Linkage Flooding

Vanda Lehel, Florian Matthes, Sebastian Riedel

Technischer Bericht

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 simultanen Duplikaterkennung bei Zitatinformationen und Personendaten berichtet.

1 Problemstellung

Immer dann, wenn die Informationen in einem System von unterschiedlichen Benutzern erfasst und gepflegt oder auch automatisch importiert werden, können Duplikate entstehen. Dies führt zu einem Bedarf nach dateninhaltsorientierter Fusion. Beispielsweise können in einem System Publikationsinformationen erfasst werden. Diese enthalten zum einen Informationen über die Autoren (Name, Adresse, Homepage) sowie Metadaten über einzelne Publikationen. Vor allem wenn die entsprechenden Daten manuell erfasst wurden, steigt die Wahrscheinlichkeit für das Vorhandensein fehlerhafter Informationen, die zu semantischen Duplikaten im Informationsbestand führen. Ein ähnliches Szenario ist der Abgleich von Publikationsinformationen eines Lehrstuhls gegenüber persönlich gesammelten Literaturreferenzen, wo zunächst ebenfalls nach Duplikaten gesucht wird.

Als Konsequenz ergibt sich die Notwendigkeit für ein Suchverfahren nach Duplikaten, das in unterschiedlichen Szenarien zum Einsatz kommt. Im Kontext von Datensätzen bedeutet *Record Linkage* die Suche nach Datensätzen, die sich auf dieselbe semantische Entität beziehen (vgl. [FS69]). Dabei werden identische oder auch nur ähnliche Datensätze gesucht und als Datensatzpaar gebunden (*Bindungspaar*). Die Bindung von semantisch identischen Datensätzen ist bei der Integration zweier Datenbestände oder der Synchronisation von Informationsobjekten zwischen Datenbeständen [Le02]

erforderlich. Das Eliminieren von Duplikaten aus einem Datenbestand (Data Cleaning) zur Erhöhung der Datenqualität bedingt ebenfalls die Anwendung eines Record Linkage Verfahrens.

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 (z. B. Adresse), sondern auch von Referenzen auf andere Datensätze (z. B. Liste der Veröffentlichungen) ab. Zwei Publikationen sind beispielsweise wahrscheinlicher identisch, wenn sich ihre Autoren gleichen, und zwei Autoren sind wahrscheinlicher identische, wenn sich die menge ihrer Publikationen überlappt. Zusätzlich können den Publikationen Begriffe als Stichwörter zugeordnet werden. Die Begriffe bilden eine eigene Taxonomie, und jeder Begriff kann mit einer Vielzahl anderer Begriffe über hierarchische sowie Synonym-Beziehungen oder auch durch Querverweise vernetzt sein. Neben den Anforderungen, die solche komplex vernetzten Strukturen innerhalb der Informationsbestände aufgrund ihrer oftmals rekursiven Natur an ein Record Linkage Verfahren stellen, können sie das Binden auch vereinfachen. Sie ermöglichen nämlich ein natürliches Suchverfahren, das mit einer Menge von Bindungspaaren beginnt und restliche Kandidaten mittels der Verbindungen im vernetzten Informationsbestand bestimmen kann (vgl. Abschnitt 3).

Nach einer Erläuterung der Schritte des Record Linkage Prozesses mit den jeweils angewandten Methoden (Abschnitt 2) wird in Abschnitt 3 ein Algorithmus vorgestellt, der eine Implementierung dieses Prozesses darstellt, und speziell auf 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 als probabilistischer, endlicher Zustandsautomat verwendet werden. Für Zeichenketten lassen sich gut Stemming-Techniken verwenden, die jeweils den Wortstamm extrahieren (vgl. [Po80]).

Im zweiten Schritt, dem **Schema Matching**, werden die Elemente der Metamodelle von A und B gebunden. Schema Matching [MGR02] kann ebenfalls als eine Record Linkage Problemstellung aufgefasst werden, in der jedoch vielmehr ähnliche anstatt doppelte

Datensätze gesucht werden, diese dafür in einer vernetzten Umgebung. Für einen Vergleich gängiger Schema Matching Verfahren sei auf [MBR01] verwiesen.

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. Ein trivialer Suchalgorithmus ergibt sich im Falle $A = B$ durch die Symmetrie der betrachteten Relationen: Wenn man (a,b) berücksichtigt hat, erübrigt sich die Aufnahme von (b,a) in den Suchraum. Außerdem können auch alle reflexiven Paare (a,a) bei der Suche übersprungen werden. Am wichtigsten ist die Menge von Suchheuristiken, die jeweils für einen Datensatz a aus A eine Menge von Datensätzen aus B auswählt, die sich bezüglich eines Distanzmaßes „in der Nähe" befinden. Die gängigen Suchheuristiken versuchen stets, die zu bindenen Datenbestände zu sortieren, um im Anschluss mittels der entstandenen Suchstrukturen, z.B. Listen oder Cluster, ähnliche Datensätze zusammenzufassen und diese dann genauer zu vergleichen. Sehr verbreitet ist die *Sorted Neighbourhood Method (SNM)*, erstmals vorgestellt in [HS95]. Sie nutzt die Tatsache, dass in heutigen Datenbanken Einträge sehr schnell nach definierten Schlüsseln sortiert werden können, und dass sich doppelte Datensätze in diesen sortierten Listen nahe beieinander befinden. Über die sortierte Liste wird dann ein Fenster fester Größe geschoben, in dem das mittlere Element mit allen weiteren Objekten im Fenster verglichen wird. Anstatt die Datensätze lexikalisch zu ordnen, um Kandidatenpaare zu finden, können auch komplexere Metriken, wie z.B. die String-Edit-Distanz (s.u.), verwendet werden, um den Datenbestand zu sortieren. Eine Reihe von Verfahren zur Suche in metrischen Räumen findet sich in [Av01].

