen an das Gesamtsystem in quellabhängige Anfragen übersetzt, um die Daten zu ermitteln. Anschließend erfolgt eine Rückübersetzung der erhaltenen Ergebnisse. Beispiele hierfür sind einmal die Mediator-Wrapper-Systeme, sowie die Peer-Data-Management-Systeme.

Der gesamte Prozess der Datenintegration wird von mehreren Problemen und Konflikten begleitet, die aus der logischen und physikalischen Verteilung der Datenbestände resultieren. Die Verteilung der Daten hat ursächlich zur Folge, dass die Datenquellen autonom entworfen, betrieben und verändert werden. Diesbezüglich wird dann von Datenheterogenität gesprochen.

Ziel der Datenintegration ist es, diese Heterogenität zu überwinden. Um dies zu bewerkstelligen gliedert man den Prozess in drei Phasen.

In der Phase der Schemaintegration, insbesondere dem Schema-Matching, geht es darum unterschiedliche Darstellungen von Quellschemata und Zielschema zu beseitigen. Dabei wird eine Abbildung zwischen diesen Schemata gesucht.

In der Phase des Object-Matching sollen unterschiedliche Repräsentationen ein und desselben realen Objektes identifiziert werden. Aus materieller Sicht ist das notwendig, um die Konsistenz der Daten zu gewährleisten. Für die virtuelle Datenintegration ist es wichtig ein redundanzfreies Anfrageergebnis zu erhalten.

In der letzten Phase der Datenfusion werden dann die erkannten unterschiedlichen Repräsentationen zu einer Repräsentation zusammengefügt.

Das Hauptproblem der Datenintegration ist es bei stark heterogenen Quellen zu entscheiden, welche Informationen die gleiche Bedeutung haben, bzw. welche Datensätze dasselbe reale Objekt referenzieren.
Es existieren dafür verschiedene Ansätze und Verfahren, die meist aber nur eine grobe Auswahl liefern. Somit bleibt die manuelle Arbeit an dieser Stelle (zur Zeit) unerlässlich. Eine Unterstützung hierfür ist durch zahlreiche Frameworks gegeben.

3 Crowdsourcing

3.1 Definition

Der Begriff Crowdsourcing bezeichnet, vereinfacht ausgedrückt, die Auslagerung von Arbeitsprozessen auf eine große Masse von Menschen. Insbesondere erfolgt diese Auslagerung unter Verwendung moderner Kommunikationstechnologien (Web 2.0 oder Social Media).

Der Begriff selbst geht auf den US-Amerikaner Jeff Howe (Journalist – Wired Magazine) zurück. Dieser kombinierte in seinem Artikel „The Rise of Crowdsourcing“ (2006) die Worte „Crowd“ (Menschen-Masse, auch Human Cloud) und „Outsourcing“ (Auslagerung von Arbeit an Drittanbieter). Eine allgemeingültige Definition für das Crowdsourcing konnte bis heute allerdings nicht erbracht werden. Das liegt insbesondere daran, dass nicht immer ein unternehmens- bzw. auftraggeberinitiierter Tätigkeitsanstoß vorliegen muss. Vielmehr generiert die (virtuelle) Community selbstständig Dienstleistungen und Produkte zur wirtschaftlichen Nutzung. [18]

Betrachtet man unterschiedliche Definitionen, so lassen sich mehrere konstituierende Merkmale des Crowdsourcings ableiten. So wird unter Crowdsourcing insgesamt eine

- reaktiv oder proaktiv entwickelte,
- kollaborativ oder wettbewerbsorientiert organisierte,
- auf freiwilliger Basis stattfindende Form
- der Leistungsgenerierung,
- durch eine heterogene, unbekannte, geographisch weit verteilte Masse
- von extrinsisch oder intrinsisch motivierten Internetnutzern,
- mit unterschiedlichem Wissensstand,
- innerhalb eines klar definierten Rahmens (Software, Zeitraum, etc.),
- unter Verwendung moderner Kommunikationsmedien (Web 2.0),

verstanden.

Hinter dem Crowdsourcing verbirgt sich also der Ansatz, dass eine heterogene Masse von individuell entscheidenden Personen insgesamt die Qualität von Expertenentscheidungen erreichen kann (Schwarmintelligenz).

In seinem Sachbuch “The Wisdom of Crowds“ (2004) argumentiert J. Surowiecki, dass Gruppenentscheidungen oft besser sind als die Lösungsansätze einzelner Experten innerhalb der Gruppe. Dafür nennt er unterschiedliche Voraussetzungen. Insbesondere müssen die Einzelentscheidungen **unabhängig** von den Entscheidungen anderer Gruppenteilnehmer getroffen und in richtiger Weise zusammengefasst werden (vgl. ebenda).

3.2 Taxonomie

Vorab sei bemerkt, dass es in der Literatur kein allgemein anerkanntes Klassifikationsschema gibt. Ebenso, wie beim Versuch eine Definition für Crowdsourcing zu geben, erschweren hier unterschiedlichste Sichtweisen die Klassifikation.

Intuitiv lässt sich unterschieden, in welcher Art und Weise ein Nutzer („Crowdworker“, oder kurz „Worker“) die Daten einbringt und ob er sich dessen bewusst ist. Ferner kann man unterscheiden, in welchem Zusammenhang das Crowdsourcing mit den benötigten Daten steht.

In ihren Arbeiten greifen G. Bombach (Belegarbeit, Jan. 2013) und T. C. Hara (Diplomarbeit, Nov. 2012), beide von der TU-Dresden, diese Ansätze auf und geben folgende grobe Taxonomie an: (ebenda)

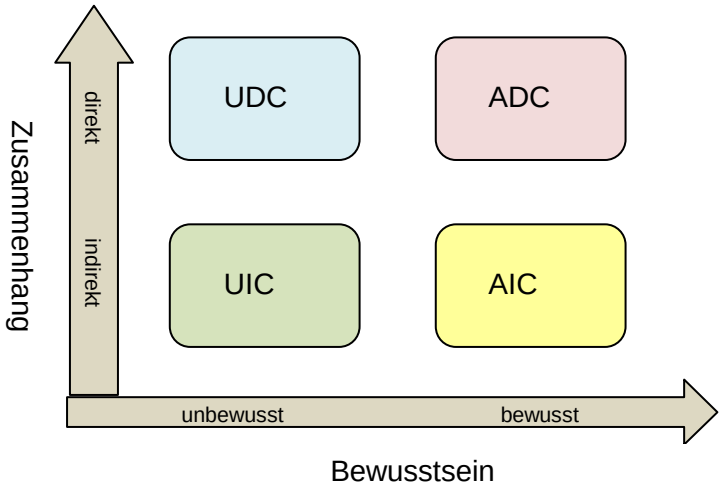

Abbildung 17 - Taxonomie Crowdsourcing

UDC – Unbewusstes direktes Crowdsourcing:

Der Worker ist sich hierbei nicht bewusst, an einem Crowdsourcing-Prozess teilzunehmen. Die von ihm eingebrachten Informationen stehen aber im direkten Zusammenhang mit den benötigten Daten.

Analog werden „Bewusstes direktes Crowdsourcing“ (ADC), „Unbewusstes indirektes Crowdsourcing“ (UIC) und „Bewusstes indirektes Crowdsourcing“ (AIC) verstanden.

Die später betrachteten Beispielen (3.5) sollen einer der hier genannten Kategorien zugeordnet werden. Diese Zuordnung ist allerdings nicht immer eindeutig zu treffen. Diesbezüglich schlagen die Autoren weitere hybride Modelle (nur bewusst bzw. unbewusst, nur direkt bzw. indirekt) vor. [10]

3.3 Arten

Ausgehend vom Crowdsourcing-Gedanken haben sich unterschiedliche Konzepte und Ausprägungsformen entwickelt. Einige davon sollen im Folgenden kurz vorgestellt werden.

3.3.1 Crowdvoting

Beim Crowdvoting wird die Crowd dazu aufgefordert, Bewertungen, Abstimmungen, Meinungen oder Empfehlungen abzugeben. [9] Hierfür finden sich in der Praxis unzählige Beispiele. Angefangen bei einfachen Umfragen zu Lieblingsprodukten, über Hotel-, Reise- oder Buchbewertungen, bis hin zur Wahl des nächsten Superstars.

3.3.2 Crowdfunding

Unter Crowdfunding versteht man das Konzept unterschiedliche Projekte mit Hilfe der Crowd zu finanzieren, wobei jeder nur einen kleinen Beitrag leistet. Beispiele liegen damit klar auf der Hand. Seien es unterschiedlichste Spendenaufrufe, oder die finanzielle Unterstützung unbekannter Musikgruppen (bspw. über sellaband.de). Sogar Wahlkämpfe von US-Politikern lassen sich auf diese Weise finanzieren. [9]

Ein weiteres Beispiel ist die Plattform kickstarter.com. Hier werden mit Hilfe der Crowd unterschiedliche, klar definierte Projekte finanziert. Dabei ist ein gewisser Zeitraum vorgegeben, in dem finanzielle Mittel nachgefragt werden. Sollte die benötigte Summe nicht zustande kommen, so werden die gezahlten Beträge rücküberwiesen. [10]

3.3.3 Crowdpuchasing

Die Idee bei diesem Konzept ist es, gemeinsam mit einer großen Gruppe verschiedene Produkte zu erwerben, mit dem Ziel entsprechende Rabatte zu erhalten. Dies ist zum Beispiel über die Plattform „letsbuyit.de“ möglich. [10]

Neben diesen durchaus relevanten Ausprägungen seien noch zwei weitere Formen genannt, welche im allgemeinen Verständnis eher mit Crowdsourcing in Verbindung gebracht werden.

3.3.4 Crowdwisdom

Dieses Konzept spiegelt am ehesten die Theorie von der „Weisheit der Vielen“ wider. Die Idee ist es, eine große Menge von Informationen zu sammeln, zu organisieren und in einer allgemein gültigen Form darzustellen. [10]

Das wohl populärste Beispiel hierfür ist die freie Wissensplattform Wikipedia.

3.3.5 Crowdworking

In Bezug zur Aufgabenbearbeitung bzw. Problemlösung ist aus Sicht von Unternehmen und aus Sicht der Forschung das Konzept des Crowdworkings von größter Relevanz. Bei diesem Konzept geht es darum eine Aufgabe in viele kleinere Teilaufgaben aufzuteilen, welche dann von der Crowd bearbeitet werden. Dabei handelt es sich in der Regel um „einfache“ Aufgaben, welche sich wenn überhaupt nur schwer durch einen Computer lösen lassen. Als Beispiel sei der Bereich der Mustererkennung genannt.

Dieses Konzept spiegelt demnach, verglichen mit den anderen Konzepten, das eingangs beschriebene Crowdsourcing am besten wider.

Um diese vielen, im einzelnen sehr speziellen, Ausprägungen besser einteilen zu können, bietet sich eine etwas allgemeinere Untergliederung an.

So werden laut dem Crowdsourcing Report 2012 allgemein sechs Formen von Crowdsourcing unterschieden.
Diese sind:

- Microworking
    - Bearbeitung von Kleinstaufgaben wie Tagging, Verschlagwortung, Texterstellung meist gegen eine äußerst geringe Bezahlung vgl. Crowdworking
- Collective Knowledge
    - Alle Formen, die zur Sammlung, Organisation und Filterung von Wissen vgl. Crowdwisdom
- Creative Content-Marktplätze
    - Auslagerung von Kreativprozessen bspw. im Design-Bereich zur Erstellung von Logos, Bannern, Webseiten, etc
- Open Innovation and Ideas
    - Auslagerung von Innovationsprozessen, wo gemeinsam mit der Crowd Problemlösungen und Produktideen erarbeitet werden
- Crowdfunding
    - vgl. 3.3.2
- Engagement und Charity
    - Verfolgung gemeinnütziger (nicht kommerzieller) Zwecke (Bsp: Betterplace.org)

3.4 Motivation der Worker

Die wohl wichtigste Frage bei allen Crowdsourcing-Projekten ist, wie man eine große Masse von Menschen dazu bewegen kann, in ihrer Freizeit freiwillig Aufgaben zu übernehmen.

Dieser Frage soll hier nachgegangen werden. Dafür werden zunächst unterschiedliche Motivationsarten vorgestellt. Anschließend wird anhand von Studien gezeigt, welche Faktoren innerhalb eines Crowdsourcing-Projektes die Motivation der Worker günstig beeinflussen.

Allgemein wird in der Literatur zwischen zwei Motivationsarten unterschieden. Diese werden als extrinsische bzw. intrinsische Motivation bezeichnet.

3.4.1 Extrinsische Motivation

Unter extrinsischer Motivation wird allgemein der Wunsch oder die Absicht verstanden, eine Handlung durchzuführen, um ein lohnendes Ziel zu erreichen (oder eine negative Folge zu vermeiden). Als lohnenswerte Ziele können speziell für Crowdsourcing-Projekte folgende Ziele verstanden werden:

- Bezahlung
- Verbesserung der eigenen Fähigkeiten
- Namentliche Nennung als Teilnehmer
- Sonstige Vorteile oder Privilegien

3.4.2 Intrinsische Motivation

Mit intrinsischer Motivation wird allgemein der Wunsch oder die Absicht bezeichnet, eine Handlung um ihrer selbst willen auszuführen. Gründe die dies veranlassen sind unter anderem:

- Spaß an der Tätigkeit
- Befriedigung persönlicher Interessen
- Neugier
- Identifizierung durch und mit der Tätigkeit
- Feedback durch die Tätigkeit oder von der Gemeinschaft
- Identifikation mit der Gemeinschaft

Eine genauere Darstellung der hier genannten Faktoren findet sich bei [17].

Weiterhin ist die Motivation nicht nur ausschlaggebend, um eine Handlung aufzunehmen. Ebenso wichtig ist es, durch geeignete Motivationsstrategien dafür zu sorgen, dass die Ausführung der Handlung aufrecht erhalten bleibt. So sind beispielsweise regelmäßige Rückmeldungen über die eigene Arbeit, bzw. deren Wert, von Bedeutung.

Darüber hinaus ist es vor allem bei bewusst stattfindendem Crowdsourcing wichtig, den Worker über den Nutzen seiner Tätigkeit, bzw. deren Sinnhaftigkeit, sowie seiner (Mit-)Verantwortung für das Ergebnis aufzuklären.

Das geht insbesondere auf das Modell der Mitarbeitermotivation von Hackman und Oldham (1980) zurück.[2]

Abschließend gilt es festzuhalten, dass die intrinsische Motivation als wesentlich effektiver angesehen wird. Darunter versteht sich, dass intrinsisch motivierten Handlungen gewissenhafter, leidenschaftlicher, also insgesamt engagierter nachgegangen wird. Diese Erkenntnis sollte somit insbesondere beim Entwurf von Crowdsourcing-Anwendungen berücksichtigt werden.

T. Werner und N. Malanowski fassen in ihrer ITA-Kurzstudie [11] die Motive, bzw. die Motivation der Crowd unter den Bezeichnungen: Faszination, Neugierde, Altruismus, User-Benefit, Wettbewerb, Spieltrieb zusammen.

