Schema Matching

- Semantische Datenintegration
- Wolfgang Machert
- Universität Bremen 06.06.2005
- Betreuer: Prof. Dr. Joachim Hammer, Sebastian Hübner

Semantische Datenintegration

- Ziel: Integration von semantisch gleichartigen Daten aus syntaktisch verschiedenartigen Datenquellen

- bisherige Betrachtungen
  - Klassifikation von Konflikten
  - Anwendung von Mediatoren
  - Ontologieansätze
- bisher nicht betrachtet
  - zuordnung einzelner Daten unterschiedlicher Schemata
  - zuordnung einzelner Daten ohne explizit vorhandenes Schema

Inhalt

- Einführung
- Match Operator und Generisches Matchen
- Ansätze
  - Schema-level Matcher
  - Instanz-level Matcher
  - Kombinations Matcher
- Anwendungsbeispiele
  - LSD
  - ARTEMIS
  - CUPID

Einführung: Daten aus unterschiedlichen Schemata

- Unterschiedliche Schema meist mit unterschiedlicher Zielstellung entwickelt
- selbst bei gleichartigen Daten unterschiedliche Struktur möglich

Beispiel: Zusammenfassung eines News-channels

- Newschannel Beschreibung in RSS 2.0:

```
<rss version="2.0">
```

```
 <channel>
  <title>my news</title>
  <link>http://www.my.org/</link>
  <description>this is my news channel</description>
   <item>
    <title>most recent entry</title>
    <link>http://news.my.org/recent.html</link>
    <pubDate>Tue, 03 Jun 2003 09:39:21 GMT</pubDate>
   </item>
 </channel>
</rss>
```

- Newschannel Beschreibung in RSS 1.0:

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns="http://purl.org/rss/1.0/"
 xmlns:dc="http://purl.org/dc/elements/1.1/">
 <channel rdf:about="http://news.my.org/">
  <link>http://www.my.org/</link>
  <title>my news</title>
  <description>this is my news channel</description>
  <items>
   <rdf:Seq>
       <rdf:li resource="http://news.my.org/recent.html"/>
   </rdf:Seq>
  </items>
 </channel>
 <item rdf:about="http://news.my.org/recent.html">
  <title>most recent entry</title>
  <dc:date>2003-06-03T09:39:21+00:00</dc:date>
 </item>
</rdf:RDF>
```

Einführung: Daten aus unterschiedlichen Schemata

Probleme bei der Zusammenführung:

- Unterschiedliche Struktur der Schemata
- Unterschiedliche Namen für einzelne Elemente z.B. pubDate - dc:date (Bezeichnerkonflikt)
- Unterschiedliche Formate der Werte z.B. RFC2822 - ISO 8601 (Datentypkonflikt)
- Besonderheit bei RSS 1.0: Elemente aus externen Schemata z.B. dc:date
- Besonderheit bei RDF: keine feste Beschränkung der Schemata

Einführung: Daten aus undefinierten Quellen

- Mitteilungen oder Mails besitzen inhärentes Schema
- Problem: Schema meist nicht (vorher) definiert, jedoch verbreitet (Nachahmungseffekt)

Beispiel: Zusage zu einem Treffen

- Mail vom Web-Server des System verschickt:

```
From: My.org WebMail <admin@my.org>
To: <office@my.org>
Subject:

subject(Meeting)
subject-date(2003:06:04)
subject-topic(Newsletter)
senderfirstname(Max)
senderlastname(Schmitt)
sendermail(max@schmitt.de)
note(Komme etwas später!)
```

- Mail vom E-Mailclient des Nutzers verschickt:

```
From: Max Schmitt <max@schmitt.de>
To: Büro <office@my.org>
Subject: [Treffen] 04.06.2003 bzgl. Newsletter

