HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Workshop „Datenreinigung“ Duplikaterkennung

8.10.2009

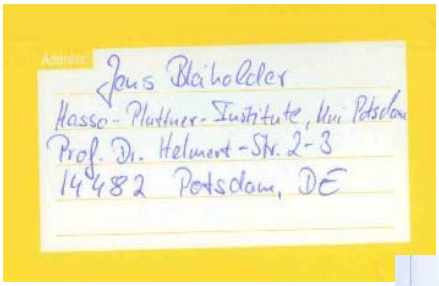
Felix Naumann

Überblick

- Das Problem der Duplikaterkennung
- Ähnlichkeitsmaße
  - Edit Distance et al.
- Algorithmen
  - Naiv
  - Blocking
  - Sorted-Neighborhood Methode
    - Naive, Multipass
    - Effizient
- Evaluierung

Wie entstehen Duplikate?

3

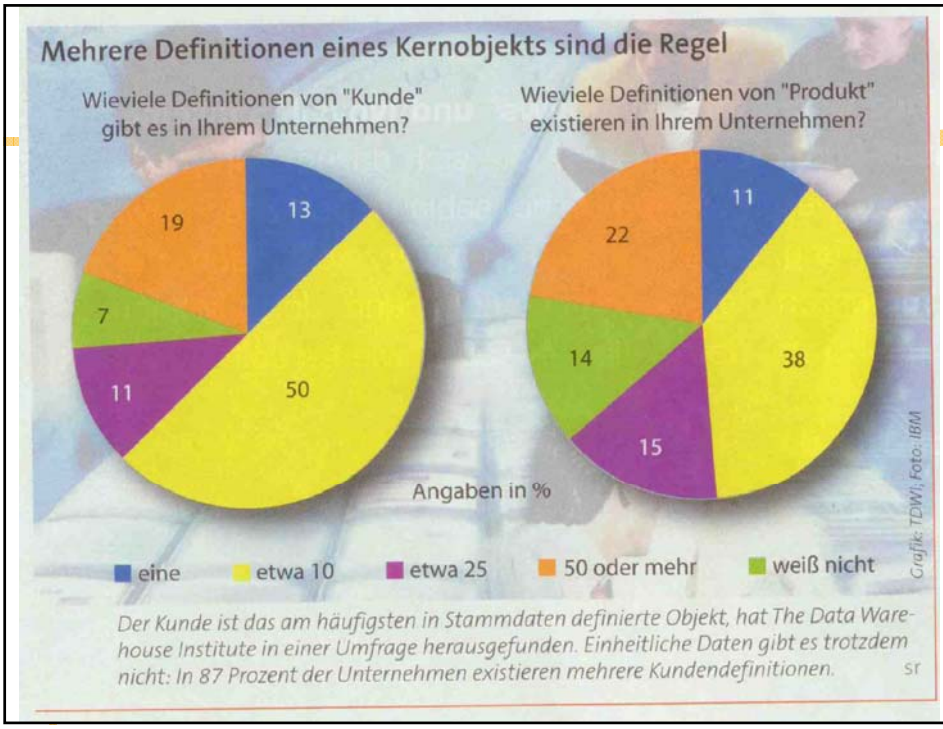


Original

Zugestellt

Wie entstehen Duplikate?

4



Duplikaterkennung

HPI Hasso Plattner Institut

6

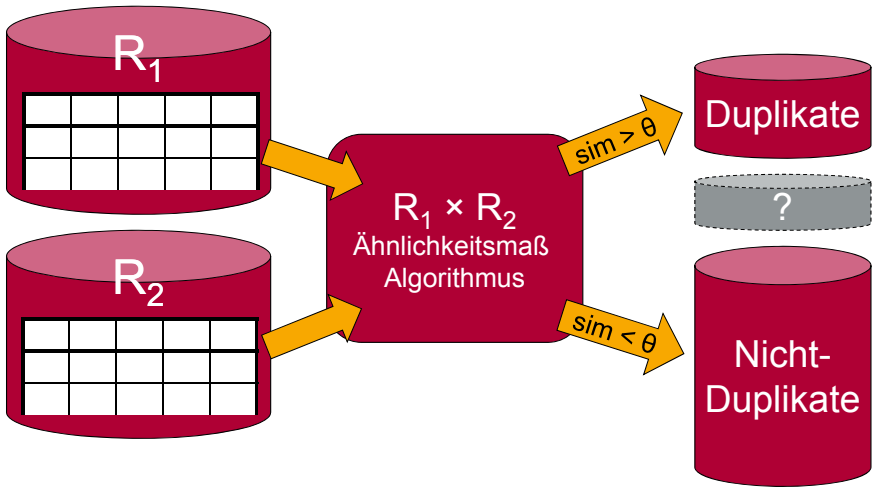
Duplikaterkennung ist das Finden mehrerer Repräsentationen desselben Realweltobjekts.