So sorgt ein zu bearbeitendes Thema, welches dem gleichen Betätigungsfeld entspricht, das ein Worker als Hobby ausführt, dafür, dass sich an dieser Tätigkeit mit persönlicher Faszination und Freude beteiligt wird. (Faszination) Weiterhin werden nach Auffassung der Autoren zahlreiche Teilnehmer zum selbstlosen Handeln motiviert, wenn sie in dem Wissen arbeiten, dass sie persönlich einen wichtigen Beitrag zur Lösung eines drängenden gesellschaftlichen Problems leisten. (Altruismus) Als ein weiteres dominantes Motiv wird das Wissen, selbst von einem Dienst oder System zu profitieren, an dem man sich beteiligt hat, angesehen. (User-Benefit)

Bemerkenswert ist, dass fünf der sechs genannten Motive eine intrinsisch motivierte Handlung nach sich ziehen.

Da die Motivation der Worker von solch großer Bedeutung für das Crowdsourcing ist, existieren mehrere Studien darüber, was die Crowd am ehesten dazu veranlasst sich einer gestellten Aufgabe zu widmen. Die Ergebnisse sollen im Folgenden kurz vorgestellt werden.

3.4.3 Externe Untersuchungsergebnisse zur Motivation der Worker

Wie aus den obigen Ausführungen bereits hervor geht, besteht ein starker Zusammenhang zwischen der Motivation der Crowd und dem verfolgten Crowdsourcing-Ansatz. Damit ist insbesondere die Art der Aufgabe, sowie die Darstellung und die Art der Bearbeitung gemeint.

Kaufmann, Schulze, Veit haben in Form einer Befragung untersucht, was die Nutzer von Amazons „Mechanical Turk“ (3.6.1) veranlasst, an den dortigen Projekten teilzunehmen. [17] Innerhalb von drei Tagen wurden 431 gültige Antworten aufgenommen. Die Technik, mit der die Daten ausgewertet wurden, sowie die Bedeutung der angegeben Werte sei an dieser Stelle nicht von Bedeutung.

[2] J. Hackman, G. Oldham: Work Redesign (Organization Development). Upper Saddle River (New Jersy): Prentice Hall, 1980

Rank		Mean	Median	Std. Dev.
1	Payment	3.0151	3.1509	0.6912
2	Task Autonomy	2.4149	2.4708	0.7787
	Skill Variety	2.4111	2.4708	0.7423
4	Task Identity	2.2535	2.2808	0.9175
	Human Capital Advancement	2.2097	2.2808	0.8282
	Pastime	2.0920	2.2808	1.2143
7	Community Identification	2.0407	2.0907	0.8803
	Direct Feedback from Job	2.0048	1.9960	0.7412
9	Signaling	1.8743	1.9006	0.9083
10	Action Significance by Values	1.7019	1.7106	0.8663
	Indirect Feedback from Job	1.6889	1.6478	0.8076
12	Social Contact	1.2882	1.1404	0.9696
13	Action Significance by Norms & Obligations	1.0023	0.7603	0.8324

Abbildung 18 - Umfrageergebnis Crowdsourcing-Motivation

Die Tatsache, dass die Bezahlung den höchsten Stellenwert einnimmt, ist nicht überraschend und dem Umstand geschuldet, dass die Plattform darauf abzielt, dass die Worker von den Auftraggebern bezahlt werden.

Diese Untersuchung zeigt allerdings, dass Begründungen, denen eine intrinsische Motivation zugrunde liegt, stärker bewertet wurden, als die extrinsisch motivierten. Die Autoren zeigen weiterhin, dass die Kategorie „Spaß und Unterhaltung“ besonders hervorsticht, da sich alle damit in Verbindung stehenden Bereiche in der ersten Hälfte des Rankings befinden. Darüber hinaus befinden sich die Bereiche „Verbesserung eigener Fähigkeiten“, „Selbstständigkeit“, „Tätigkeitsvielfalt“ und „Identifizierung mit der Aufgabe“ in derselben Region. Dies zeigt vor allem, wie wichtig die Faktoren sind, die direkt von der Aufgabe abhängen. [17]

Soziale Faktoren waren den Befragten hingegen weniger wichtig. Das liegt womöglich daran, dass die Plattform eher auf eine Kommunikation zwischen Auftraggeber und Worker setzt, als auf eine Kommunikation innerhalb der Community.

Zu einem ähnlichen Ergebnis gelangt auch eine internationale Studie der Plattform 12designer.com. Bei dieser Studie wurden 700 Mitglieder der Community befragt, was sie zur Teilnahme an den Design-Wettbewerben motiviert. Dabei gaben 55% der Befragten als Motivationsfaktor die Möglichkeit an, ihre Design-Kompetenzen zu verbessern. [3]

J. Uhlmann, F. Tommasini, Prof. H.-J. Stark kommen in einer empirischen Untersuchung über die Motivation von Teilnehmern bei der freiwilligen Erfassung von Geodaten, ebenfalls zu ähnlichen Ergebnissen. Die Auswertung von 421 gültigen Umfragebögen ergab zunächst, dass für die überwiegende Mehrheit (82%) der Befragten die Unabhängigkeit der Planung / Durchführung wichtig sei. Weiterhin sind für durchschnittlich 62% der Befragten die soziale Gemeinschaft, für 92% die

[3] www.ptext.de/nachrichten/12designercom-erforscht-internationalen-studie-nutzer-crowdsourcing-angeboten-mo-182145

Tätigkeit selbst (Sammeln von Daten) und für 73% die Arbeit an einem großen Projekt mit anderen Freiwilligen starke motivierende Faktoren.[4]

3.4.4 Gamification

Da festgestellt wurde, dass „Spaß und Unterhaltung“ mit zu den stärksten Motivationsfaktoren zählen, soll an dieser Stelle noch auf sogenannte Gamifications eingegangen werden.

Unter Gamification versteht man den Einsatz von charakteristischen Elementen aus Spielen in spielfremden Prozessen / Anwendungen / Bereichen.[5]

Solche spieltypischen Elemente sind beispielsweise Punkte, Level, Zertifikate, Trophäen, Titel und damit einhergehend natürlich auch ein eigenes Profil / ein eigener Avatar (virtuelle Spielfigur). Bezogen auf das Crowdsourcing könnten so einzelne Microjobs als Missionen (Quests) dargestellt werden, wobei der Spieler (der Worker) selbst entscheiden kann, ob er diese annimmt oder ablehnt. [12] (Entscheidungsautonomie)

Eine ausführliche Untersuchung über den Einsatz von Gamification zur Motivation in Crowdsourcing-Projekten findet sich bei [12].

3.5 Beispiele

Die im Nachfolgenden vorgestellten Beispiele sollen einerseits unterschiedliche Einsatzmöglichkeiten von Crowdsourcing-Anwendungen aufzeigen. Andererseits sollen unterschiedliche Entwurfsstrategien sowie verschiedene Ansätze, welche einem Crowdsourcing-Projekt zugrunde liegen können, sichtbar werden.

3.5.1 Foldit

Aufbauend auf dem reinen Outsourcing-Projekt Folding@Home der Stanford University hat sich das Projekt Foldit (http://fold.it) der University of Washington entwickelt.

Bei Folding@Home wird mit freiwillig zur Verfügung gestellter Hardware der Aufbau von Proteinen, insbesondere deren Faltung, erforscht. Das Programm zur Berechnung fungiert dabei als Bildschirmschoner, d.h. wann immer der betroffene Rechner im Leerlauf ist, wird die Hardware für entsprechende Berechnungen genutzt.

Foldit erweitert dies um die Komponente Mensch, insbesondere seine Fähigkeit zur 3D-Mustererkennung. Bislang konnte lediglich verstanden werden, wann eine Faltung „gut ist“. Der Prozess der Faltung selber, sowie die dazugehörigen Regeln geben allerdings noch Rätsel auf. Die Idee von Foldit ist nun eine Art Puzzle, wo der Mensch die Aufgabe erhält, ein Protein mit spielerischen Mitteln in eine möglichst dreidimensionale Struktur zu bringen. Vorgehensweisen erfolgreicher Spieler werden dann untersucht und man versucht deren Vorgehen in bestehende Software einzubauen. Mit diesem Vorgehen schnitt Foldit bei Wettbewerben gegen mehrere hundert Konkurrenzsysteme, welche rein algorithmisch arbeiteten, stets besser ab. [11]

[4] www.fossgis.de/konferenz/2010/attachments/97_FOSSGIS_2010_Motivation_VGI_Projekte_web.pdf
[5] S. Deterding et. al.: “Gamification: Toward a Definition”, 2011

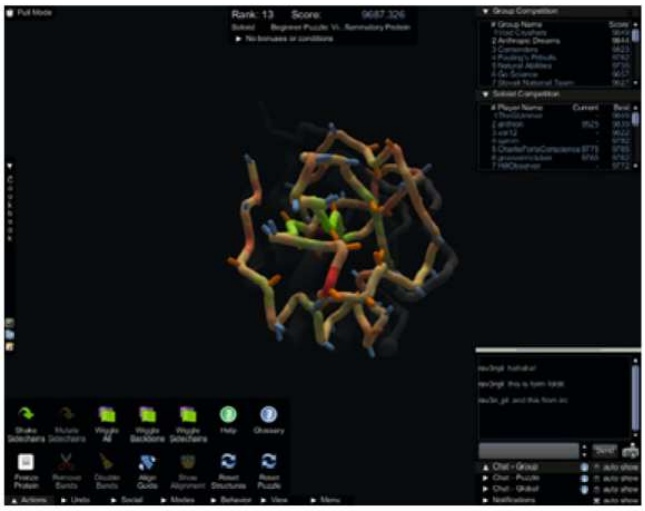

Abbildung 19 - Foldit der University of Washington

Bezogen auf die anfangs vorgestellte Taxonomie lässt sich sagen, dass es sich bei Foldit um bewusstes Crowdsourcing handelt, die Daten aber nur indirekt mit benötigten Informationen zusammenhängen. (AIC)

3.5.2 Air Visibility Monitoring[6]

Amerikanische Wissenschaftler der University of Southern California (USC) haben eine Smartphone-App entwickelt, mit der sich die Feinstaubbelastung beispielsweise in Großstädten messen lässt. Die Idee ist es, dass ein Nutzer mit dieser Application ein Himmelsfoto aufnimmt, welches mit zusätzlichen Informationen (Standort, Zeit) an einen zentralen Server zur Auswertung weitergeleitet wird.

Der Anwender erhält daraufhin eine Auswertung der derzeitigen Belastung in seinem Umfeld. [11] Dieser Ansatz verfolgt mehrere Ziele. Einerseits werden relevante Daten zur Feinstaubbelastung erhoben. Weiterhin wird dadurch ein flächendeckendes „Sensornetzwerk“, mit nahezu keinen Kosten, aufgebaut. Weiterhin werden nur Daten erhoben, die den Menschen direkt betreffen, also keine „unnützen“ Daten von unbewohnten Gebieten. Andererseits erfolgt eine nahezu Echtzeit-Information der derzeitigen Situation an die Bürger.

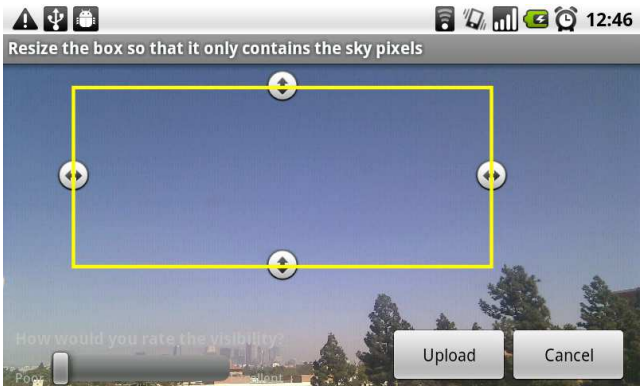

Abbildung 20 - Userinterface von Air Visibility Monitoring

Bemerkenswert bei diesem Ansatz ist die gegenseitige Win-Win-Situation.

[6] http://robotics.usc.edu/~mobilesensing/Projects/AirVisibilityMonitoring

Eine Einteilung in das vorgeschlagene Klassifikationsschema gestaltet sich hier etwas schwierig. Definitiv lässt sich festhalten, dass die eingebrachten Daten in direktem Zusammenhang mit den erforderlichen Informationen stehen. Ob der Nutzer nun in dem Bewusstsein handelt, diese Daten für eine flächendeckende Untersuchung bereitzustellen, oder lediglich an seinem direkten Nutzen interessiert ist, ist fraglich. Somit lassen sich sowohl Kriterien für bewusstes direktes Crowdsourcing (ADC) und unbewusstes direktes Crowdsourcing (UDC) finden.

Dieser Ansatz eines mobilen Sensornetzwerkes findet sich auch in anderen Anwendungen. „Flu Data goes mobile“ von der Harvard Medical School ermöglicht es dem Benutzer lokale Krankheitsfälle von Infektionskrankheiten (bspw. Schweinegrippe) zu melden. Forscher können dann durch mehrere Meldungen die Verbreitung einer Krankheit einschätzen oder ggf. verifizieren. Als Gegenleistung erhalten die Nutzer Informationen über Krankheitsfälle in ihrer Umgebung.

Dieser Ansatz ist insbesondere von Bedeutung, weil das so gewonnene dynamische Sensornetzwerk unmittelbar an das menschliche Schwarmverhalten gekoppelt ist. [11]

3.5.3 Perseid Project[7]

Der Grundgedanke bei diesem Projekt besteht darin, dass eine geografisch weit verteilte Crowd identische Ereignisse aus unterschiedlichen Perspektiven erfasst. Diese unterschiedlichen Perspektiven werden dann zu einem konsistenten Gesamtbild zusammengefügt. Das bedeutet also, dass durch ein großräumiges Netzwerk Beobachter ihre Beobachtungen mit anderen Teilnehmern synchronisieren können, wodurch ein räumliches Modell entsteht.

Diesen Ansatz verfolgte Chris Crawford 2010 bei seinem Perseid Project. Dabei sollten die weit verteilten Teilnehmer die nördliche Hemisphäre beobachten und bei jedem Auftauchen einer Sternschnuppe eine Taste auf ihrem mobilen Endgerät drücken. Aus den überlappenden Zeitstempeln und den unterschiedlichen geografischen Informationen konnte so eine dreidimensionale Karte des Sternschnuppen-Schauers angefertigt werden. [11]

Bei diesem Ansatz findet offensichtlich ein bewusstes Crowdsourcing statt, bei dem die eingebrachten Daten mit den erforderlichen Informationen in direktem Zusammenhang stehen. (ADC)

3.5.4 Google Image Labeler

Der Google Image Labeler ist eine Anwendung, mit der Schlagwörter für Bilder hinterlegt werden. Dies erst ermöglicht die heute bekannte Bildersuche im Internet.

Dabei wird zwei einander unbekannten Menschen das gleiche Bild gezeigt. Beide sollen dann (getrennt voneinander) das Bild in mehreren isolierten Wörtern möglichst schnell (intuitiv) beschreiben. Je mehr identische Wörter die beiden Spieler genannt haben, desto mehr Punkte werden ihrem Punktekonto gutgeschrieben. [11]

[7] http://www.centauri-dreams.org/?p=13831

Dabei ist offensichtlich, dass ein Wort, welches von vielen Spielern genannt wird, den Inhalt eines Bildes am besten beschreibt.

