Linkage Flooding: Ein Algorithmus zur dateninhaltsorientierten Fusion in vernetzten Informationsbeständen

Vanda Lehel, Florian Matthes, Sebastian Riedel

Lehrstuhl Software Engineering betrieblicher Informationssysteme
TU München, Institut für Informatik
Boltzmannstrasse 3
85748 Garching
lehel@in.tum.de
matthes@in.tum.de

Abstract: Dieses Papier stellt ein spezielles Record Linkage Verfahren (Linkage Flooding) vor, das für die Suche nach Duplikaten in vernetzten Informationsbeständen optimiert ist. Nach einer kurzen Erläuterung von Anwendungsszenarien des Record Linkage sowie der Vorstellung des Record Linkage Prozesses wird der Linkage Flooding Algorithmus beschrieben und über experimentelle Ergebnisse bei der Duplikaterkennung bei MP3-Sammlungen berichtet.

1 Problemstellung

Immer dann, wenn Informationen aus verschiedenen Systemen zu integrieren sind oder von unterschiedlichen Benutzern erfasst und gepflegt werden, können Duplikate entstehen. Dies führt zu einem Bedarf nach dateninhaltsorientierter Fusion. Beispielsweise können in zwei Systemen Musikbibliotheken verwaltet werden, die Informationen über Lieder, Interpreten, Alben und Genres enthalten. Diese Informationen sind möglicherweise syntaktisch verschieden, Liednamen können zum Beispiel falsch geschrieben oder Genres verschieden benannt sein. Vereinigt man beide Datenbestände, entstehen doppelte Datensätze, die dasselbe Lied oder Genre beschreiben. Ein ähnliches Szenario ist die Synchronisation von Publikationsinformationen eines Lehrstuhls gegenüber persönlich gesammelten Literaturreferenzen [Le02], wo zunächst ebenfalls nach Duplikaten gesucht wird. Mit der Erfassung dieser Duplikate beschäftigt sich *Record Linkage,* die Suche nach Datensätzen, die sich auf dieselbe semantische Entität beziehen, vgl. [FS69].

Record Linkage ist dann trivial, wenn Datensätze einen Schlüssel besitzen, der die referenzierten Entitäten eindeutig kennzeichnet. Ist dies nicht der Fall und sind insbesondere syntaktische Differenzen vorhanden, so erweist sich diese Aufgabe jedoch als deutlich komplexer. Im Falle von vernetzten Informationen in einer Datenbank hängt die Bedeutung der betreffenden Datensätze nicht nur von ihren Attributen, sondern auch von Referenzen auf andere Datensätze ab. Zwei Genres in einer Musik-Taxonomie sind

beispielsweise wahrscheinlicher identisch, wenn sich ihre Untergenres gleichen, deren Identität wiederum von der Gleichheit ihrer Obergenres abhängt.

Nach einer Erläuterung der Schritte des Record Linkage Prozesses mit den jeweils angewandten Methoden (Abschnitt 2) wird in Abschnitt 3 ein Algorithmus vorgestellt, der speziell für die Suche nach Duplikaten in vernetzten Datenbeständen optimiert ist. Das Papier endet mit einer Bewertung des Verfahrens anhand von experimentellen Ergebnissen bei dem Einsatz in einem Informationsportal sowie einem Ausblick auf zukünftige Forschungsarbeiten.

2 Der Record Linkage Prozess

Der Record Linkage Prozess, der diesem Papier zugrunde liegt, besteht aus sechs wesentlichen Schritten, die nachfolgend beschrieben werden, wie sie allgemein auf zwei Datenbestände A und B angewendet werden; Record Linkage innerhalb eines Datenbestandes ist ein Spezialfall mit $A = B$.