Hallo,
Ich werde etwas später kommen!
```

Match Operator und Generisches Matchen

- Schema: Menge von Elementen welche durch Struktur verbunden sind
- verbunden mit Repräsentation
- Mapping: Zuordnung von Element(en) aus einem Schema zu Element(en) aus anderem Schema
- verschiedene Mappingformen
  - gerichtet - abbildung zwischen elementen
    - catalog#xpointer(catalog/books) = books#xpointer(books/hardcover | books/softcover)
    - rss1#xpointer(&rdf;RDF/item) partOf rss2#xpointer(rss/channel/item)
    - S2.From = concat(S1.senderfirstname, S1.senderlastname, S1.sendermail)
    - S3.isOnAgenda = S1.subject-date > current-date()
  - ungerichtet - relation zwischen Elementen (S1.Name, S2.Name)

- Match: Operation über zwei Schemata welches Mapping zurückliefert
- Vergleichbar mit join:
  - join Operation über Daten
  - match Operation über Metadaten
- (partial) OuterMatch: jedes Element eines bestimmten Schema in mapping enthalten
- full OuterMatch: jedes Element beider Schemata in mapping enthalten

Match Operator und Generisches Matchen

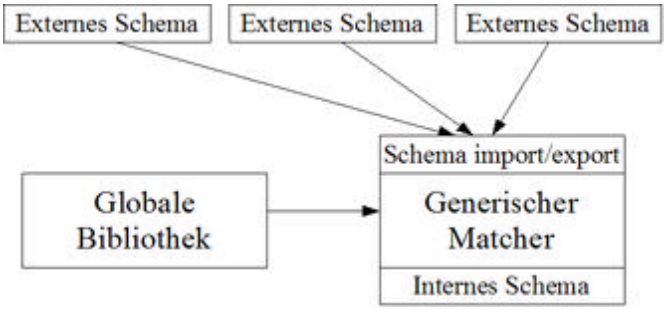

Abb: Aufbau eines Generischen Matchers (Quelle: RB01, S.337)

- import aus externen Schemata in internes Schema
- matcher arbeitet nur auf internem Schema
- verwendung von externen Bibliotheken (Wörterbücher, Thesauri, ...)
- export von internem Schema in externe Schemata

- aber: nicht alle Abbildungen in Praxis auffindbar
- indentifikation von Kandidaten durch Matcher
- Möglichkeit des Nutzers bezüglich Kandidaten
  - akzeptieren
  - verwerfen
  - verändern
  - eigene hinzufügen

Ansätze

Ansätze für Matcher-Entwicklung

- Instanz vs. Schema matching: Operation auf Daten oder Metadaten?
- Element vs. Struktur matching: Operation auf Einzelelementen oder Schema-Stuktur?
- Sprachverarbeitung vs. Constraintverwendung
- Kardinalität des Matchers
- Verwendung externer Informationen

Ansätze: Schema-level Matcher - Granularität

- Element-level: betrachtung einzelner Atomarer Daten (z.B. Attribute, Spalten)
- auch isolierte Betrachtung gröberer Elemente (z.B. Knoten eines Baumes)
- implementierung ähnlich zu join
- beispiele siehe Match Operator

- Struktur-level: betrachtung vollständiger oder partieller Strukturen
- partial structured Match vs. full structured Match
- partial structured Match am Beispiel RSS 2.0 vs. RSS 1.0

```
- rss                -rdf:RDF
+- channel           +- channel
                      @rdf:about
```