Abbildung 21 - Google Image Labeler

Es ist davon auszugehen, dass es sich hierbei um ein unbewusst stattfindendes Crowdsourcing handelt. Ferner stehen die eingebrachten Daten in direktem Zusammenhang, mit den benötigten Informationen. (UDC)

3.5.5 reCAPTCHA[8]

CAPTCHAs sind kleine meist grafische Tests, um menschliche User von Bots zu unterscheiden. Diese Aufgaben im Bereich der Mustererkennung sind für Menschen, im Gegensatz zum Computer, keine große Herausforderung.

Die Idee ist es nun, diesen Verifikationsprozess mit einer gewinnbringenden Aufgabe zu verbinden. Dabei wird dem Benutzer ein unbekanntes Wort vorgelegt, bei dem Texterkennungsalgorithmen bislang versagt haben. Derartige Wörter stammen beispielsweise aus eingescannten historischen Schriften. Die Fehlerrate wird dahingehend minimiert, dass ein solches „Artefakt“ gleichzeitig mehreren Benutzern vorgelegt wird. Eine hohe Übereinstimmung bei der Übersetzung lässt dann eine korrekte Erkennung vermuten. [11]

Bis heute wurden so ca. 500 Millionen für den Computer unlesbare Wörter entschlüsselt. [11]

Der eigentliche Ansatz liegt also darin, sehr kleine, für den Menschen schnell lösbare Aufgaben zur Verifikation von Usern (vs. Bots) einzusetzen.

Da die Crowd im Allgemeinen nicht weiß, dass hierbei ein Crowdsourcing-Prozess stattfindet, lässt sich dieser Ansatz der Kategorie UDC zuordnen.

Neben diesen hier genannten Beispielen finden sich bei [11] noch weitere, erläuterte Anwendungsgebiete.

[8] www.google.com/recaptcha

3.6 Plattformen

3.6.1 Amazon Mechanical Turk[9]

Amazon Mechanical Turk (MTurk) ist eine Plattform, auf der eine kommerzielle Auslegung des Crowdsourcing verfolgt wird. Ein Auftraggeber stellt dort Microjobs ein, die mit einem PC über das Internet bearbeitet werden können. Bei diesen Microjobs handelt es sich klassischer Weise um Aufgaben, die sich nicht durch einen Computer erledigen lassen. Die Arbeitsaufgabe ist dabei extrem einfach gehalten und klar definiert. Sie bedarf also keiner weiteren Einarbeitung. Pro gelöster Aufgabe (Hit – Human Intelligence Task), kann ein Auftragnehmer zwischen $0,01 und $30, abhängig vom Aufwand, verdienen. [11]

Der Auftraggeber bezahlt ferner eine Gebühr an Amazon, abhängig von der Bezahlung der Arbeit, sowie der Anzahl der erfüllten Aufgaben. [11]

3.6.2 Clickworker[10]

Die deutsche Plattform Clickworker (Teil der humangrid GmbH - Dortmund) funktioniert ähnlich wie Amazon MTurk. Auftragnehmer nehmen verschiedene Microjobs an und werden dafür entlohnt.

In Abgrenzung zu MTurk besitzt hier der Auftraggeber allerdings die Möglichkeit, ein komplettes Projekt, bzw. umfangreiche Aufträge an die humangrid GmbH weiterzuleiten. Diese Aufträge werden dann von den Betreibern der Plattform selbst in kleine Microjobs zerlegt und zur Bearbeitung online gestellt. Ebenfalls setzen die Betreiber die zerlegten Projekte wieder zusammen und senden das komplette Ergebnis an den Auftraggeber zurück. [11]

Ein weiterer Unterschied zu MTurk besteht darin, dass ein Auftragnehmer (Clickworker) nicht jeden beliebigen Auftrag annehmen kann. Clickworker wählt die Worker entsprechend ihrer Qualifikationen aus. Diese Qualifikationen gibt jeder Clickworker bei seiner Registrierung an oder erwirbt sie durch spezielle Trainings der Plattform. Weiterhin verifizieren die Plattforminhaber diese angegeben Qualifikationen durch selbige Trainings. [11]

[9] https://www.mturk.com/mturk/welcome
[10] www.clickworker.com

Folgende Abbildung zeigt das Arbeitsprinzip der Plattform[11].

Abbildung 22 - Workflow der Clickworker-Plattform

3.6.3 12designer[12]

Bei der deutschen Plattform 12designer, welche mittlerweile vom amerikanischen Konkurrenten 99design übernommen wurde, handelt es sich um einen Marktplatz zur Umsetzung von Designaufgaben.

Dabei wird eine Designaufgabe (Logo, Webseiten, Flyer, etc.) mit einem ausgeschriebenem Preisgeld von einem Auftraggeber online gestellt. In Form eines offenen Ideenwettbewerbes geben Mitglieder der Crowd (Profi-Designer, Amateur-Designer, sonstige kreativ veranlagte User) ihre Entwürfe ab. Der Auftraggeber wählt dann den Gewinner aus, welcher dementsprechend das Preisgeld erhält.

Interessant ist dabei, dass die Rückmeldung über den eigenen Entwurf, sowie die Verbesserung eigener Fähigkeiten, durch die Entwürfe der anderen, ebenso große Motivationsfaktoren sind, wie der finanzielle Aspekt.

3.6.4 Die Open Source Plattform PyBossa

Für den Zeck der Forschung ist es von zentraler Bedeutung eine freie, öffentlich zugängliche, Plattform zu finden, die es ermöglicht Crowdsourcing-Aufgaben an eine echte Community zu stellen. Dadurch kann im Speziellen untersucht werden, wie sich die Crowd bei einzelnen Aufgaben verhält, welche Aufgabenpräsentation angemessen ist und ob der Crowdsourcing-Prozess die gewünschten Resultate liefert.

[11] http://www.clickworker.com/das-clickworker-prinzip
[12] http://www.12designer.com/de/

Für den dieser Arbeit zugrunde liegenden Bereich der Datenintegration verspricht die Plattform "PyBossa" genau dies zu leisten. Hierbei handelt es sich um eine Plattform für Microjobs ähnlich wie Amazon MTurk. Derzeit sind auf dieser Plattform[13] 2351 (Stand 20.10.2013) Mitglieder registriert, wobei eine Registrierung nur für die Erstellung bzw. Veröffentlichung von Projekten notwendig ist. Um an Projekten mitzuarbeiten ist dies in der Regel nicht nötig.

Um ein eigenes Projekt zu erstellen sind im Wesentlichen drei Schritte notwendig:

1. Anlegen eines Projektrahmens
2. Importieren / Erzeugen der Aufgaben (Tasks) über einen Task Creator
3. Darstellen der Aufgaben über einen Task Presenter

Für diese drei Schritte stellt die Plattform ein Web Interface, sowie eine API zur Verfügung. Um einen neuen Projektrahmen anzulegen sind wenigstens drei Informationen notwendig: Name des Projektes, Kurzform, sowie eine kurze Beschreibung. Zusätzlich kann definiert werden, ob unregistrierte Nutzer an dem Projekt teilnehmen dürfen und in welcher Kategorie (Science, Social, Art, etc.) das Projekt angesiedelt ist. Das komplette, dafür zur Verfügung stehende Web-Formular ist in Abbildung 23 dargestellt.

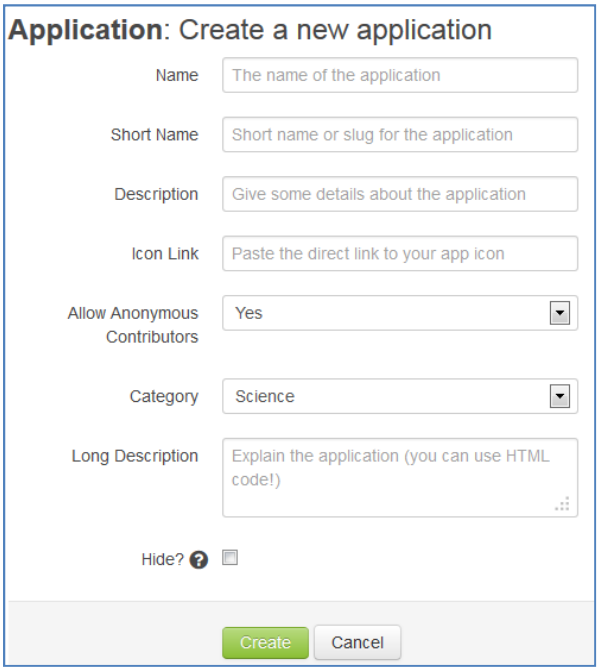

Abbildung 23 - Web Interface, create application

Beim Task Creator handelt es sich um ein Python Skript, welches die Aufgaben erzeugt. Über das Web Interface werden unterschiedliche Möglichkeiten, die Daten zu importieren, unterstützt. Beispielsweise kann ein CSV-File (comma-seperated values), welches sich in einer DropBox oder ähnlichem befindet, importiert werden. Beim Task Presenter handelt es sich um HTML Seite mit Javascript Elementen, um die Aufgaben dem Nutzer zu präsentieren und dessen Antworten zu

[13] www.crowdcrafting.org

speichern. Eine vollständige Beschreibung sowie ein Step-by-Step-Tutorial findet sich in der ausführlichen PyBossa Dokumentation.[14, 15]

Die Plattform ermöglicht es weiterhin festzulegen, wie die einzelnen Aufgaben abgearbeitet werden (Task Scheduler) und welche Priorität unterschiedliche Aufgaben besitzen (Task Priority). Weiterhin kann definiert werden, wie viel Antworten für jede Aufgabe abgegeben werden müssen (Task Redundancy). Diese Einstellungen können wieder über die API oder das Web Interface vorgenommen werden. Abbildung 24 zeigt das Web Interface.

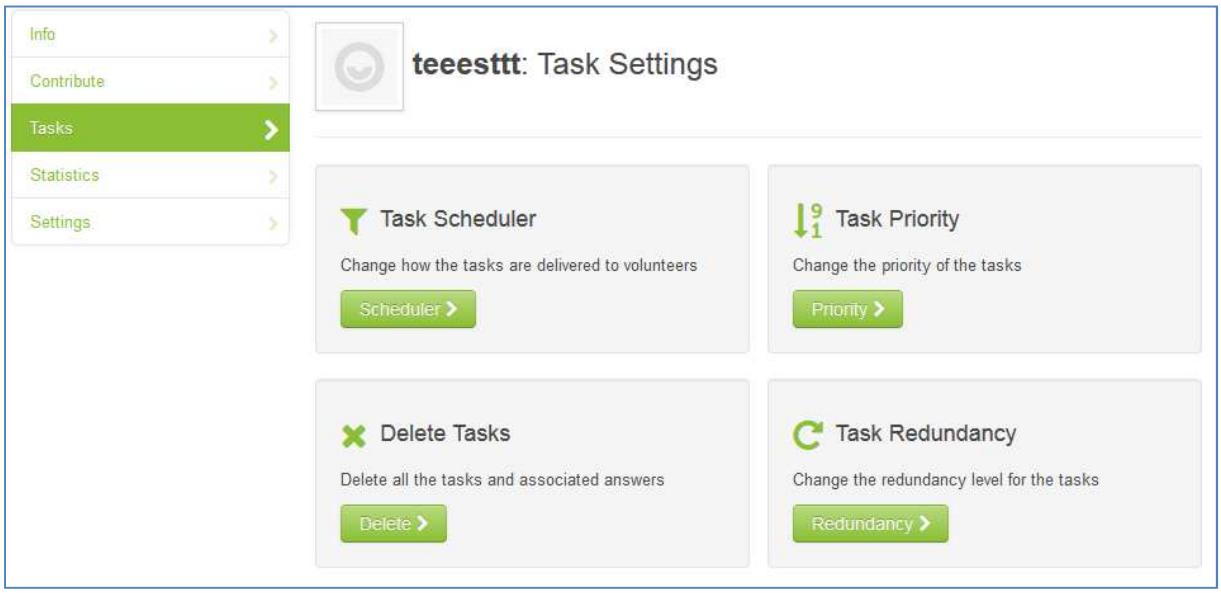

Abbildung 24 - Web Interface, task settings

Weiterhin steht der Quellcode von mehreren Demo-Projekten zur Wiederverwendung zur Verfügung.[16] Darüber hinaus besteht die Möglichkeit die Plattform auf einem eigenen GNU/Linux-Server laufen zu lassen. Die Installationsanleitung sowie alle wichtigen Hinweise zur Administration finden sich in der Dokumentation.[17]

Die gesammelten Ergebnisse des eigenen oder anderer Projekte lassen sich jederzeit im CSV oder JSON Format exportieren. Das heißt es besteht die Möglichkeit für weiterführende Prozesse die bisherigen Zwischenergebnisse zu verwenden.

[14] http://docs.pybossa.com/en/latest/
[15] http://docs.pybossa.com/en/latest/user/tutorial.html
[16] http://crowdcrafting.org/app/flickrperson/
http://crowdcrafting.org/app/pdftabletranscribe/
http://crowdcrafting.org/app/urbanpark/
http://crowdcrafting.org/app/mapknitter/

[17] http://docs.pybossa.com/en/latest/installing_pybossa.html

Abbildung 25 - Web Interface, Task-Status

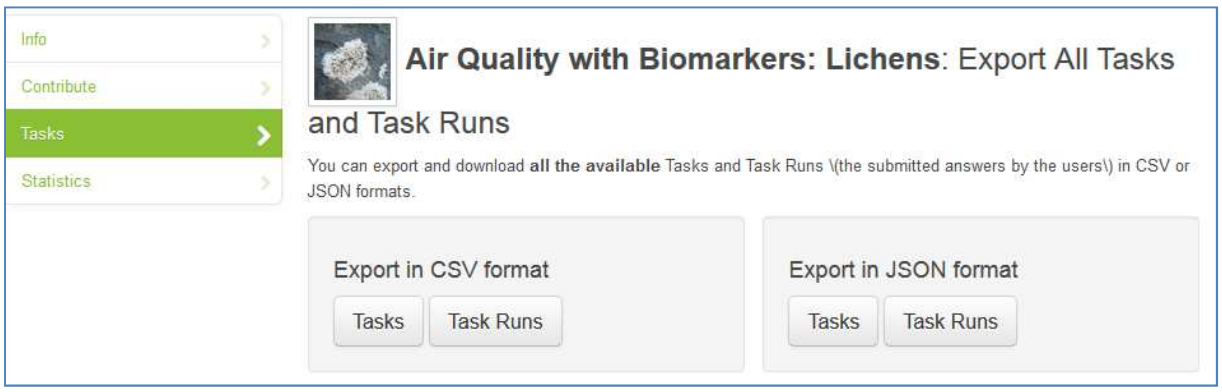

Abbildung 26 - Web Interface, Task Export

Derzeit sind hauptsächlich Projekte angelegt, welche die menschliche Fähigkeit der Bilderkennung, Mustererkennung oder spezielles Domänenwissen erfordern.[18]

Beispiele:

- Shell JIV Transcription:[19]
  Hierbei handelt es sich um eine Erfassung von Daten aus Meldungen von Zwischenfällen. Dabei steht der Meldungsbericht in Form eines PDF's zur Verfügung. Aufgabe des Users ist

[18] http://crowdcrafting.org/app/category/featured/
[19] http://crowdcrafting.org/app/shelltranscription/

es nun, die darin enthaltenen Längen und Breitenangaben zu erfassen.