- Problem 1: Repräsentationen sind nicht identisch.
  - *Fuzzy duplicates*
- Lösung: Ähnlichkeitsmaße
  - Wert- und Datensatzvergleiche
  - Domänenunabhängig oder -abhängig

- Problem 2: Die Datenmenge ist groß.
  - Quadratischer Aufwand: Jedes Paar muss verglichen werden.
- Lösung: Algorithmen
  - Z.B. Vergleiche durch Partitionierung vermeiden

Duplikaterkennung

7



Wirkungen von Duplikaten

8

- Mehrfache Zusendung von Katalogen
- Rechnungen werden doppelt bezahlt
- Banken
  - Überschreiten des Kreditlimits wird nicht erkannt
- Lagerhaltung / Einkauf
  - Zu niedriger Lagerbestand einzelner Waren wird ausgewiesen.
  - Kein Ausnutzen von Mengenrabatten bei Bestellungen
- Gesamtumsatz eines Kunden bleibt unbekannt.
- Mehraufwand in der IT
- Sinkende Kundenzufriedenheit
- Potenziale und Gefahren nicht erkannt
- Inkorrekte Kennzahlen

Kunde	Umsatz
BMW	20.000
BaMoWe	5.000.000
Bayerische Motorenwerke	300.000
...	...

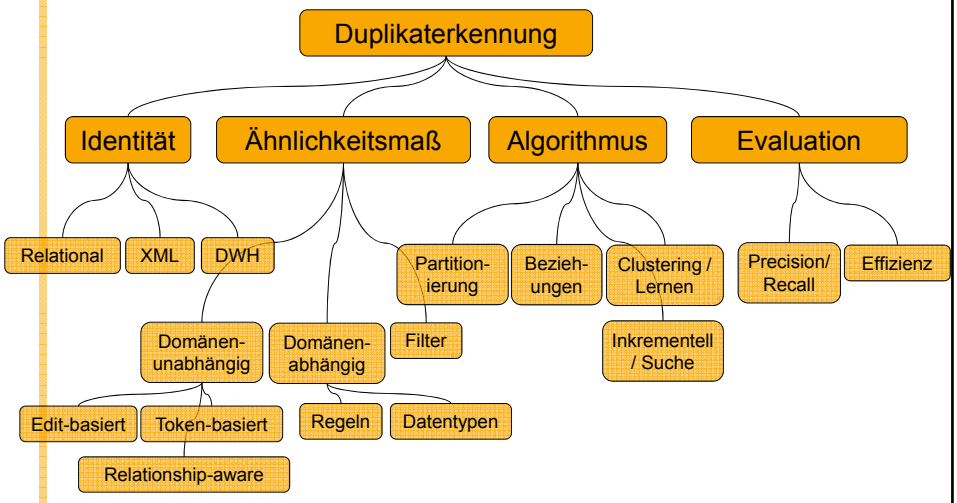
"Duplikaterkennung" hat viele Duplikate

9

Household matching
Dubletten
Duplicate detection
Mixed and split citation problem
Record linkage
Object identification
Match
Deduplication
Object consolidation
Fuzzy match
Entity resolution
Entity clustering
Approximate match
Identity uncertainty
Reference reconciliation
Merge/purge
Hardening soft databases
Householding
Reference matching

Duplikaterkennung

10



Überblick

- Das Problem der Duplikaterkennung
- Ähnlichkeitsmaße
  - Edit Distance et al.
- Algorithmen
  - Naiv
  - Blocking
  - Sorted-Neighborhood Methode
    - Naive, Multipass
    - Effizient
- Evaluierung