Während der **Vorverarbeitung** werden die Datensätze in den Datenbeständen strukturiert und normalisiert, damit sie leichter zu vergleichen sind. Dazu können Verfahren wie in [CC02] das Hidden Markov Model oder für Zeichenketten Stemming-Techniken verwendet werden. Im zweiten Schritt, dem **Schema Matching**, werden die Elemente der Metamodelle von A und B gebunden (vgl. [MGR02]). Der folgende Schritt ist die **Suche** nach potentiellen Kandidaten für Bindungspaare, in der der Suchraum $S \subset A \times B$, also $S \in \mathcal{P}(A \times B)$, in den beiden Datenbeständen durch Anwendung geeigneter Suchheuristiken eingeschränkt wird. Sehr verbreitet ist hierfür die *Sorted Neighbourhood Method (SNM)*, erstmals vorgestellt in [HS95]. Anschließend folgt in Schritt 4 der **Vergleich** der Datensatzpaare aus dem Suchraum S mittels einer Vergleichsfunktion. Diese basiert normalerweise auf Distanzen zwischen den Attributen des Datensatzpaares, wie z.B. der String-Edit- oder Jaro-Distanz. Um die konsistenten Bindungswerte für die Datensätze des gesamten Datenbestandes unter Berücksichtigung ihrer Vernetzung zu berechnen, werden *globale Vergleichsverfahren* als Methoden eingesetzt. Eine wichtige globale Methode stellt das Similarity Flooding Verfahren [MGR02] dar, auf dem das in Abschnitt 3 des Papiers beschriebene Linkage Flooding Algorithmus basiert. In Schritt 5 wird das **Filtern** durchgeführt: Ein *Filter* ist formal eine Funktion, die eine Bindungsmatrix auf eine andere abbildet (z. B. Berechnung der transitiven Hülle). Das **Entscheiden** über das Binden von zwei Datensätzen als Abschluss des Record Linkage Prozesses steht in einem Informationssystem für das Erstellen einer expliziten Verbindung, einem *Link*, zwischen diesen Datensätzen. Ob ein Link erstellt werden soll, ist abhängig von dem Vergleichswert eines Datensatzpaares. Sucht man doppelte Datensätze, werden nur Paare (a,b), für die dieser Wert einen Schwellwert überschreitet, gebunden. Bei Paaren, deren Vergleichswerte unter diesen Schwellwert fallen, die aber mit einer gewissen Wahrscheinlichkeit Links sein könnten, so genannte *Candidates*, lässt man den Benutzer entscheiden, ob es sich um einen Link oder keinen Link, einen *Nolink*, handelt.

3 Der Linkage Flooding Algorithmus

In diesem Abschnitt wird der Linkage Flooding Algorithmus als ein globales Vergleichsverfahren vorgestellt, das die Vernetzung von Informationen in den Datenbeständen in einer natürlichen Weise berücksichtigt. Linkage Flooding übernimmt den Ansatz von Similarity Flooding, da sich dieses Verfahren für die Ähnlichkeitssuche auf Taxonomien als sehr brauchbar erwiesen hat und zudem durch seine Einfachheit überzeugt. Es wurden dabei die folgenden zusätzlichen Erweiterungen vorgenommen:

- Verallgemeinerung der Iterationsgleichung, so dass anstatt einer gewichteten Summe prinzipiell jede Vergleichsfunktion für den Vergleich von Datensätzen verwendet werden kann.
- Optimierung des Verfahrens, damit keine Vergleichswerte unnötig berechnet werden, und das Verfahren auch für große Datenmengen skaliert.

Die Iterationsgleichung ergibt sich dabei wie folgt:

$$M'_i(a,b) = f(a, b, M_{i-1}) \tag{3.1}$$

$$M_i(a,b) = norm\,(M'_i(a,b)) \tag{3.2}$$

M_i ist eine Bindungsmatrix zur Iteration i und enthält die Bindungswerte zwischen allen Paaren aus $A \times B$. f benutzt bestimmte Bindungswerte aus M_{i-1} sowie die Attribute und Referenzen von a und b, um $M_i(a,b)$ zu berechnen. Welche Bindungswerte aus M_{i-1} in die Berechnung mit eingehen, hängt von der Vergleichsfunktion f und den Referenzen der Datensätze a und b ab, die somit einen Abhängigkeitsgraphen auf $A \times B$ induzieren. Die Funktion *norm* ist eine Normalisierungsfunktion, die alle Bindungswerte in den Bereich [0,1] abbildet. Die bereits angesprochene Optimierung nutzt Suchheuristiken (vgl. Abschnitt 2), damit nicht in jedem Iterationsschritt die Gleichungen $|A| \times |B|$ mal ausgewertet werden müssen. Um den Algorithmus in dieser Hinsicht zu optimieren, werden *Quellen* und *Senken* eingeführt. Eine Quelle ist ein Datensatzpaar, deren Bindungsgrad einen bestimmten Schwellwert überschreitet (Bindungspaar) und mittels der Funktion f andere Paare beeinflusst. Eine Senke ist ein Datensatzpaar, das sich entweder in der durch eine Suchheuristik bestimmten Kandidatenmenge befindet oder in dessen Umgebung sich Quellen befinden, die sie beeinflussen, also referenzierte Datensätze im vernetzen Informationsbestand.