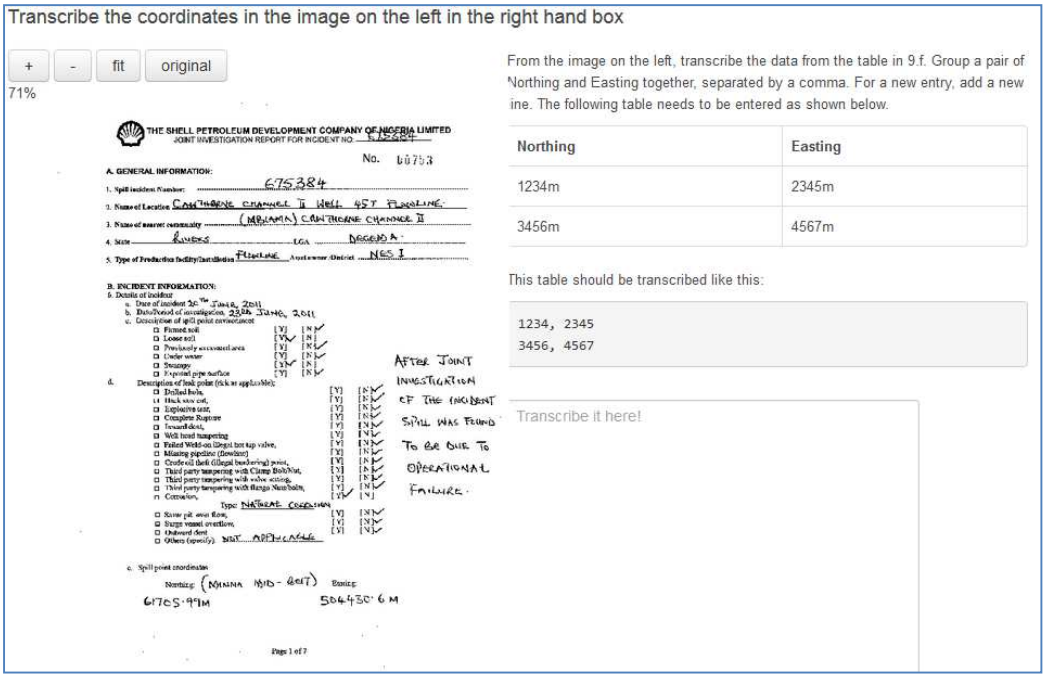

Abbildung 27 - Shell JIV Transcription

- Feynman's flowers:[20]

  Bei diesem Projekt des London Centre for Nanotechnology besteht die Aufgabe darin, den Mittelpunkt und die Axenausrichtung von Molekülen zu bestimmen.

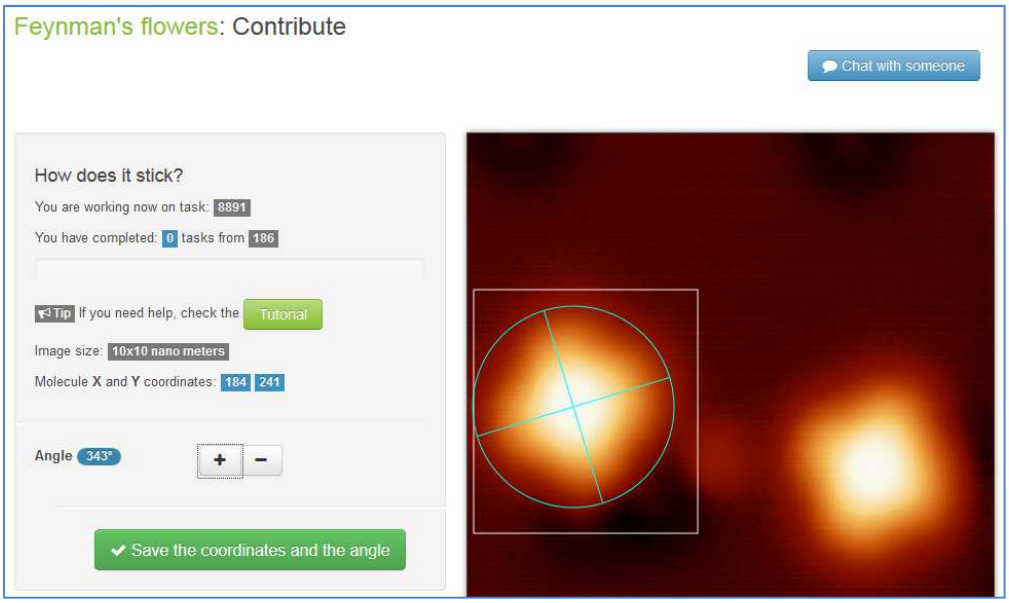

Abbildung 28 - Feynman's flowers

[20] http://crowdcrafting.org/app/feynmanflowers/

- Understand the meaning of words:[21]
  Die Aufgabe bei diesem Projekt besteht zunächst darin einen vorgegeben Satz zu verstehen. Danach sollen einzelne Textbestandteile einem definierten Begriff zugeordnet werden.
  Bei dem Satz: "*The flood exterminated the rats by cutting off access to food.*" muss also beispielsweise dem definierten Begriff "*victim: the living entity that dies as a result of the killing.*" der Textbestandteil "*the rats*" zugeordnet werden, sofern dieser zur Auswahl vorgegeben ist.

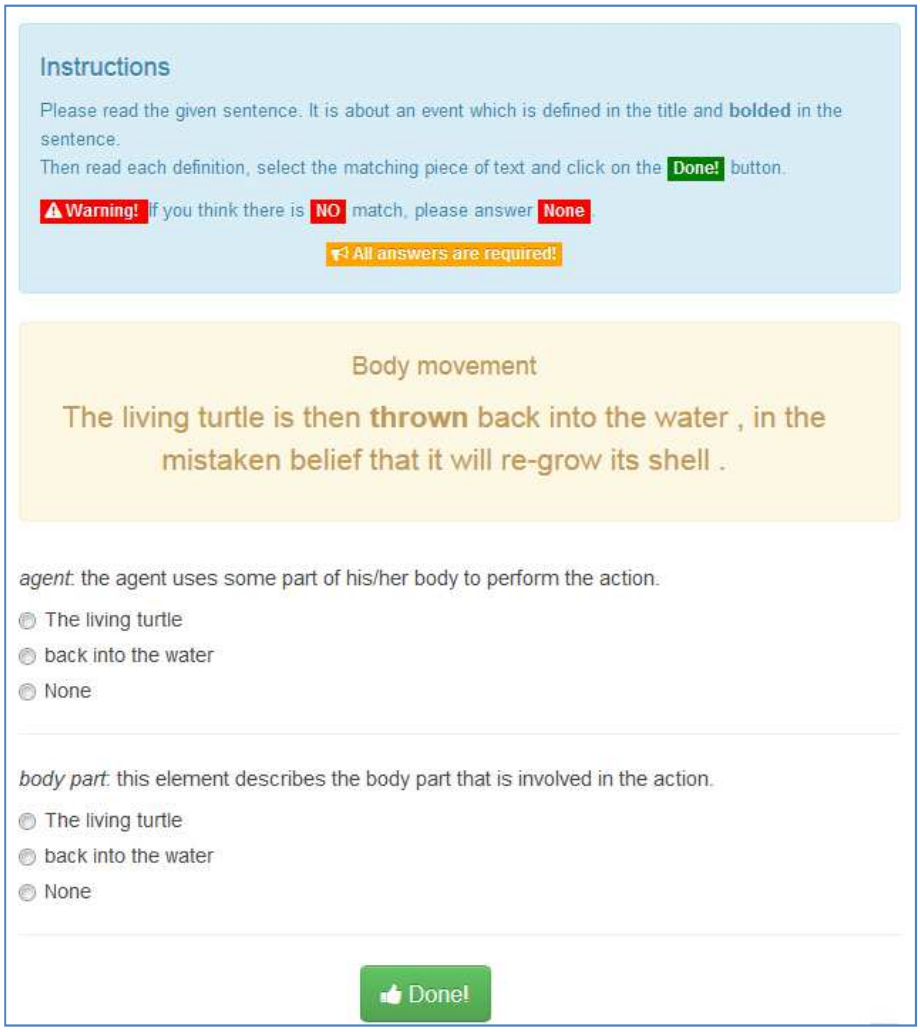

Abbildung 29 - Understand the meaning of words

Diese ausgewählten Beispiele zeigen die Mächtigkeit dieser Plattform. Ferner können unterschiedliche Aspekte der Motivation bedient werden. So kann bei der Task-Erstellung eine umfassende Aufklärung über den Nutzen und die Relevanz der Arbeit vorgenommen werden (vgl. Abbildung 23, Long Description). Zudem steht eine umfassende statistische Auswertung des Arbeitsprozesses zur Verfügung, wo unter anderem die aktivsten User vorgestellt werden.

[21] http://crowdcrafting.org/app/semroles/

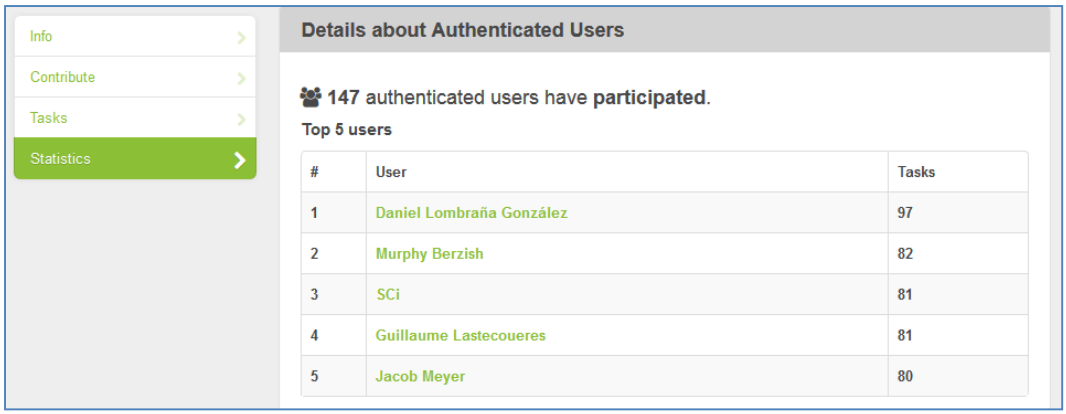

Abbildung 30 - Top 5 User

Die Plattform erscheint im Allgemeinen für Crowdsourcing-Projekte im Bereich der Datenintegration geeignet. Wie eine spezielle Implementierung eines solchen Projektes aussehen könnte bleibt an dieser Stelle offen.

3.6.5 Weitere Plattformen

Eine Auflistung von insgesamt 53 Crowdsourcing-Plattformen ist beispielsweise unter http://grassgreenmedia.com/crowdsourcing-plattformen-in-der-ubersicht/ zu finden.

Die Plattformen werden hierbei nach den Modellen von Papsdorf unterschieden. [21] Dieser unterscheidet zwischen „offenen Ideenwettbewerb“, „ergebnisorientierte, virtuelle Microjobs“, „userdesignbasierte Massenfertigung“, „userkollaboration basierende Ideenplattform“ und „indirekter Vernutzung von Usercontent“.[22]

[22] http://grassgreenmedia.com/welche-crowdsourcing-modelle-gibt-es/

3.7 Zusammenfassung Crowdsourcing

Crowdsourcing ist ein moderner Begriff, welcher bedingt durch unterschiedliche Sichtweisen nicht einheitlich definiert ist. Allgemein verstehen wir unter Crowdsourcing eine Bearbeitung unterschiedlichster Aufgaben durch eine große, geografisch weit verteilte Masse von Menschen. Ermöglicht wird dies erst durch moderne Kommunikationsmedien, dem Web2.0.

Gründe für das Crowdsourcing gibt es eine Menge. Sei es die Kompensation mangelnder Rechenzeit, die Reduktion hoher Kosten oder die Nutzung spezieller menschlicher Fähigkeiten zur Bewältigung maschinell unlösbarer Aufgaben.

Die wichtigste Frage hierbei ist die, wie man eine Vielzahl von Menschen dazu bewegen kann, bzw. was die Menschen dazu motiviert an solchen Aufgaben teilzunehmen. Es hat sich gezeigt, dass neben finanziellen Gründen auch Spaß an der Arbeit, Neugierde, soziale Komponenten oder die Chance sich weiterzubilden starke Motivationsfaktoren sind.

Somit ist davon auszugehen, dass Anwendungen, die gezielt auf diese Faktoren aufbauen besonders erfolgreich sind. Insbesondere scheinen Elemente, welche aus Rollenspielen etc. bekannt sind, wie Level, Trophäen usw., von großer Bedeutung zu sein. (sog. Gamification)

Weitere wichtige Komponenten, die aus Umfragen hervorgehen, sind die Selbstbestimmung der Arbeit, das Wissen über den Nutzen der Tätigkeit und die Rückmeldung über die eigene Leistung.

Insgesamt sollten also folgende Punkte berücksichtigt werden:

- Profil für den Worker zur persönlichen Gestaltung
- Vergabe von Punkten, Medaillen, Level, etc. (Gamification)
- Anzeige über aktuellen Bearbeitungsfortschritt
- Möglichkeit frei zu entscheiden welche Aufgabe bearbeitet wird
- Auflistung der Top-Worker aufgabenbezogen und allgemein (virtuelle Ruhmeshalle)
    - Rückmeldung über die erbrachte Leistung
- ansprechendes Layout
- Aufklärung über Nutzen / Sinnhaftigkeit der Arbeit

4 Crowdsourcing im Bereich der Datenintegration

Im weiteren Verlauf der Arbeit soll untersucht werden, wie sich die unterschiedlichen Crowdsourcing-Ansätze auf die Probleme der Datenintegration übertragen lassen. Dafür werden zunächst die verschiedenen Probleme genau formuliert. Anschließend werden verschiedene Crowdsourcing-Ansätze zur Bearbeitung dieser Probleme diskutiert. Abschließend werden einzelne Methoden zur Vor- und Nachbereitung der Daten vorgestellt. Die Arbeit schließt mit einem Fazit, welches nochmal die zentralen Erkenntnisse zusammenfasst und Anreize für weitere Arbeiten liefert.

4.1 Hauptprobleme der Datenintegration

Wie aus Kapitel 2 hervor geht, tauchen während der Datenintegration unterschiedliche Probleme auf. Dabei bietet es sich an, im Folgenden zwischen Problemen im Bereich des Schema-Matchings, sowie Problemen im Bereich des Object-Matchings zu differenzieren.

4.1.1 Probleme beim Schema-Matching

Die Probleme im Bereich des Schema-Matchings resultieren hauptsächlich aus der starken Heterogenität unterschiedlicher Datenquellen. Insbesondere werden gleiche Sachverhalte aus der Realität in unterschiedlicher Art und Weise modelliert. Daraus resultiert eine hohe Variabilität, was die Anzahl und Wahl der Attribute, deren Verteilung auf Relationen und die Beziehungen der Relationen untereinander betrifft.

Um nun eine Abbildung eines Modells auf ein anderes zu erhalten, ist vor allem die Kenntnis von der Semantik der Attribute bedeutsam. Diese ist allerdings nicht unbedingt offensichtlich. In der Regel wird an dieser Stelle kontextbezogenes Expertenwissen benötigt.