Token-basierte Ähnlichkeitsmaße

- Tokens
  - Words / Terms
  - n-grams
- Jaccard
  - |{gemeinsame token}| / |{alle token}|
- TFIDF
  - *Term frequency* (tf)
  - *Inverse document frequency* (idf)
  - TFIDF: log (tf+1) x log idf
  - Häufige Wörter haben niedriges Gewicht.
  - Ähnlichkeit ist Kosinus der Termvektoren, gewichtet durch TFIDF.
- …

Edit-basierte Ähnlichkeitsmaße

13

- Edit-Distanz / Levenshtein-Distanz [Levenshtein 1965]
  - Minimale Anzahl von Edit-Operationen um den einen String in den anderen umzuwandeln.
  - Domänenspezifische Kosten
- Jaro [Jaro 1989] / Jaro-Winkler [Winkler 1999]
  - Common letters within ½ string length
  - Transposed letters
- Soundex
  - 4-stelliger Code für jedes Wort
  - Verfügbar als Skalar-Funktion in DB2
- …

Edit Distance - Grundlagen

14

Maß zur Ermittlung des „Abstandes" zweier Zeichenketten

- Literatur: z.B.: [Kuk92]

Abstand := Anzahl an Operationen zur Überführung einer Zeichenkette S_1 in eine Zeichenkette S_2 durch

- Einfügung (Insert)
- Löschung (Delete)
- Ersetzung (Replace)
- „Übereinstimmung" (Match)

Edit Distance – Grundlagen

15

- Edit distance ist minimaler Abstand
- Fragen
  - Wie groß ist der Abstand?
  - Welches Transkript(e) entspricht diesem Abstand?
    - Nicht wichtig bei Datenreinigung
- Kosten pro Operation festlegen
  - Meist jeweils Kosten 1 (Insert, Update, Delete) bzw. 0 (Match)
  - Ggf. andere Kosten
    - Insert 1, Delete 1, Update 2
    - Auch: Abhängig von Buchstaben
      - Typewriter distance
      - Biologie

Edit Distance – Beispiel

16

- Beispiel „HASE" ⇒ „RASEN"
  - triviale Umformung durch Einfügung und Löschung mittels Leerzeichen # am Anfang und Ende
  - HASE##### ####RASEN
- Transkript: DDDDIIIII
- Kosten 9
  - Nicht minimal!

Edit Distance – Beispiel

17

- Beispiel „HASE" ⇒ „RASEN"
  - H → R durch Ersetzung (R)
  - A → A durch Übereinstimmung (M)
  - S → S durch Übereinstimmung (M)
  - E → E durch Übereinstimmung (M)
  - „ "→ N durch Einfügung (I)
- Transkript: RMMMI
- EditDistance(HASE,RASEN) = 2
  - Minimal! Woher weiß man das?

Edit Distance – Berechnung

18

Dynamische Programmierung

- Sei D(i,j) die edit-distance der Strings S_1 und S_2
- Falls $|S_1| = m$ und $|S_2| = n$ ist D(m,n) die (minimale) edit-distance
- Berechne D(m,n) durch Berechnung von minimalen Teillösungen für alle Kombinationen $i \in [0,m]$ und $j \in [0,n]$
- Prinzip der Optimalität: Bestes (minimales) Transkript zweier Teilstrings ist auch Teil des besten Gesamt-Transkripts.

Edit Distance – Berechnung

19

Edit-Distance-Matrix
(Initialisierung)

		H	A	S	E
	0	1	2	3	4
R	1				
A	2				
S	3				
E	4				
N	5				

$D(i, 0) = i$

$D(0, j) = j$

$D(i, j) = \min \{$

$D(i-1, j) + 1,$

$D(i, j-1) + 1,$

$D(i-1, j-1) + d(i, j)$
$\}$

wobei $d(i, j) = 0$ bei Gleichheit, $d(i, j) = 1$ sonst

Edit Distance – Berechnung

20

Edit-Distance-Matrix
(Berechnung)