Anschließend folgt in Schritt 4 der **Vergleich** der Datensatzpaare aus dem Suchraum S, wobei in der Praxis das Suchen und Vergleichen oft miteinander kombiniert werden, um den Speicherbedarf des Record Linkage Prozesses zu verringern. Das Vergleichen findet tatsächlich auf vier verschiedenen Ebenen statt, für die jeweils eigene Vergleichsverfahren existieren, und zwar auf Literalen, Datensätzen, typisierten Datensatzmengen sowie global auf dem gesamten Datenbestand. Der Vergleich von Literalen erfolgt im allgemeinen über Metriken, wie die String-Edit-Distanz für Zeichenketten (vgl. [Le66]), in der die Anzahl der Einfüge-, Lösch- sowie Ersetzoperationen gezählt werden, um von der Zeichenkette s_1 zu s_2 zu gelangen, oder die Jaro-Metrik, die auf der Anzahl und Reihenfolge von gemeinsamen Zeichen in s_1 und s_2 basiert. Auf der Ebene der Datensätze werden die Ergebnisse des Vergleichens von Attributen und Referenzen kombiniert (also die *Vergleichsfunktionen* für Literale), um einen Vergleichswert für ein Datensatzpaar zu berechnen. Die am nächsten liegende Möglichkeit, n Vergleichswerte für Attribute und Referenzen zu kombinieren, ist ihre gewichtete Addition. Eine andere Möglichkeit bilden regelbasierte Systeme wie in [LLL00], wo ein Domänenexperte von vornherein Regeln aus Und- und Oder-Verknüpfungen von binären Prädikaten festlegt, nach denen Datensätze verglichen werden. Für typisierte Datensatzmengen gilt entsprechend, dass sich Vergleichswerte aus der Kombination der Vergleichsfunktionen für die einzelnen Datensätze aufstellen lassen. Um die konsistenten Bindungswerte für die Datensätze des gesamten Datenbestandes zu berechnen, werden *globale Vergleichsverfahren* als Methoden eingesetzt. Sie bilden die zwei Datenbestände A und B auf eine *Bindungsmatrix* M aus

$\Re^{n \times m}$ ab, wobei *n* und *m* die Kardinalitäten von *A* und *B* sind. Das Feld $M_{i,j}$ entspricht dabei dem Bindungswert zwischen dem *i*-ten Datensatz aus *A* und dem *j*-ten Datensatz aus *B*. Eine wichtige globale Methode stellt das Similarity Flooding Verfahren [MGR02] dar, bei dem die Abhängigkeitsmatrix zwischen Datensatzpaaren als Sekundärstruktur benutzt wird, durch die während des iterativen Algorithmus die sich beeinflussenden Vergleichswerte „fließen". Der in Abschnitt 3 des Papiers beschriebene Linkage Flooding Algorithmus basiert auf diesem Verfahren.

In Schritt 5 wird das **Filtern** durchgeführt: Nachdem ein Verfahren konsistente Bindungswerte mit Hilfe von Vergleichsfunktionen für den gesamten Datenbestand berechnet hat, wird das Bindungsergebnis unter Berücksichtigung von globalen Eigenschaften der Bindungsrelation angepasst. Ein *Filter* ist formal eine Funktion, die eine Bindungsmatrix auf eine andere abbildet. Berechnet werden kann dabei z. B. die transitive Hülle, um zusätzlich die Transitivität der Duplikat-Beziehung auszunutzen.

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 der Vergleichsfunktion c_R. Sucht man doppelte Datensätze, werden nur Paare (a,b), für die $c_{Equivalence}(a,b)$ 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. Dass hier nicht nur Schwellwerte sondern auch Nutzer über Bindungen entscheiden müssen, hängt von den potentiellen Folgen dieser Entscheidungen ab. Es muss beispielsweise in einem Data Cleaning Szenario vermieden werden, fälschlicherweise Datensätze zu löschen, bei denen es sich gar nicht um Duplikate 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. Allerdings wurden für das Linkage Flooding die folgenden zusätzlichen Erweiterungen vorgenommen:

- Verallgemeinerung der Iterationsgleichung, so dass anstatt der gewichteten Summe (vgl. Abschnitt 2) 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.