4.1.2 Probleme beim Object-Matching

Im Bereich des Object-Matchings, also der Duplikaterkennung, kommt es zu Schwierigkeiten, wenn für dieselben realen Objekte unterschiedliche Beschreibungen, bzw. Bezeichnungen, verwendet werden. Erschwerend kommt hinzu, dass im Prozess der Duplikaterkennung eine erhebliche Menge an Datensätzen geprüft werden muss. Die Entscheidung, ob es sich bei zwei Datensätzen um ein Duplikat, also um ein und dasselbe reale Objekt handelt, stellt für den Menschen keine besondere Herausforderung dar. Für eine maschinelle Bearbeitung ist es allerdings nicht möglich immer korrekt zu entscheiden, ob es sich bei einer Beschreibung beispielsweise nur um eine Formulierung mit Zusatzinformationen, oder um ein gänzlich anderes Objekt handelt.

4.1.3 Einsatzmöglichkeiten von Crowdsourcing

Aus den formulierten Problemen und der Kenntnis derzeitiger Ansätze wie dem maschinellen Lernen, lassen sich mehrere Einsatzmöglichkeiten des Crowdsourcings für die Datenintegration ableiten.

Somit kann einerseits eine Verifikation eines bereits bestehenden oder neu entwickelten Systems ermöglicht werden, indem die Crowd befragt wird, wie gut die berechneten Ergebnisse einander matchen.
Weiterhin können durch die Crowd-Worker Trainingsdaten für überwachte Lernverfahren generiert werden. Unter Trainingsdaten versteht man hierbei eine Menge „perfekt gematchter" Datensätze. Bedingt durch die Tatsache, dass an dieser Generierung mehrere, bis viele, Worker beteiligt sind, kann dieser Prozess relativ effizient ablaufen, wenn in kurzer Zeit viele verschiedene Daten generiert werden.
Ein ähnlicher Aspekt ist die Erstellung sogenannter Benchmarks durch die Crowd. Dabei handelt es sich um ein vollständiges Matching mehrerer Schemata, welches als Standard angenommen wird. Maschinell generierte Lösungen werden dann mit diesen Benchmarks verglichen, was eine Beurteilung der verwendeten Verfahren ermöglicht. Letztlich kann die Arbeit der Crowd-Worker an all den Stellen zum Einsatz kommen, wo menschliche Entscheidungen gefragt sind. Sei es die Bestätigung bzw. die Vervollständigung von Matchergebnissen, oder die Auswahl der besseren Strategiekombination.

4.2 Mögliche Ansätze

Bei den einzelnen Crowdsourcing-Ansätzen gilt es zunächst zu unterscheiden, ob prinzipiell eine Bezahlung der Crowd-Worker vorgesehen ist, oder nicht. Diese Entscheidung hängt maßgeblich vom Hintergrund der Datenintegrationsaufgabe ab. Handelt es sich beispielsweise um einen einmaligen (außergewöhnlichen) Vorgang im Rahmen einer Unternehmensübernahme, so ist eine Bezahlung der Worker, also ein Ansatz mit Paid-Crowdsourcing über eine entsprechende Plattform (3.6), aus mehreren Gründen sinnvoll. Einerseits wird auf diese Weise eine verhältnismäßig schnelle Bearbeitung, andererseits ein gewisser Qualitätsstandart durch die Plattform erreicht.
Für Aufgaben, denen ein wesentlich höheres Datenvolumen zugrunde liegt, oder Aufgaben, die sich regelmäßig wiederholen, sowie gemeinnützige Aufgaben, wie die Erstellung eines universellen Produkt-Kataloges oder die Verbesserung von Suchergebnissen, oder für Aufgaben aus der Forschung sind andere Ansätze ebenso relevant. Darunter verstehen sich Ansätze wie:

- die Entwicklung von Minispielen oder Rollenspielen
    - insbesondere die Verwendung von Gamifications
    - Minispiele wie Memory
- die Nutzung von Microtasks zur Identifikation von Bots
    - vergleichbar mit dem reCAPTCHA-Projekt

Der Ansatz, der auf die Schaffung einer gegenseitigen Win-Win-Situation abzielt (vgl. 3.5.2 Air Visibility Monitoring), lässt sich nicht auf den Bereich der Datenintegration übertragen. Auch wenn der Nutzer von eventuell optimierten Suchanfragen bei E-Bay, Amazon, etc. profitiert, so bleibt der

unmittelbare, sofortige Nutzen für den Anwender aus, was den Ansatz in dieser Hinsicht nicht praktikabel macht.

4.3 Probleme beim Einsatz von Crowdsourcing

Durch den Einsatz von Crowdsourcing-Techniken eröffnet sich die Möglichkeit die für den Computer nicht eindeutig lösbaren Probleme verhältnismäßig schnell zu bearbeiten. Allerdings treten mit dem Einsatz von Crowdsourcing neue Probleme auf. So ist zunächst festzuhalten, dass die Entscheidungen der Crowd-Worker grundsätzlich nicht immer korrekt sind. Sei es aus Unwissenheit, Böswilligkeit oder einfach aus Versehen. Für den Spiel-Ansatz und den Verifizierungs-Ansatz, ergibt sich sofort die Frage, wie (automatisch und ohne Kenntnis der Lösung) entschieden werden soll, ob die Eingabe des Anwenders korrekt ist. Ob er also Punkte im Spiel, oder Zugang zu einer Anwendung erhält, da er als „kein Bot“ identifiziert wurde.

Zur Auflösung dieser Probleme wird im Folgenden eine Idee geliefert. Es sei aber gesagt, dass es gewiss unterschiedliche Möglichkeiten gibt, die genannten Probleme zu lösen.

4.3.1 Lösungsansatz

Um User, welche böswillig handeln, also bewusst falsche Antworten auswählen, auszuschließen, bietet es sich an, mehrere Verifizierungsaufgaben voran zu stellen. Das sind Aufgaben, die sich rein äußerlich nicht von den eigentlichen Aufgaben unterscheiden. Jedoch ist die korrekte Lösung dieser Aufgaben bereits bekannt. Um böswillige Worker nun auszuschließen, werden nur diejenigen zugelassen, die alle Verifizierungsaufgaben korrekt bearbeitet haben. Diese oder ähnliche Methoden werden von Plattformen wie Amazon MTurk bereits verwendet. [20]

Inkorrekte Antworten, die unbeabsichtigt, oder aus Unwissenheit heraus entstehen, sorgen dafür, dass einzelne Aufgaben im Allgemeinen mehrfach und an unterschiedliche Worker gestellt werden müssen.

Die Antwort auf die Frage, wie ohne Kenntnis der Lösung entscheiden werden kann, ob eine Nutzereingabe korrekt oder inkorrekt ist, liefert das reCAPTCHA-Beispiel (3.5.5). Wird ein und dieselbe Aufgabe nahezu zeitgleich, von hinreichend vielen Nutzern bearbeitet, so kann man grundsätzlich davon ausgehen, dass die Lösung, welche am häufigsten gewählt wurde, die korrekte Lösung ist.

Andere Ansätze könnten auf den Ideen gründen, dass die Crowd zur Bewertung der Crowdsourcing-Lösungen herangezogen wird, da die Einschätzung fremder Arbeit meist kritischer geschieht, als die Einschätzung der eigenen Arbeit. Ebenfalls vielsprechend ist der Ansatz, dass nur dann Punkte erzielt werden können, wenn die Lösung mit der eines anderen Workers übereinstimmt. (vgl. 3.5.4)

4.4 Vorbereitung / Nachbereitung von Informationen

Aus der Problemformulierung unter 4.1 wird ersichtlich, dass die Informationen, welche von der Crowd bearbeitet werden sollen, in gewisser Art und Weise vorbereitet werden müssen. So ist es beispielsweise notwendig, die Informationsfülle entsprechend zu reduzieren und nur die für das Matching signifikanten Informationen auszuwählen. Insbesondere erhöht das die Zugänglichkeit der Arbeit vor allem für Nicht-Experten.

Darüber hinaus ist zu klären, wie mit den gewonnenen Daten weiter verfahren werden soll. Da es notwendig ist, gleiche Aufgaben mehrfach zu stellen, liegen natürlicherweise keine eindeutigen Ergebnisse vor.

Wie diese Vorbereitung, bzw. Weiterverarbeitung, vollzogen werden kann, ist Gegenstand zahlreicher Forschungen. Im Weiteren werden zwei Forschungsarbeiten aufgegriffen und die Idee, sowie die gewonnenen Erkenntnisse dargestellt. Dafür werden die beiden Bereiche Schema-Matching und Object-Matching separat betrachtet. An dieser Stelle ist noch zu erwähnen, dass die vorgestellten Ansätze im Bereich des Paid-Crowdsourcing angesiedelt sind. Die Erkenntnisse lassen sich aber auch für andere Crowdsourcing-Anwendungen gewinnbringend verwenden.

4.4.1 Allgemeine Vorgehensweise

Die naive Vorgehensweise, der Crowd alle Daten zur Verfügung zu stellen, bzw. von der Crowd alle möglichen Match-Konstellationen prüfen zu lassen, ist in vielerlei Hinsicht nicht praktikabel. So müssen Paare, die offensichtlich nicht als Match in Frage kommen, nicht an die Crowd weiter gereicht werden. Insbesondere im Bereich der Deduplizierung ist es notwendig durch eine maschinelle Vorverarbeitung das Datenvolumen (Kreuzprodukt über der Menge der Datensätze) drastisch zu reduzieren, um einerseits Kosten zu sparen und andererseits eine schnellere Auswertung zu erhalten.

Somit besteht der grundlegende Ansatz darin, maschinelle Lösungen und Crowdsourcing zu kombinieren. Es werden also nur Daten an die Crowd weitergegeben, die nach einer maschinellen Auswertung als mögliche Match-Kandidaten erkannt wurden. Schematisch lässt sich diese allgemeine Arbeitsweise wie folgt darstellen:

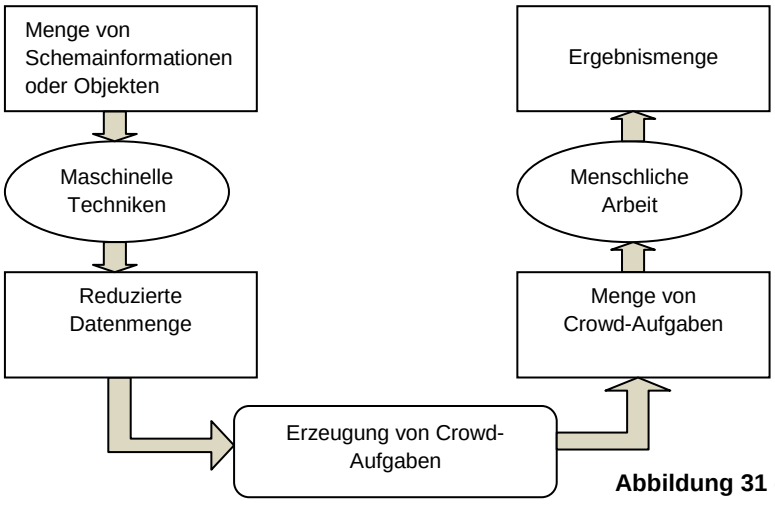

Abbildung 31 - Kombinierte Mensch-Maschine Arbeitsweise

4.4.2 Bereich Schema-Matching

Das Schema-Matching ist der zentrale und wohl anspruchsvollste Bereich der Datenintegration. Wie bereits angesprochen, ist zur Definition einer Abbildung zwischen zwei Schemata die Kenntnis der Semantik einzelner Schema-Elemente notwendig. Dafür wird in der Regel domänenspezifisches Expertenwissen benötigt. Dieser Umstand stellt für einen Crowdsourcing-Ansatz eine besonders schwierige Herausforderung dar. Es besteht zwar durchaus die Möglichkeit, dass sich in der Crowd einige Experten befinden, allerdings wird der Großteil im Allgemeinen von „interessierten Laien“ gestellt.

Somit bietet es sich nicht an Korrespondenzen zwischen einzelnen Schemata durch die Crowd finden zu lassen, geschweige denn den kompletten Matching-Prozess von den Crowd-Workern ausführen zu lassen. Vielmehr kann die Arbeitsleistung der Crowd dahingehend genutzt werden, berechnete Matching-Ergebnisse zu prüfen. Hung et al. greifen diese Idee auf und haben in [19] untersucht, wie sich Kontextinformationen aus dem Schema nutzen lassen, um die Worker bei dieser Validierungsaufgabe zu unterstützen. Dafür schlagen die Autoren die folgenden drei Arten von Kontextinformationen vor:

a) Alle alternativen Möglichkeiten
    - Alle möglichen Matching-Kandidaten, die maschinell berechnet wurden
b) Transitive Folgerung
    - Bereits bestehende Korrespondenzen werden aufgezeigt
c) Transitiver Verstoß
    - Sonderform einer transitiven Folgerung, wenn durch Beziehungen zwischen unterschiedlichen Schemata ein Kreis entsteht, so dass zwei unterschiedliche Attribute eines Schemas (transitiv) aufeinander abgebildet werden

Diese unterschiedlichen Zusatzinformationen sollen dem Worker helfen sich die Semantik der einzelnen Schemaelemente zu erschließen. Die Auswertung hat ergeben, dass die transitiven Folgerungen den Crowd-Workern geholfen haben korrekte Korrespondenzen als solche zu erkennen. Allerdings wurden auch falsche Korrespondenzen verstärkt als korrekt angesehen. Die Informationen bezüglich transitiver Verstöße hatten hingegen zur Folge, dass falsche Korrespondenzen als solche erkannt wurden. Abbildung 32 zeigt das genaue Ergebnis. ‘x‘ bezeichnen dabei falsche und ‘o‘ richtige Korrespondenzen. Die einzelnen Darstellungen entsprechen den unterschiedlichen Informationstypen (analog zu oben nummeriert).

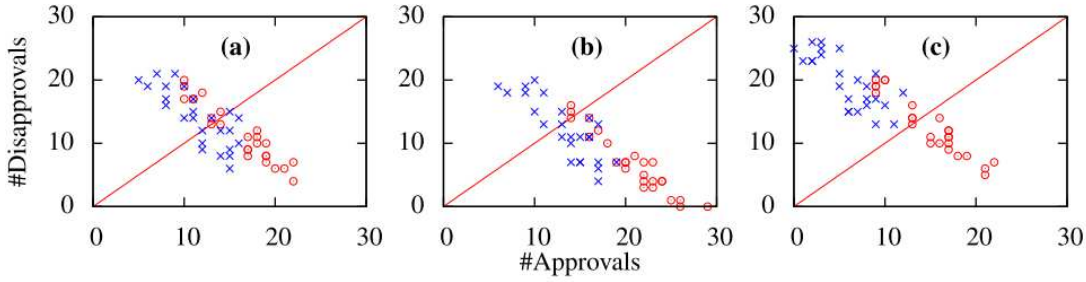

Abbildung 32 - Wirkung von Kontextinformationen

Weiterhin haben Hung et al. untersucht, welche Auswirkungen unterschiedliche Aggregationstechniken auf das Gesamtergebnis haben. Das Ziel dieser Untersuchung bestand darin, bei gleichbleibender Fragenanzahl ein möglichst genaues Gesamtergebnis bei der Zusammenfassung der Antworten einzelner Crowd-Worker zu erhalten.