Der Linkage Flooding Algorithmus wird wie folgt ausgeführt: Zunächst wird die Bindungsmatrix M_0 **initialisiert**, indem im Falle $A \neq B$ die Bindungswerte auf den Wert Null gesetzt werden. Die Menge Q der Quellen sowie die Menge S der Senken ist anfangs leer. Nun werden durch eine **Suchheuristik** aus allen Paaren Bindungskandidaten ausgewählt: Die Menge der Senken wird gefüllt. Diese ist im weiteren Verlauf stets die Menge an Datensatzpaaren, für die in der nächsten Iteration der Bindungsgrad berechnet werden soll. Der nächste Schritt ist das **Fluten**; in jeder Flutiteration i wird zunächst für alle Senken die Funktion f evaluiert (siehe Gl. 3.1) und in der Bindungsmatrix M_i gespeichert. Abbildung 3.1 (a.) zeigt einen Ausschnitt aus einem Abhängigkeitsgraphen vor dem Fluten von Bindungswerten. Zum gegebenen Zeitpunkt sind alle Bindungswerte gleich Null. Die gezeigten Senken sind durch eine Suchheuristik bestimmt worden. Die Bindungswerte der Paare, die durch gestrichelte

Linien mit Senken verbunden sind, werden neben den Attributen der Datensätze zum Berechnen des nächsten Bindungswerts der jeweiligen Senke verwendet. Durchgezogene Linien stellen Beeinflussungen dar, die jedoch nicht zur Auswertung von *f* benutzt werden, da sie nicht auf Senken zeigen. Der **Abbruch des Verfahrens** geschieht nach derselben Methode wie im Similarity Flooding Verfahren: Es wird die maximale Änderung gesucht, die den Senken während des Flutens widerfahren ist. Liegt diese unter einem Schwellwert, kann abgebrochen werden. Andernfalls wird in M_i die **Menge der Quellen aktualisiert**: Jede Senke, deren Bindungsgrad einen Schwellwert übersteigt, wird zu einer Quelle, da sie höchstwahrscheinlich ihre Umgebung beeinflusst. In Abbildung 3.1 (b.) sind neue Quellen durch schwarz gefüllte Kreise dargestellt. Man erkennt, dass einige der Senken aus Abbildung 3.1 (a.) zu Quellen geworden sind. Senken, für die das nicht gilt, haben einen zu geringen Bindungswert. Mit den gefundenen Quellen wird anschließend die **Menge der Senken aktualisiert**. Dabei wird für jede Quelle betrachtet, welche anderen Paare sie mittels *f* beeinflusst. Hier kommen die durchgezogenen Linien aus der Abbildung 3.1 (a.) zum Tragen. In Abbildung 3.1 (b.) erkennt man auch die neuen Senken an den Enden dieser durchgezogenen Linien.

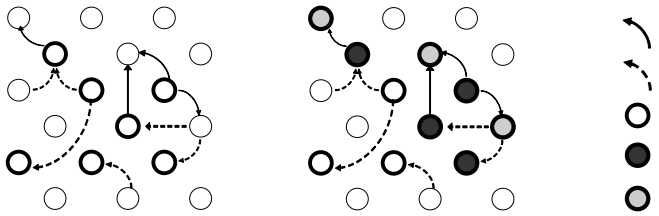

Abbildung 3.1: Ein Abhängigkeitsgraph vor (a.) und nach (b.) dem Fluten

4 Experimentelle Ergebnisse und Ausblick

Der Linkage Flooding Algorithmus wurde als eine unabhängige Softwarekomponente in Java realisiert, um sie einer Vielzahl von Client-Applikationen verfügbar zu machen [Ri03]. Eine Integration wurde in einem bestehenden Portal zur Dokumenten-Verwaltung und Skill-Management, dem infoAsset Broker [iA01], vorgenommen. Die Testdaten bildet eine persönliche Musikbibliothek, die aus der Integration zweier Bestände mit jeweils 60 Liedern - von denen 50 identisch sind, jedoch teilweise unterschiedlich und fehlerhaft benannt - hervorgegangen ist Die Lieder befinden sich in Alben, wieder mit zum Teil fehlerhafter Rechtschreibung, und in Genre-Taxonomien, deren Kategorien in beiden Bibliotheken unterschiedlich betitelt sein können. Interpreten werden als Zeichenkettenattribute von Liedern betrachtet. Beide Bibliotheken stehen bereits in identischen Formaten zur Verfügung, so dass die Vorverabeitung und das Schema Matching entfallen. Für das Vergleichen von Liedern bzw. deren Metadaten reicht die Betrachtung ihrer lokalen Attribute. Die Funktion *f* aus Gleichung 3.1 wird als Fuzzy-Prädikat realisiert, das wahr ist, wenn sich Liedtitel und Interpretenname zweier Lieder gemäß der Jaro-Metrik in einem mittleren Maß ähneln oder sich Interpretennamen stark ähneln und Titel in einem geringeren Maße. Mit dieser Funktion