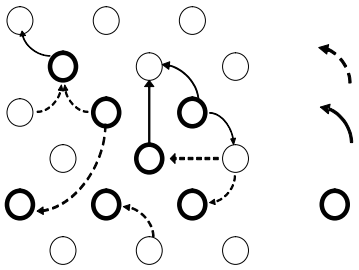

Abbildung 3.1: Ein Abhängigkeitsgraph vor dem Fluten

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 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 3.2 sind neue Quellen durch grau gefüllte Kreise dargestellt. Man erkennt, dass einige der Senken aus Abbildung 3.1 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 zum Tragen. In Abbildung 3.2 erkennt man auch die neuen Senken an den Enden dieser durchgezogenen Linien.

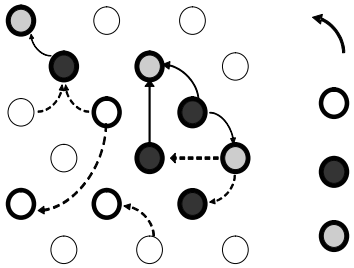

Abbildung 3.2: Ein Abhängigkeitsgraph nach dem Fluten

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 bilden zum einen die Lehrstuhl-Bibliografiedatenbank mit ca. 1300 Publikationen und 400 Autoren sowie eine persönliche Bibliografie (ca. 300 Publikationen und 150 Autoren). Beim Import in das Portal entstehen sowohl neue *Document*-Objekte, die die Metadaten der Publikationen repräsentieren, wie auch *Person*-Objekte, die die Autoren der jeweiligen Arbeiten kennzeichnen. Beide Quellen enthalten zum Teil dieselben Literaturreferenzen, so dass nach dem Import Duplikate entstehen. Linkage Flooding wird durchgeführt, um diese Duplikate im Datenbestand zu finden. Im Portal müssen für die Informationsobjekt-Typen *Document* und *Person* zur Duplikaterkennung adäquate Vergleichsfunktionen auf Datensatzebene definiert werden. Diese berücksichtigen wesentliche Attribute der Objekte wie Titel von Dokumenten oder Vornamen und Nachnamen von Personen. Für das Experiment wird der Linkage Flooding Algorithmus ohne manuelle Intervention durchgeführt. Anschließend werden die Vergleichsfunktionen manuell optimiert und das Verfahren wiederholt, bis die

Ergebnisse zufrieden stellend sind. Damit erhält man geeignete Parametrisierungen der Vergleichsfunktionen, die für die künftige automatische Durchführung des Linkage Flooding Algorithmus benutzt werden können.

Es zeigt sich, dass die Titel stets bis auf sehr wenige Eingabefehler korrekt sind. Genau dann, wenn ihre Titel übereinstimmen, sind also zwei Dokumente Duplikate. Die wenigen Tippfehler fängt man z.B. durch die Berechnung der String-Edit-Distanz zwischen den Titeln ab. Bei der Durchführung des Linkage Flooding Verfahrens resultieren aber 5 falsche Links, also Paare, die fälschlicherweise als Links klassifiziert wurden. Dies rührt daher, dass die String-Edit-Distanz entscheidende Unterschiede in Versionsnummern nur als marginale Differenzen erkennt. Es liegt nahe, eine neue Metrik zu definieren, die Abhilfe schafft, in dem sie verstärkt das Tauschen und Einfügen von Ziffern berücksichtigt. Für Personen ist der Test auf Äquivalenz schwieriger. In Vornamen und Nachnamen findet man nämlich nicht nur Tippfehler, oft und besonders bei ausländischen Namen kennen die Verfasser der Metadaten die Schreibweisen nicht. Zudem werden Vornamen sehr oft abgekürzt. Vergleicht man dann einen Vornamen mit seiner abgekürzten Version, so errechnen die herkömmlichen Metriken große Abstände. Die Jaro bzw JaroWinkler-Metrik ist speziell dafür ausgelegt, mit Namen und ihren verschiedenen Schreibweisen umzugehen. Die Duplikatfunktion für Personen benutzt daher diese Metrik. Die beschriebene Konfiguration war letztendlich erfolgreich beim Binden der Datensätze, und so konnten bei den Dokumenten alle Duplikate korrekt gefunden sowie bei den Personen das Verhältnis von gefundenen zu insgesamt vorhandenen Duplikaten von 20% auf 98% erhöht 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

[Av01] Avez, C.; Navarro, E.; Baeza-Yates, G.; Marroqu, R.: Searching in metric spaces. ACM Computing Surveys, Vol. 33, Nr. 3, S. 271-321, 2001.

[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.

[Le66] Levenshtein, V. I.: Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707-710, 1966.

[LLL00] Lee, M.; Ling, T. W.; Low, W. L.: IntelliClean: A knowledge-based intelligent data cleaner. In *Knowledge Discovery an Data Mining*, S. 290-294, 2000.

[MBR01] Madhavan, J.; Bernstein, P. A.; Rahm, E.: Generic Schema Matching with Cupid. In *The VLDB Journal*, S. 49-58, 2001.

[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.

[Po80] Porter, M. F.: An algorithm for suffix stripping, *Program*. Vol. 14, Nr. 3, S. 130-137, 1980.