Dafür werden zwei Einschränkungen berücksichtigt, die sich aus dem Schemakontext ergeben.

- 1-1 constraint
    - Nur eine von mehreren Korrespondenzen kann gültig sein.
- Circle constraint
    - Die Gültigkeit mehrerer Korrespondenzen impliziert die Gültigkeit einer weiteren Korrespondenz.

Eine Auswertung mit unterschiedlichen Fehlerschranken ε hat ergeben, dass unter Berücksichtigung dieser Informationen weniger Kosten aufgetreten sind, also weniger Fragen an die Crowd notwendig waren.

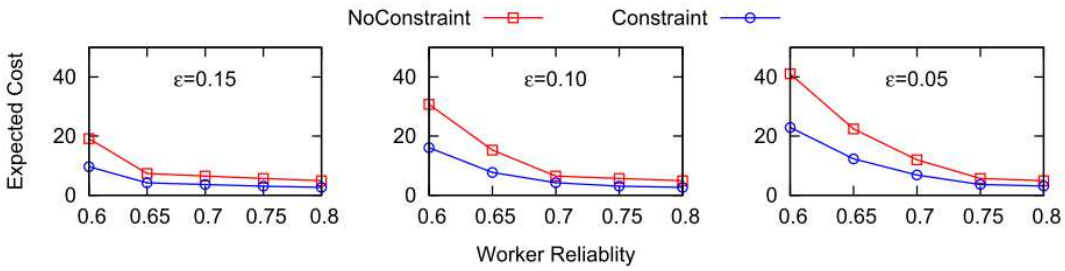

Abbildung 33 - Auswirkung von Aggregationsbeschränkungen

4.4.3 Bereich Object-Matching

Wie bereits angesprochen ist im Bereich der Duplikaterkennung der naive Ansatz alle möglichen Paarungen zu vergleichen nicht weiter zu verfolgen. Insbesondere wären bei Tabellen mit ein paar tausend Datensätzen unverhältnismäßig viele Vergleiche notwendig, was dazu führt, dass Aufwand und Nutzen in keinem ausgewogenen Verhältnis mehr stehen.

Bei [20] wird unter diesem Kontext ein kombinierter Ansatz gemäß 4.4.1 vorgestellt. Zunächst erfolgt im Zuge der maschinellen Vorverarbeitung eine grobe Auswahl möglicher Match-Kandidaten. Dabei wird als Ähnlichkeitsmaß die Jaccard-Ähnlichkeit (B.5) verwendet. Es hat sich gezeigt, dass schon eine geringe Ähnlichkeitsschranke ausreichend ist, um die Anzahl der Paare drastisch zu reduzieren, ohne dabei mögliche Duplikate zu übersehen.

(a) *Restaurant* Dataset

Threshold	Total #Pair	Matches	Recall
0.5	161	83	78.3%
0.4	755	99	93.4%
0.3	4,788	105	99.1%
0.2	23,944	106	100%
0.1	83,117	106	100%
0	367,653	106	100%

(b) *Product* Dataset

Threshold	Total #Pair	Matches	Recall
0.5	637	335	30.5%
0.4	1,427	571	52.1%
0.3	3,154	805	73.4%
0.2	8,315	1,011	92.2%
0.1	37,641	1,090	99.4%
0	1,180,452	1,097	100%

Abbildung 34 - Einfluss des Ähnlichkeitsmaßes auf die Duplikaterkennung

Die Spalte Recall gibt dabei an, wie viel Prozent der Duplikate erkannt wurden.

Um die Gesamtanzahl der Crowd-Aufgaben und damit die Kosten weiter zu reduzieren wurden mehrere Paare zu einer Aufgabe zusammengefasst. Dabei wurden zwei verschiedene Strategien verfolgt, aus denen unterschiedliche Aufgabentypen entstanden sind. Einmal die paarweisen Vergleiche, bei denen je zwei Datensätze miteinander verglichen werden. Hierbei ist somit nur die Entscheidung zu treffen, ob beide Datensätze das gleiche reale Objekt beschreiben oder nicht.

Eine komplette Aufgabe besteht dann aus *k* Paaren, die verglichen werden. Abbildung 35 zeigt ein Beispiel mit zwei Paaren.

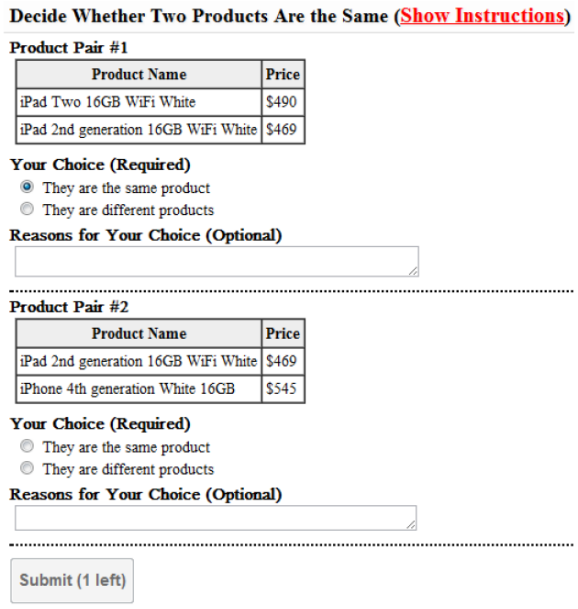

Abbildung 35 - paarweise Vergleichsaufgabe

Die zweite Strategie besteht darin, eine Teilmenge aus der Menge möglicher Matching-Paare auszuwählen. Der Crowd-Worker erhält dann die Aufgabe Duplikate durch Vergabe einer (identischen) Nummer zu markieren. Abbildung 36 zeigt die Umsetzung dieser Strategie.

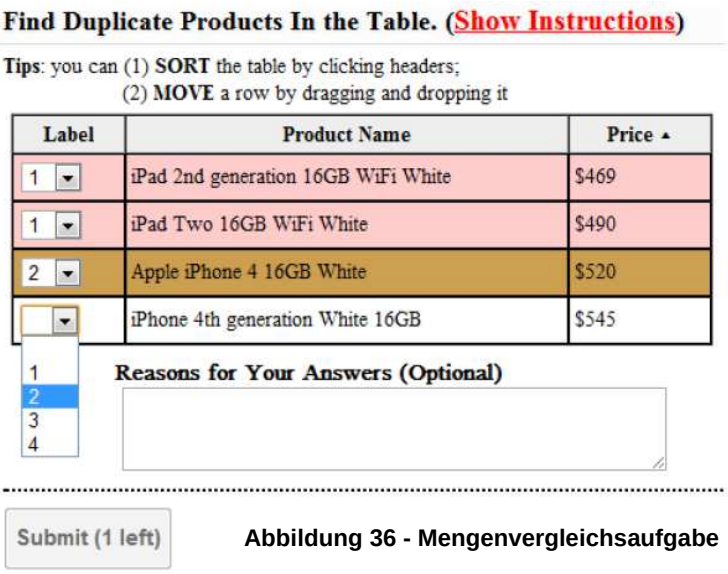

Abbildung 36 - Mengenvergleichsaufgabe

Zunächst wurde festgestellt, dass die Bearbeitung einer Mengenvergleichsaufgabe schneller abgeschlossen wird, als eine paarweise Vergleichsaufgabe. Vergleicht man hingegen die insgesamt benötigte Zeit, um eine komplette Datenmenge abzuarbeiten, so kommt es zu dem überaschenden Ergebnis, dass die Aufgaben mit den paarweisen Vergleichen schneller abgeschlossen wurden. Das liegt nach Ansicht der Autoren eventuell daran, dass dieser Aufgabentyp den Anschein erweckt leichter bearbeitbar zu sein und somit stärker nachgefragt wird. Liegen jedoch Datenbestände mit sehr vielen Duplikaten vor (Tabelle: *Product + Dup*), so wurde der andere Aufgabentyp eher beendet.

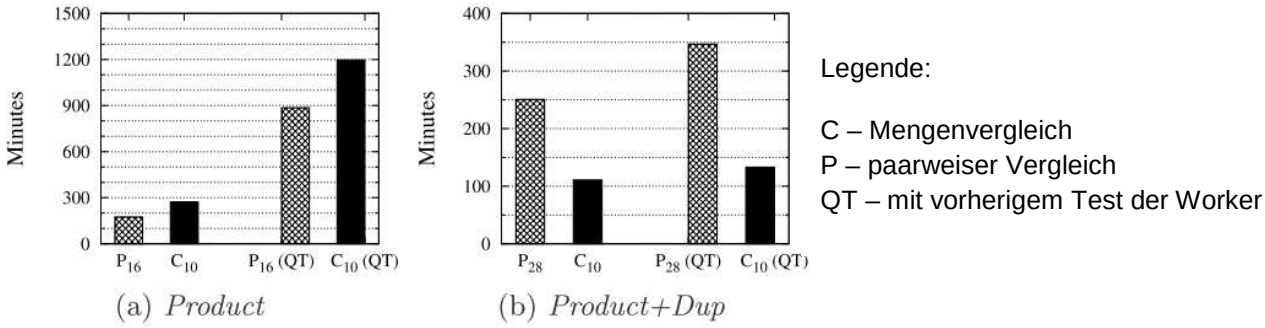

Abbildung 37 - Vergleich der gesamten Bearbeitungszeit

Das zentrale Ergebnis der Arbeit ist allerdings, dass durch die Verwendung von Crowdsourcing-Techniken erheblich bessere Ergebnisse, als mit „herkömmlichen“ maschinellen Techniken, erzielt wurden, wie Abbildung 38 verdeutlicht.

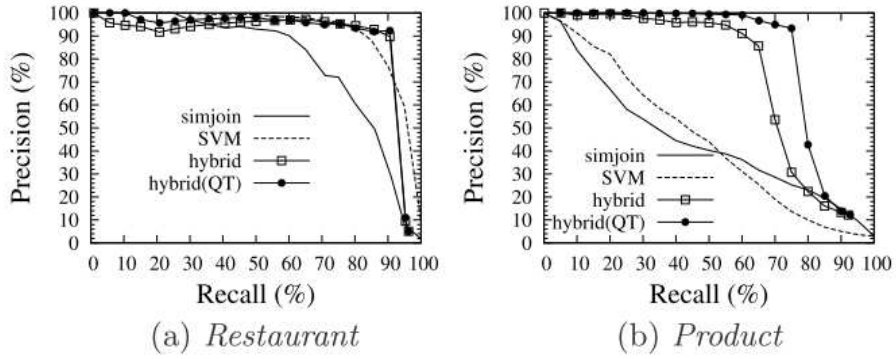

Abbildung 38 - Vergleich von kombinierten und rein maschinellen Ansätzen

Um die Kurven zu erhalten, wurde angenommen, dass das Ergebnis einer Object-Matching-Technik eine sortierte Liste von Paaren ist. Die Sortierung ergibt sich dabei aus der Wahrscheinlichkeit, dass zwei Objekte übereinstimmen. Von dieser Liste werden die ersten n Einträge als Match angesehen. Die Kurven ergeben sich nun, indem man n variiert und die Werte *Precision* bzw. *Recall* gegeneinander betrachtet. *Precision* beschreibt dabei, wie viel Prozent der gefundenen Duplikate tatsächlich Duplikate sind.

4.5 Offene Fragen

Im Zuge dieser Arbeit bleiben mehrere Fragen unbeantwortet. So konnte keine genaue Aussage getroffen werden, welche Kosten mit dem Einsatz von Crowdsourcing verbunden sind. Es kann gesagt werden, dass für Microjobs der vorgestellten Größenordnung bei AMT mit einer Worker-Vergütung von \$0,02 – \$0,20, zuzüglich einer Zahlung von \$0,005 für die Veröffentlichung, für jeden Hit zu kalkulieren ist. Inwieweit sich die kumulativen Gesamtkosten von den derzeitigen Kosten unterscheiden und in welchem Verhältnis diese zum gewonnenen Nutzen stehen bleibt ungeklärt. Ebenfalls liegen keine Zahlen vor, mit welchen Kosten die Entwicklung von Crowdsourcing-Anwendungen verbunden ist.

Weiterhin liegen keine Untersuchungen vor, welche die Frage beantworten, wie häufig gleiche Aufgaben gestellt werden müssen, um insgesamt ein entsprechend sicheres Ergebnis zu erhalten.

An diesen Punkten könnten weitere Untersuchungen ansetzen.

Bezüglich der vorgestellten Open-Source-Plattform PyBossa (3.6.4) bieten sich ebenfalls weitere Arbeiten an. Insbesondere erscheint eine praktische Arbeit, welche die Erstellung eines Projektes vorsieht, erstrebenswert.

5 Fazit

Die Datenintegration ist ein zentraler Bestandteil moderner Informationsverarbeitung. Durch die immer weiter steigende Informationsvielfalt und somit immer größer werden Datenvolumina ist eine überwiegend automatische Verarbeitung dieser Informationen unerlässlich. Eine rein maschinelle Verarbeitung der Daten wird allerdings dadurch erschwert, dass die einzelnen Datenquellen unterschiedlich strukturiert, modelliert sind, bzw. ihnen unterschiedliche Sprachen (SQL, XML, uvm.) zugrunde liegen. Um diese Heterogenität zu überwinden wird im Bereich des Schema-Matchings eine Abbildung gesucht, die es ermöglicht die Daten einer Quelle in die Struktur einer anderen Quelle zu überführen, oder die Daten mehrerer Quellen zu einer globalen logischen Struktur zusammenzufassen.
Darüber hinaus ist es aus Gründen der Datenintegrität notwendig, Datensätze aufzuspüren, die ein und dasselbe reale Objekt repräsentieren. Beides gestaltet sich für rein algorithmische Lösungen jedoch nahezu unmöglich. So werden immer wieder Matching-Paare übersehen oder fälschlicher Weise als Match angenommen. Insbesondere lässt sich die Semantik, die hinter einer speziellen Modellierung steckt, kaum maschinell erschließen.

Demnach ist man bei dem gesamten Prozess auf menschliche Fähigkeiten angewiesen. Aufgrund der drastischen Datenmenge ist diese Arbeit aber kaum von Einzelnen zu leisten, bzw. benötigt eine Bearbeitung entsprechend viel Zeit, trotz der Unterstützung zahlreicher Matching-Frameworks.

Somit ist der Versuch naheliegend, diese Arbeit auf mehrere Arbeiter zu verteilen. In der neueren Forschung und Entwicklung taucht dabei immer wieder der Begriff Crowdsourcing auf. Dabei wird auf die Arbeitsleistung und das Wissen einer sehr großen Masse von Menschen geschickt zurückgegriffen. Diese helfen zum Teil bewusst, zum Teil unbewusst dabei größere Aufgaben zu bearbeiten oder Probleme zu lösen, indem jeder Einzelne meist nur einen geringen Beitrag leistet.