können gegebene Datensatzpaare eindeutig als Links oder Nolinks identifiziert werden; eine Betrachtung von vernetzten Informationen ist nicht nötig. Bei der Suche nach Kandidaten kann der Linkage Flooding Algorithmus jedoch bereits helfen. Mittels einer SNM-Iteration mit dem Liednamen als Schlüssel werden die meisten, jedoch nicht alle Kandidaten gefunden. Durch die Tatsache, dass auch Alben verglichen werden und in diesem Zusammenhang ebenso die enthaltenen Lieder, erzeugt der Algorithmus auf natürliche Weise neue Kandidaten für den Vergleichsvorgang. Musikalische Genres mit ihren Verbindungen zu Untergenres sowie enthaltenen Liedern stellen ein gutes Anwendungsgebiet für das Linkage Flooding Verfahren dar. Oft sind semantische Duplikate hier nämlich ausschließlich durch vernetzte Informationen zu erkennen. Mit Hilfe einer Vergleichsfunktion *f* in Form eines komplexen Fuzzy-Prädikates, das Titel, Unter- und Obergenres sowie die eigentlichen Lieder eines Genres beachtet, können alle Duplikate identifiziert werden. Insbesondere an dieser Stelle hat sich die allgemeinere Vergleichsfunktion in Gleichung 3.1 bewährt: durch die Verwendung eines Fuzzy-Prädikates konnte genau zwischen tatsächlichen Duplikaten und lediglich ähnlichen Genres unterschieden werden. Im Suchvorgang für Genres wird nur eine kleine Anzahl Duplikate gefunden; Genrepaare wie „Hip-Hop“ und „Rap“ fallen durch das Raster. Diese werden aber durch das Linkage Flooding Verfahren zur Kandidatenmenge hinzugefügt, wenn die Untergenres verglichen werden. Das gesamte Verfahren konvergiert nach zehn Iterationen und braucht dafür 4 Sekunden auf einem Pentium III 850Mhz PC. Durch die Optimierung mittels Quellen und Senken konnte die Menge an Vergleichspaaren von etwa 8000, die das Similarity Flooding Verfahren benötigt, auf 1600 reduziert und der Rechenaufwand so deutlich verringert werden.

Das in diesem Paper vorgestellte Linkage Flooding Verfahren kann die Basis einer Vielzahl von Vertiefungen, Erweiterungen und softwaretechnischen Entwicklungen sein. So bietet die entwickelte Softwarekomponente bereits die Möglichkeit, Linkage Flooding für vernetzte Informationsbestände beim Data Cleaning oder auch bei Synchronisationsszenarien einzusetzen (Details finden sich in [Ri03]).

Literaturverzeichnis

[CC02] Christen, P.; Churches, T.: Probabilistic Name and Address Cleaning and Standardisation. *The Australasian Data Mining Workshop*, 2002.

[FS69] Fellegi, I.P.; Sunter, A. B.: A theory for record-linkage. *Journal of the American Statistical Association*, Vol. 64: 1183--1210, 1969.

[HS95] Hernandez, M. A.; Stolfo, S. J.: The Merge/purge Problem for Large Databases. In *SIGMOD Conference*, S. 127-138, 1995.

[iA01] The infoAsset Broker - Technical White Paper, ID 0110-011, infoAsset AG, http://www.infoasset.de, Hamburg, September 2001.

[Le02] Lehel, V.: Synchronisation von Informationsobjekten zwischen Portalen. Diplomarbeit, Technische Universität Hamburg-Harburg, http://wwwmatthes.in.tum.de, 2002.

[MGR02] Melnik, S.; Garcia-Molina, H.; Rahm, E.: Similarity flooding: A versatile graph matching algorithm. In *Proc. 18th ICDE Conference*, 2002.

[Ri03] Riedel, S.: Entwicklung eines Modells, Verfahrens und softwaretechnischen Rahmenwerks für Record Linkage in semantischen Netzen. Diplomarbeit, Technische Universität Hamburg-Harburg, 2003.