```
+- link            +- link
+- title            +- title
+- description     +- description
                   +- items
+- item           +- item
 +- link           @about
 +- title          +- title
 +- pubDate        +- dc:date
```

Ansätze: Schema-level Matcher - Kardinalität

- Matching einzelner Elemente mit mehreren anderen Elementen möglich (ähnlich Relationskardinalitäten)
  - 1:1
  - 1:n
  - n:1
- addressierung mehrerer Elemente mittels Ausdrücken (siehe Match Operator)
- structure-Matching: n:m Kardinalität möglich
- lokale vs. globale Kardinalitäten
  - lokal: mehrere Mappings auf ein Element möglich
    - books#xpointer(books/hardcover) partOf catalog#xpointer(catalog/books)
    - books#xpointer(books/softcover) partOf catalog#xpointer(catalog/books)
  - global: nur ein Mapping auf ein Element
    - catalog#xpointer(catalog/books) = books#xpointer(books/hardcover | books/softcover))
- Kardinalitäten auch für Instanz-level Matching möglich

Ansätze: Schema-level Matcher - Linguistik-basiert

- matching gleichartiger Namen
- Zwei Ansätze: Namens Matching, Beschreibungs Matching

- Namensgleichheit
  - Identität von Namen - z.B. gleiche Namen im Namensraum (rdf:RDF <=> rdf:RDF)
  - Gleichheit der Kanonischen Namen - expansion von Prefixen/Sufixen (pubDate == publicationDate)
  - Gleichheit von Synonymen (publication == created)
  - Gleichheit von Hypernymen - verwendung von Taxonomien oder Ontologien (pubDate isA Date)
  - Ähnlichkeit von Namen basierend auf Teilwörtern, Soundex,...
  - Benutzergegebene Gleichheiten (rss/item/link == rdf:RDF/item/@about)

  - verwendung externer Wörterbücher (Ontologie, Thesaurus,...)
  - Homonyme irreführend (catlog/books != books)
  - anwendung auf unterschiedlichen Granularitätsebenen
  - anwendung auf Matches höherer Kardinalität

- Beschreibungsgleichheit
  - Schlüsselwörtern oder Synonymen aus Beschreibungen
    - rss/channel/title: The name of the channel...
    - rdf:RDF/channel/title: A descriptive title for the channel.
  - Verwendung von Natural Language Processing-Techniken zur Gleichheitsbestimmung

Ansätze: Schema-level Matcher - Constraint-basiert

- Constraints beider Schemata für Matcher verwendbar
  - Datentypen
  - Schlüsselcharacteristika (required, unique key)
  - Kardinalitäten
  - Relationen (isA, partOf)
- Synonymtabellen hilfreich
  - xsd:int == int
  - float/double == numeric
- aus Strukturen ableitbar (partOf, isA)
- Constraint Information allein führen zu Clustern (n:m mappings)

```
- rss                        -rdf:RDF
+- channel                   +- channel
                              @rdf:about(xsd:anyURI)
 +- link(xsd:anyURI)          +- link(xsd:anyURI)
 +- title(xsd:string)         +- title(xsd:string)
 +- description(xsd:string)   +- description(xsd:string)
                              +- items
 +- item                     +- item
  +- pubDate(xsd:dateTime)    +- dc:date(xsd:dateTime)
```

Ansätze: Schema-level Matcher - Wiederverwendung von Mappings

- viele Schemata ähnlich
- nutzung bereits vorhandener Schema-Mappings bzw. Schema-Fragmente

```
- rss              -rdf:RDF              -html
+- channel         +- channel
                    @rdf:about
 +- link            +- link
 +- title            +- title            +-head/title +-body/h1[1]
 +- description     +- description       +-body/p[1]
                    +- items
 +- item           +- item               +-body/ul[1]/li
  +- link           @about
  +- title          +- title             +-body/ul[1]/li/ul/li[1]
  +- pubDate        +- dc:date           +-body/ul[1]/li/ul/li[2]
```

- Speicherung in Schemabibliothek oder Thesaurus
- Wiederverwendung allgemeiner (standardisierter) Schemata (bsp. W3C, OASIS, IEEE, RFC) -

Möglichkeit des vorhandensein eines Matchers
- Problematisch: vorhandenes Schema mit verschiedener Domäne (bzw. Zweckentfremdung von Schemata) - bsp. HTML

Ansätze: Instanz-level Matcher

- notwendig wenn kein Schema vorhanden
- erzeugen eines Schema aus Instanzdaten (bsp. erzeugen einer DTD aus XML datei)
- verwendung von Instanzen um Fehler in Schema zu erkennen
- Ansätze aus Schema-level Matching anwendbar
  - Text: linguistische Ansätze
  - Numerische Werte: Constraint Ansätze
- identität mittels häufig auftretender gleicher Instanzen
  Beispiel: Identifikation von S1.Product = concat(S2.Vendor, S2.Product)