Da für die Aufgaben der Datenintegration menschlichte Arbeit unabdingbar ist, bietet es sich also an eben dieses Crowdsourcing zu nutzen. Dabei gilt es jedoch mehrere Aspekte zu berücksichtigen. Zunächst ist nicht jeder Ansatz für diesen Bereich brauchbar. Neben einer Bezahlung der sog. Crowd-Worker besteht noch die Möglichkeit Minispiele zu konzipieren, bei denen im Verlauf des Spielens die benötigten Leistungen erbracht werden. Es hat sich gezeigt, dass sich Konzepte, die eine gewisse Konkurrenz zwischen den Crowd-Workern schüren, oder Konzepte bei denen nur gemeinsam ein Ziel erreicht werden kann, als erfolgreich erweisen. Allgemein ist bei einem bewusst stattfindenden Crowdsourcing immer zu überlegen, wie die Crowd motiviert werden soll. Forschungen und Umfragen in diese Richtung haben gezeigt, dass neben einer finanziellen Vergütung auch ein Feedback über die erbrachte Leistung, sowie eine Aufklärung über den Nutzen der getätigten Arbeit besonders motivierend sind.

Darüber hinaus ist es insbesondere im Bereich des Schema-Matchings wichtig die Crowd-Worker während ihrer Arbeit zu unterstützen. So können verschiedene Zusatzinformationen für einen Worker hilfreich sein, sich die Semantik einzelner Elemente zu erschließen, um eine korrekte Entscheidung zu treffen.

Ebenfalls gilt es zu überlegen, in welcher Art und Weise die Ergebnisse der Crowd-Worker weiterverarbeitet werden können. In diesem Bereich gibt es verschiedene Forschungen. Allgemein ist festzuhalten, dass durch eine Kombination von algorithmischen Lösungen und der Arbeit der Crowd bereits sehr gute Leistungen erzielt werden konnten.

Das Crowdsourcing ist jedoch nicht nur mit Vorteilen verbunden. So müssen bei der Planung einer Anwendung und bei der Auswertung der Ergebnisse böswillige User bzw. Spammer berücksichtigt werden.

Diese Arbeit zeigt insgesamt unterschiedliche Ansatzpunkte für Crowdsourcing-Techniken im Gebiet der Datenintegration. Sie gibt einen Überblickt darüber, welche Faktoren für eine derartige Anwendung zu berücksichtigen sind und betrachtet verschiedene Crowdsourcing-Konzepte, welche für die Probleme der Datenintegration vielversprechend erscheinen.

In wie weit sich die Kosten, die für die Entwicklung einer Crowdsourcing-Anwendung anfallen, von den bisherigen Kosten unterscheiden und in welchem Verhältnis diese Kosten zu einem eventuellen Mehrwert, der durch eine solche Anwendung erzielt werden kann, stehen, bleibt unbeantwortet. Diese und weitere offene Fragen liefern Ansätze für weitere Arbeiten.

6 Literaturverzeichnis:

[1] Vorlesungsskript Dr. M. Hartung, Universität Leipzig, Datenintegration SoSe 2013

[2] Vorlesungsskript Prof. Dr.-Ing. Ulf Leser, Humboldt-Universität Berlin, Informationsintegration SoSe 2012

[3] Vorlesungsskript Prof. T. Kudraß, HTWK Leipzig, Informationsintegration SoSe 2010

[4] Vorlesungsskript Prof. Dr.-Ing. Ulf Leser, Humboldt-Universität Berlin, Informationsintegration WS 2006/2007

[5] Erhard Rahm, Mehrrechner-Datenbanksysteme, Grundlagen der verteilten und parallelen Datenbankverarbeitung

[6] Vorlesungsskript Melanie Herschel, Universität Tübingen, Datenintegration & Datenherkunft, WS 2010/2011

[7] K. Hildebrand, M. Gebauer, H. Hinrichs, M. Mielke: Daten- und Informationsqualität – Auf dem Weg zur Information Excellence – 2. Auflage, Springer Verlag, 10.02.2011

[8] Prof. Dr. Felix Naumann: „Methoden der Dublettenerkennung“ in IQReport, 2007

[9] Leimeister, J. M. (2012): Crowdsourcing: Crowdfunding, Crowdvoting, Crowdcreation. In: Zeitschrift für Controlling und Management (ZFCM), Ausgabe/Nummer: 56, Erscheinungsjahr/Year: 2012. Seiten/Pages: 388-392

[10] G. Bombach, Belegarbeit: „Untersuchung von Methoden des expliziten Crowd-Sourcing für Location-Based Services in Map-Biquitous“, TU-Dresden, Jan. 2013

[11] Thomas Werner, Norbert Malanowski: „Crowdsourcing – Lösen von schwierigen Problemen durch Menschenmassen im Internet“ ITA-Kurzstudie, herausgegeben durch: Zukünftige Technologien Consulting der VDI Technologiezentrum GmbH, Düsseldorf 2011

[12] Mark Andrae, „Einsatz von Gamification zur Motivation in Crowdsourcing-Projekten“, schriftliche Masterarbeit, Ruhr-Universität Bochum, 2012 (abrufbar unter: stromfrei.de/static/Gamification.pdf)

[13] Armin Roth and Felix Naumann. *Qualitäts- und Semantik-gesteuerte Anfragebearbeitung für Peer-basierte Datenmanagementsysteme (PDMS)*. In INFORMATIK 2004 - Band 1, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, Germany, pages 341-345, 2004.

[14] Sergey Melnik, Hector Garcia-Molina, Erhard Rahm, Similarity Flooding: A Versatile Graph Matching Algorithm, 2002

[15] Hanna Köpcke, Andreas Thor, Erhard Rahm, Learning-Based Approaches for Matching Web Data Entities. In IEEE Computer Society, July/August 2010

[16] D. Aumueller, Hong-Hai Do, S. Massmann, E. Rahm: “Schema and Ontology Matching with COMA ++“, Sigmod 2005

[17] Kaufmann, N., Schulze, T., Veit, D. (2011). “More than fun and money. Worker Motivation in Crowdsourcing – A Study on Mechanical Turk“. AMCIS 2011 Proceedings

[18] Martin, Lessmann, Voß: “ Crowdsourcing: Systematisierung praktischer Ausprägungen und verwandter Konzepte“, Institut für Wirtschaftsinformatik, Universität Hamburg, 2007

[19] N. Q. V. Hung, N. T. Tam, Z. Miklós, K. Aberer: “On Leveraging Crowdsourcing Techniques for Schema Matching Networks“, in “The 18th International Conference on Database Systems for Advanced Applications (2013)“

[20] Jiannan Wang, Tim Kraska, Michael J. Franklin, Jianhua Feng: „CrowdER: Crowdsourcing Entity Resolution“, Proceedings of the VLDB Endowment, Vol. 5, No. 10, Aug. 2012

[21] Christian Papsdorf: „Wie Surfen zur Arbeit wird. Crowdsourcing im Web2.0“, Frankfurt a. M. / New Yourk: Campus, 2009

[22] Vorlesungsskript Prof. Dr. Felix Naumann, HU-Berlin, Workshop "Datenreinigung", SoSe 2006

[23] Peter Hofmann, Information Retrieval Seminar: Phonetische Suche, Johannes Gutenberg-Universität, WS 2009/10

[24] [GS62] – D.Gale, L. S. Shapley, “College Admissions and the Stability of Marriage“. In “The American Mathematical Monthly“, Vol. 69, No. 1, Jan. 1962, 9-15

[25] K. Hildebrand, M. Gebauer, H. Hinrichs, M. Mielke: Daten- und Informationsqualität – Auf dem Weg zur Information Excellence – 2. Auflage, Springer Verlag, 10.02.2011, Seite 124

Alle in der Arbeit angegebenen Internetquellen wurden zuletzt am 27.10.2013 abgerufen.

Anlage A - Kombination von Matchstrategien

Um die Vorteile der vorgestellten Match-Strategien zu nutzen und ihre Probleme teilweise zu kompensieren, bietet es sich an, die einzelnen Verfahren zu kombinieren. Wie bereits erwähnt, lassen sich zwei Kombinationsmöglichkeiten unterscheiden. Einerseits ist es möglich mehrere Verfahren auf eine Menge von Korrespondenzen

$$K = \{(a, b) | a \in A, b \in B\}$$

anzuwenden, um anschließend eine Auswahl der Paare (a, b) zu erhalten, die von mehreren, oder unterschiedlichen, Verfahren als Match vorgeschlagen werden. (Hybrid-Verfahren) Eine andere Möglichkeit ist es die einzelnen Verfahren nacheinander zu verwenden. Dabei setzt ein Verfahren auf dem Ergebnis des jeweils vorherigen Verfahrens auf. (Composite-Verfahren)

Dabei kommt es allerdings zu der Frage, in welcher Reihenfolge die Verfahren angewendet und welche Konfiguration der einzelnen Verfahren für eine kombinierte Nutzung geeignet ist. Unter der Konfiguration eines Verfahrens wird hierbei beispielsweise die Wahl von Schwellwerten bei der Match-Erkennung, oder die Wahl der zu vergleichenden Eigenschaften verstanden.

Ein Ansatz diese Fragen zu beantworten beruht auf dem sogenannten machine learning.

A.1 Maschinelles Lernen

Die Grundlage hierfür bildet zunächst eine homogene Menge H von bereits bewerteten Korrespondenzen wieder zwischen zwei Elementmengen A und B.

$$H = \{(a, b, s) | a \in A, b \in B, s \in [0,1]\}$$

Dabei bezeichnet s (similarity value) das Ähnlichkeitsmaß zwischen den Elementen $a \in A$ und $b \in B$. Ist $s > T$, so werden a und b als Match betrachtet. T ist dabei ein vordefinierter Schwellwert (engl. “threshold“). Auf dieser Grundlage ist es möglich, eine flexible Kombination von Matching- und Auswahl-Strategien vorzunehmen. H bezeichnen wir fortan als Trainigsdaten. Diese werden in der Regel manuell von Systemexperten erstellt. Um den manuellen Aufwand zu reduzieren, was das Ziel von (semi-) automatischen Matching-Algorithmen ist, ist die Frage danach, wie viel Training notwendig ist, von entscheidender Bedeutung.

Aus den Trainingsdaten wird zunächst eine Teilmenge ausgewählt. Mit Hilfe dieser Teilmenge wird dann ein Modell erzeugt, in welchem eine Wahl der Lernalgorithmen und Match-Strategien festgelegt ist. Dabei werden die Matching-Verfahren auf die ausgewählten Tupel (a, b) angewendet und die berechnete Übereinstimmung (s‘) mit dem korrekten (vorgegeben) Wert s vergleichen. Daraus lassen sich dann die besten Strategiekombinationen, sowie die günstigsten Konfigurationen ableiten. Dieses Modell soll später auf die „eigentlichen“ Quell- und Zieldaten angewendet werden. [15]

Bei der Auswahl der Trainingsdaten kommt es darauf an, sowohl Mapping-Paare ($s = 1$), als auch Non-Mapping-Paare ($s = 0$) zu verwenden, damit die Trainings-Phase nicht von trivialen (non-matching) Paaren dominiert wird. Für die Effizienz kommt es insbesondere darauf an, eine balancierte Auswahl zu treffen.

In der Studie von [15], auf der diese Ausführungen beruhen, werden des Weiteren vier Lernmethoden, für die Wahl der Match-Strategien, vorgestellt.

Namentlich sind das Entscheidungsbäume (decision trees), logistische Regression (logistic regression), SVM's (Support Vector Machines), sowie eine Kombination dieser drei Methoden. Letzteres ist allerdings mit einem sehr großen Aufwand (Kosten) verbunden, da zunächst die Match-Strategien durchgeführt werden müssen, welche von den drei anderen Lernmethoden vorgeschlagen wurden. Erst danach ist eine entsprechende Kombination der Ergebnisse möglich.

Bei einem Entscheidungsbaum werden die Match-Verfahren, die verwendet werden, sowie deren Ausführungsreihenfolge festgelegt. In jedem inneren Knoten wird dann getestet, ob ein gewisser Wert für ein spezielles Match-Verfahren überschritten ist. Dementsprechend wird dann der nächste Knoten gewählt. In den Blättern des Baumes ist dann die komplette Entscheidung enthalten.

Für die logistische Regression und die SVM's sei nur so viel gesagt, dass hier eine gewichtete Kombination der einzeln berechneten Ähnlichkeitswerte betrachtet wird.

Zusammenfassend lässt sich sagen, dass beim Ansatz des maschinellen Lernens günstige Konfigurationen und Kombinationen unterschiedlicher Strategien anhand von verifizierten Trainingsdaten gesucht werden. Diese Konfigurationen werden dann auf reale Daten angewendet, um ein Matching dieser Daten zu erhalten. Insgesamt wurden mit diesem Ansatz bereits positive Resultate erzielt.

Problematisch ist nach wie vor, dass entsprechend brauchbare Trainingsdaten vorliegen müssen. Diese zu erhalten ist weiterhin mit einem hohen manuellen Aufwand verbunden. Crowdsourcing könnte eine Möglichkeit sein, diesen Aufwand zu verringern, genauer gesagt zu verteilen.

Ein Ergebnis der Studie von [15] ist insbesondere, dass mit der Kombination unterschiedlicher Lernalgorithmen stets die besten Ergebnisse erzielt wurden.

A.2 Globales Matching

Ein weiteres Problem, bei der Kombination von Matching-Verfahren, entsteht durch 1:n (bzw. n:1) Beziehungen zwischen den Attributen zweier Schemata.

So können Attribute eines Schemas die gleiche Ähnlichkeit bezüglich einem Attribut eines anderen Schemas besitzen:

Abbildung 39 - Globales Matching - ähnliche Attribute

Oder mehrere Attribute können in einem anderen Schema zusammengefasst sein:

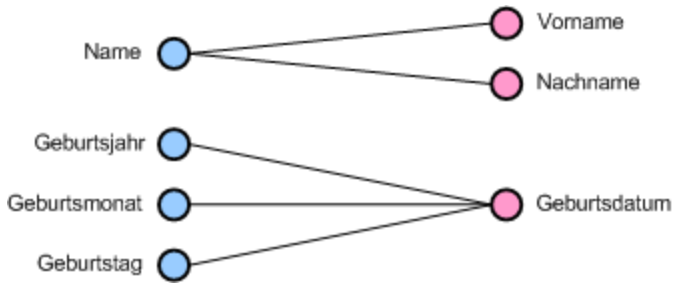

Abbildung 40 - Globales Matching - zusammengefasste Attribute

Um diesem Problem zu begegnen ist es notwendig nicht nur die einzelnen Attribute und deren Ähnlichkeit, sondern komplette Tabellen (oder Schemata) zu betrachten. Dies ist die Grundidee des globalen Matchings. [6]

Ein klassisches Problem hierbei besteht darin Attributpaare aus einer Menge von Attributen, die untereinander verschiedene Ähnlichkeiten aufweisen, so auszuwählen, dass stets nur 1:1 Abbildungen vorliegen und dabei ein möglichst gutes Gesamtmatching (maximale Gesamtzufriedenheit) vorliegt. Dies wird in der Literatur als Stable-Marriage-Problem bezeichnet. (vgl. B.2)

Anlage B - Algorithmen und Verfahren

Während der Datenintegration werden im Bereich des Schema-Matchings und des Object-Matchings immer wieder verschiedene Ähnlichkeitsmaße benötigt, um zu aussagen zu können, wie wahrscheinlich eine Übereinstimmung zweier Schema-Elemente bzw. zweier Datensätze ist. Ferner dienen Gruppierungs-Strategien dazu, nur relevante Informationen miteinander zu vergleichen.

Im Folgenden sollen einige dieser Algorithmen vorgestellt werden. Es ist dabei allerdings zu betonen, dass dies nur eine geringe Auswahl aus einer sehr großen Palette unterschiedlicher Verfahren darstellt.

B.1 Similarity Flooding

Das Similarity Flooding ist ein Algorithmus, der die Struktur eines Schemas berücksichtigt, um Korrespondenzen zwischen einzelnen Schema-Elementen aufzuspüren.

Zunächst sei hierbei eine initiale Ähnlichkeit zwischen zwei Schemaelementen gegeben. Diese initiale Ähnlichkeit erhält man durch einen anderen Ansatz (2.6.1 bzw. 2.6.2). Zunächst werden die Schemata durch Graphen dargestellt. Die Idee ist es nun eine Fixpunktiteration zur Berechnung der Ähnlichkeit. Dabei werden zwei Knoten unterschiedlicher Graphen als ähnlich betrachtet, wenn all ihre benachbarten Knoten ähnlich sind. Das heißt, dass Netzwerk wird so lange mit Ähnlichkeiten „geflutet“, bis ein Gleichgewicht (Fixpunkt) erreicht ist.

Genauer durchläuft der Algorithmus 4 Phasen:

1. Wie bereits angesprochen findet zunächst eine Umwandlung der Schemata S1, S2 in Graphen G1, G2 statt.
   G1 = SQL2Graph(S1);
   G2 = SQL2Graph(S2);

2. Als Nächstes wird ein “grobes” label-based matching durchgeführt, um die initiale Ähnlichkeit zu bestimmen.
   initialMap = StringMatch(G1, G2);

3. Nun folgt die erwähnte Fixpunktiteration, wobei die Ähnlichkeit zweier Elemente sich auf die Ähnlichkeit ihrer Nachbarn auswirkt. Die Anfangswerte dieser Iteration stehen dabei in *initialMap*.
   product = SFJoin(G1, G2, initialMap); (SFJoin = Similarity Flooding)

4. Im letzten Schritt werden die Daten gefiltert. Das bedeutet, es werden nur Ähnlichkeiten mit einem Wert größer als einem Schwellwert (engl. Threshold) akzeptiert.
   result = SelectThreshold(product);

Auf genaue Details des Algorithmus, wie die Definition der Ähnlichkeitsfunktion, oder verschiedene Filteransätze, wird hier nicht eingegangen.

Die Vorteile dieses Algorithmus liegen darin, dass er für jeden Schematyp anwendbar ist, er keine Trainingsphase benötigt und eine flexible Filterung liefert.

Nachteilig ist, dass er nur mit gerichteten Graphen arbeiten kann und auf gleiche Schemamodelle begrenzt ist. Das bedeutet, dass beispielsweise ein Matching eines XML-Schemas gegen ein anderes XML-Schema verwertbare Ergebnisse liefert, ein Matching eines Relationales Schemas gegen ein XML-Schema allerdings nicht. Darüber hinaus hat das initialMatch einen sehr großen Einfluss auf das Ergebnis, was wieder die Frage nach einem „guten Matching“ aufwirft.

B.2 Stable-Marriage-Problem / ~Algorithmus

Beim Stable-Marriage-Problem sind n Attribute in Schema A (Frauen) und m Attribute in Schema B (Männer) gegeben. Es werden nur 1:1 Matches zugelassen (je eine Frau heiratet je einen Mann). Weiterhin ist eine Rangliste für jedes der $n + m$ Attribute gegeben. Diese ergibt sich aus einem oder mehrerer Matching-Verfahren und ist in der Regel symmetrisch. Gesucht wird nun ein stabiles globales Matching (eine Paarung). Das heißt es soll niemals gelten:

- f_1 heiratet m_1, f_2 heiratet m_2,
- aber f_1 bevorzugt m_2 und m_2 bevorzugt f_1

Der Algorithmus von Gale und Shapley liefert für dieses Problem eine stabile Paarung. [24] Es werden dabei die Ranglisten der Attribute einer Menge durchgegangen und die entsprechend bevorzugten Attribute der anderen Menge als Match vorgemerkt. Tritt dabei der Fall auf, dass ein Attribut ein bereits vergebenes matchen soll, so wird vergleichen welches Attribut von dem „zu matchenden“ Attribut bevorzugt wird. Ist es das bereits vorgemerkte, so wird bei dem neuen Attribut in der Rangliste vorgerückt. Ist es das neue Attribut, so wird dieser Match vorgemerkt und bei dem ehemaligen Partner wird in der Rangliste vorgerückt.

Folgendes Beispiel (entnommen aus [6]) soll die Arbeitsweise des Algorithmus illustrieren:

Es seien folgende Ranglisten gegeben:

Männer (1-4)				
1:	B	D	A	C
2:	C	A	D	B
3:	B	C	A	D
4:	D	A	C	B

Frauen (A-D)				
A:	2	1	4	3
B:	4	3	1	2
C:	1	4	3	2
D:	2	1	4	3

Es werden die Attribute 1 – 4 durchgegangen. Dabei werden die Paarungen (1,B) und (2,C) als mögliche Matches vorgemerkt. Bei dem Versuch die Paarung (3,B) zu markieren kommt es zum Konflikt, da B bereits potentieller Partner von 1 ist.

Da aber in der Rangliste von B die 3 vor der 1 steht, wird die Paarung (3,B) als Match markiert und (1,B) verworfen. Daraufhin wird ein neuer Partner für 1 gesucht. Das ergibt nach Rangliste von 1 die Paarung (1,D).

Männer (1-4)				
1:	B	**D**	A	C
2:	**C**	A	D	B
3:	**B**	C	A	D
4:	D	A	C	B

Frauen (A-D)				
A:	2	1	4	3
B:	4	**3**	1	2
C:	1	4	3	**2**
D:	2	**1**	4	3

Bei dem Versuch 4 mit D zu matchen, tritt selbiger Konflikt auf. Allerdings steht der derzeitige Partner von D (die 1) vor der 4 (in Rangliste von D). Demzufolge wird der Match verworfen, was den letzten Match (4,A) liefert.

Männer (1-4)				
1:	B	**D**	A	C
2:	**C**	A	D	B
3:	**B**	C	A	D
4:	D	**A**	C	B

Frauen (A-D)				
A:	2	1	**4**	3
B:	4	**3**	1	2
C:	1	4	3	**2**
D:	2	**1**	4	3

Der Algorithmus liefert somit die 4 stabilen Paare (1,D), (2,C), (3,B) und (4,A). Allerdings wird hierbei keine maximale Gesamtzufriedenheit erreicht.

Um die Gesamtzufriedenheit zu beurteilen, betrachten wir die Zufriedenheit jedes Einzelnen. Dafür bilden wir jedes Paar (x, y) auf eine natürliche Zahl ab, welche sich aus dem Ranking des jeweiligen Partners ergibt.

$$(x, y) \rightarrow r_x(y) + r_y(x)$$

Dabei ist $r_x(y)$, resp. $r_y(x)$, das Ranking von y bei x, resp. von x bei y. Das Maß für die Gesamtzufriedenheit ergibt sich dann aus der Summe der Einzelzufriedenheiten. Offensichtlich gilt:

Je kleiner diese Summe, umso höher die Gesamtzufriedenheit.

Für die oben ermittelte Paarung ergeben sich somit folgende Werte:

(x, y)	$r_x(y)$	$r_y(x)$	$r_x(y) + r_y(x)$
(1,D)	2	2	4
(2,C)	1	4	5
(3,B)	1	2	3
(4,A)	2	3	5
Gesamtzufriedenheit			17

Die beste Gesamtzufriedenheit für dieses Beispiel liefert allerdings die folgende Paarung:

Männer (1-4)				
1:	B	**D**	A	C
2:	C	**A**	D	B
3:	**B**	C	A	D
4:	D	A	**C**	B

Frauen (A-D)				
A:	**2**	1	4	3
B:	4	**3**	1	2
C:	1	**4**	3	2
D:	2	**1**	4	3

(x, y)	$r_x(y)$	$r_y(x)$	$r_x(y) + r_y(x)$
(1,D)	2	2	4
(2,A)	2	1	3
(3,B)	1	2	3
(4,C)	3	2	5
Gesamtzufriedenheit			15

Weitere Mapping-Algorithmen, die derartige Paare von Elementen zweier Mengen bilden, sind ohne nähere Betrachtung:

- Maximum Weighted Bipartite Graph Matching Algorithm
  - Mapping unterschiedlich großer Schemata
  - Maximierung der Gesamtzufriedenheit
- Royal Couples Algorithm
- Royal Groups Algorithm
  - Mapping unterschiedlich großer Schemata und kann n:m-Mapping erzeugen

B.3 Editier-Distanz

Bei der Editier-Distanz handelt es sich um ein mögliches Ähnlichkeitsmaß, welches zwischen zwei Zeichenketten definiert werden kann.

Die Editier-Distanz beschreibt die Anzahl der Operationen (replace, insert, delete), die notwendig sind, um eine Zeichenkette S1 in eine andere Zeichenkette S2 zu überführen.

Dafür wird eine (|S1|+1) × (|S2|+1) – Matrix D gemäß folgendem Algorithmus belegt:

Algorithmus:
Seien $S1 = (a_1, \ldots, a_n)$ und $S2 = (b_1, \ldots, b_m)$.
Sei D eine $(n+1) \times (m+1)$-Matrix.
Die Belegung von D ist wie folgt gegeben:

$$
\begin{aligned}
D_{0,0} &= 0 \\
D_{i,0} &= i, \quad 1 \leq i \leq n \\
D_{0,j} &= j, \quad 1 \leq j \leq m \\
D_{i,j} &= \min \begin{cases} D_{i-1,j-1} & +0,\ falls\, a_i = b_j \\ D_{i-1,j-1} & +1\, Ersetzung \\ D_{i,j-1} & +1\, Einfügung \\ D_{i-1,j} & +1\, Löschung \end{cases}
\end{aligned}
$$

Nachdem alle Matrixeinträge berechnet wurden, steht die Editier-Distanz in $D_{n+1,m+1}$.

Abbildung 41 - Editier-Distanz Algorithmus

Zum Beispiel sei S1 = „AUTO“ und S2 = „RAD“. Dann liefert der Algorithmus folgende Matrixbelegung:

	j	0	1	2	3
i		ε	R	A	D
0	ε	0	1	2	3
1	A	1	1	1	2
2	U	2	2	2	2
3	T	3	3	3	3
4	O	4	4	4	4

D

Abbildung 42 - Editier-Distanz Beispiel 1

Also besitzt die Editier-Distanz von S1 nach S2 den Wert 4. Es müssen also bei einem Wort mit 4 Buchstaben 4 Änderungsoperationen vorgenommen werden. Das impliziert eine Ähnlichkeit von 0. Wählen wir hingegen S1 als MASSE und S2 als TRASSE, so beträgt die Editier Distanz 2. Das impliziert eine Ähnlichkeit von 4/6 = 0,66.

	j	0	1	2	3	4	5	6
i		ε	T	R	A	S	S	E
0	ε	0	1	2	3	4	5	6
1	M	1	1	2	3	4	5	6
2	A	2	2	2	2	3	4	5
3	S	3	3	3	3	2	3	4
4	S	4	4	4	4	3	2	3
5	E	5	5	5	5	4	3	2

D

Abbildung 43 - Editier-Distanz Beispiel 2

B.4 SOUNDEX-Maß

Ein weiteres Ähnlichkeitsmaß ist das sogenannte SOUNDEX-Maß. Hier wird bei der Ähnlichkeitsbewertung zusätzlich zur Editier-Distanz die Aussprache von Wörtern berücksichtigt. Dabei wird einem Wort ein Code zugewiesen, welcher aus dem ersten Buchstaben des Wortes,

gefolgt von 3 Ziffern, besteht. Die Ziffern ergeben sich aus den nachfolgenden Konsonanten im Wort. [8]

Welcher Konsonant dabei welchen Wert hat, lässt sich einer Tabelle, beispielsweise unter [23], entnehmen. Weitere detaillierte Erläuterungen finden sich ebenda.

B.5 Jaccard-Ähnlichkeit

Ein anderes Ähnlichkeitsmaß ist die sogenannte Jaccard-Ähnlichkeit. Dabei handelt es sich um ein tokenbasiertes Ähnlichkeitsmaß. Dabei wird die Schwäche der Editier-Distanz, sensibel auf Vertauschungen zu reagieren, berücksichtigt. Zeichenketten werden zunächst in einzelne Wörter bzw. Token zerlegt, um anschließend zu Vergleichen, wie viele Token die beiden Zeichenketten gemeinsam haben.

Die Jaccard-Ähnlichkeit ist dabei das Verhältnis zwischen gemeinsamen Token zu gesamter Tokenanzahl.
Sei beispielsweise S1 = "iPad Two 16GB WiFi White" und S2 = "iPad 2nd generation 16GB WiFi White". Dann beträgt die Jaccard-Ähnlichkeit zwischen S1 und S2

$$J(S1,S2) = \frac{|\{\text{iPad, 16GB, WiFi, White}\}|}{|\{\text{iPad, 16GB, WiFi, White, Two, 2nd, generation}\}|} = \frac{4}{7} \approx 0.57$$

Ein weiteres Verfahren ist die term-frequency / inverse-document-frequency. Hierbei werden die einzelnen Token zusätzlich gewichtet, abhängig davon wie häufig es in seiner Zeichenkette selbst und wie häufig es in allen betrachteten Zeichenketten auftaucht. [8]

B.6 Canopy Clustering

Das Canopy Clustering ist eine relativ einfache, aber sehr effiziente Methode, überlappende Teilmengen (Cluster) aus einer vorgegebenen Datenmenge zu erhalten.[23]

Hierfür werden eine Abstandsfunktion $d(x,y)$, eine Menge von Punkten L (Datenwerte) und zwei Schwellwerte $T_1 > T_2$ vorgegeben. Das Verfahren läuft dann in 5 Schritten ab[24]:

1. Wähle einen zufälligen Punkt p aus L
2. Berechne für alle anderen Punkte x aus L den Wert $d(x,p)$
3. Alle Punkte x mit $d(x,p) < T_1$ liegen in einem Cluster
4. Alle Punkte x mit $d(x,p) < T_2$ werden aus L entfernt
5. Wiederhole die Schritte 1 bis 4, solange bis L leer ist

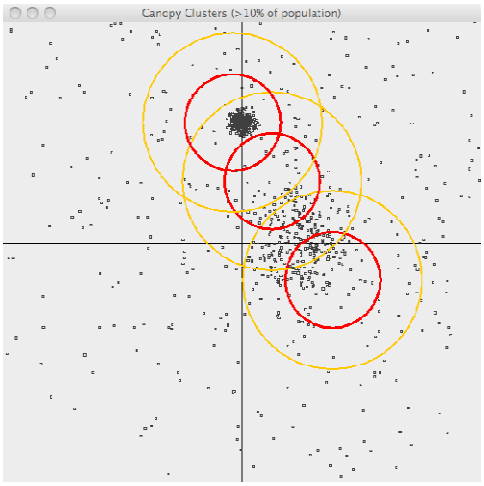

Abbildung 44 - Canopy Clustering

[23] https://cwiki.apache.org/confluence/display/MAHOUT/Canopy+Clustering
[24] de.slideshare.net/001ashishk/canopy-clustering-algorithm

Erklärung

"Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann."

Ort	Datum	Unterschrift
……..	……	……