```
S1.Product                | S2.Vendor     | S2.Product
-------------------------------------------------------
SuSE Linux 9              | Macromedia    | DreamWeaver
Adobe Photoshop           | SuSE          | Linux 9
SuSE Linux 8              | RedHat        | Fedora Core
SuSE Linux 9              | SuSE          | Linux 9
OmniGroup OmniGraffle 3   | Adobe         | InDesign
```

- für Struktur-level Matching nicht geeignet (zu viele Möglichkeiten)

Ansätze: Kombinations Matcher

- kombination von unterschiedlichen Matchern meist informativer
- zwei Möglichkeiten
  - Hybride Matcher
  - Composite Matcher

- Hybride Matcher
  - (feste) vereinigung mehrerer Matcher
  - Performanter da vermeidung iterativer Durchläufe
- Composite Matcher
  - verbindung der Resultate mehrerer Matcher, sequentielle Ausführung
  - generischer (jedoch langsamer) Ansatz
  - sortierung und veränderung der Reihenfolge möglich

Ansätze: LSD

- Machine Learning basierend
- Phase 1: Lernphase
  - Verwendung eines bereits gematchten Schemas
  - Verwendung mehrerer Lernalgorithmen (Instanzen, Stuktur...)

- Phase 2: Klassifikation
  - Extraktion von Informationen aus Quell-Schema
  - unterschiedliche Lernalgorithmen bekommen ihre Informationen
  - Rückgabe eines Vorraussagewertes (welches Element wurde mit welcher Sicherheit bestimmt {(E1, s1), (E2, s2),...})
  - Auswertung aller Vorraussagen in Meta-Lernalgorithmus
  - gleiches Verfahren mit allen Elementen
  - Mapping wird in Vorraussage-Kombinierer bestimmt
- Verwendete Lernverfahren:
  - Whirl: Klassifikation nach TF/IDF und Nearest Neighbour
  - Naive Bayesian: arbeitet auf Wortfrequenzen
  - Name: Klassifikation von Schema Elementnamen nach TF/IDF
  - Ländername: Klassifikation nach Datenbank mit Ländernamen
- Meta-Lernverfahren: berechnung von gewichteten Summen für einzelne Lernverfahren in Lernphase
- Kombinationslerner: berechnung anhand Label (Elementnamen):
  - p, q vorbestimmte Schwellwerte
  - L1 > p%
  - L1 - L2 >= q

Ansätze: Artemis

- berechnung von Neigungen (0..1) zwischen Attributen
  - Namen: Kombination generischer u. Domainspezifischer Thesaurus
  - Datentypen: generische Liste mit Datentypkompatibilitäten
  - Strukturelle Affinität: Ähnlichkeiten zwischen Entitäten
- Clustering von Attributen basierend auf Neigungen
- Generierung von Sichten aus Cluster

Ansätze: Cupid

- Hybrider Matcher, Schema basieren, Element und Stuktur Matching
- verwendung von Schlüsseln, Referenzen und Sichten
- Ähnlichkeitskoeffizient für Schema Bäme (0..1)
  - Phase 1: Linguistik Matching, ergibt Koeff. für jedes Elementpaar
    - erkennen von Matches basieren auf Namen, Datentypen,...
    - verwendung von Thesaurus (Abkürzungen, Acronyme, Synonyme)
  - Phase 2: Struktur Matching, ergibt Stuktureller Ähnlichkeitskoeffizient für jedes Elementpaar
    - erkennen von Matches basierend auf Kontext (match der Eltern-, Geschwisterknoten?)
    - verwendung der Matches aus Phase 1
- Phase 3: berechnung der gewichteten Ähnlichkeit
  - wstruct - constante (0..1)
  - wsim = wstruct * ssim + (1- wstruct) * lsim
  - Neigung zu Linguistik oder Struktur Matching (abhängig von wstruct)
- Auswahl des Mappings basierend auf maximalem gewichteten Ähnlichkeitswert

Quellen

- [CD99] - Castano, S.; De Antonellis, V. (1999): A schema analysis and reconciliation tool environment. In: Proc Int Database Eng Appl Symp (IDEAS), pp. 53–62
- [DDL00] - Doan, A.; Domingos, P; Halevy, A. (2001): Learning source descriptions for data integration. In: Proceedings WebDB Workshop, 81-92
- [MBR01] - Madhavan, J.; Bernstein P. (2001): Generic schema matching with CUPID. In: Proceedings 27th International Conference On VLDB, 49-58
- [RB01] - Rahm, E.; Bernstein, P. (2001): A survey of approaches to automatic schema matching. In: The VLDB Journal 10:334-350