		H	A	S	E
	0	1	2	3	4
R	1	1			
A	2				
S	3				
E	4				
N	5				

$D(i, 0) = i$

$D(0, j) = j$

$D(i, j) = \min \{$

$D(i-1, j) + 1,$

$D(i, j-1) + 1,$

$D(i-1, j-1) + d(i, j)$
$\}$

wobei $d(i, j) = 0$ bei Gleichheit, $d(i, j) = 1$ sonst

Edit Distance – Berechnung

21

Edit-Distance-Matrix (Berechnung)

		H	A	S	E
	0	1	2	3	4
R	1	1	2		
A	2				
S	3				
E	4				
N	5				

D(i, 0) = i

D(0, j) = j

D(i, j) = min {

D(i-1, j) + 1,

D(i, j-1) + 1,

D(i-1, j-1) + d(i, j)

}

wobei d(i, j) = 0 bei Gleichheit, d(i, j) = 1 sonst

Edit Distance – Berechnung

22

Edit-Distance-Matrix (Ergebnis)

		H	A	S	E
	0	1	2	3	4
R	1	1	2	3	4
A	2	2	1	2	3
S	3	3	2	1	2
E	4	4	3	2	1
N	5	5	4	3	2

Transkript durch Traceback rückwärts zum kleinst-möglichen Wert nach

- links = DELETE
- oben = INSERT
- diagonal = MATCH oder REPLACE

Edit Distance – Komplexität

23

- Vorteile:
- Maß für den Unterschied zweier Zeichenketten
- Ordnung der Treffer möglich, durch Normierung auf max (m,n).
- Liefert „gute" Ähnlichkeitswerte für Attributwerte.
- Einfach erweiterbar (unterschiedliche Gewichtung der Operationen)
- Nachteile:
- Quadratische Komplexität O(m x n)
  - m: Länge von S1
  - n: Länge von S2
  - Aufbau Matrix: m x n
  - Traceback: m + n
- Buchstabendreher werden relativ hoch bewertet.
- Nicht geeignet für Textdaten oder numerische Werte.

SOUNDEX

24

- Gleichklingende Wörter werden in einer identischen Zeichenfolge codiert.
  - Soundex („Naumann") = Soundex („Neuman") = N550
  - Soundex-Code für ein Wort besteht aus seinem ersten Buchstaben gefolgt von drei Ziffern
  - Ziffern repräsentieren die nach dem Anfangsbuchstaben folgenden Konsonanten
  - Ähnliche Laute besitzen den gleichen Code (B, F, P und V werden z.B. alle mit der Ziffer "1" codiert).
- Soundex wurde von Russell für die Indizierung der Familiennamen der Volkszählung (Census) in den USA entwickelt und 1918 patentiert.

Spezialisierte Ähnlichkeitsmaße

25

- Stimmen zwei Datumsangaben nicht überein, kann ein Wert für die Ähnlichkeit (zwischen 0 und 1) bestimmt werden:
  - Zerlegen beider Datumsangaben in die Komponenten: Tag, Monat, Jahr
  - Jede Übereinstimmung wird bewertet, z.B. Tag: 0,3; Monat: 0,3; Jahr: 0,4
  - Test, ob einer der häufigen Fehler vorliegt. Geringfügigere Fehler zuerst testen.
  - Liegt einer der häufigen Fehler vor, führt dies zur Aufwertung des Ähnlichkeitswertes, z.B. Vertauschung von Tag und Monat: 0,5
  - Bei immanenter Ähnlichkeit (1. / 1.1. / 1.1.00) ebenfalls Aufwertung, z.B. 0,1
- Beispiel: 6.9.2005 ⇔ 9.6.2005
  - Jahr identisch: 0,4; Tag und Monat vertauscht: 0,5 → Summe = 0,9

Gesamtähnlichkeitsmaß

26

- Ähnlichkeit pro Attribute ermitteln.
- Dann zusammenführen
  - Durchschnitt
  - Equational theory
- Problem: Was tun mit fehlenden Werten

Equational theory (Gleichungstheorie)

27

- diktiert die Logik der Domänenäquivalenz oder Kettenäquivalenz.
- benutzt die deklarative regel-basierte Sprache.
- benutzt vordefiniertes Ähnlichkeitsmaß (Abstandsfunktion) mit vordefinierten Grenzwert.
  - Edit distance
  - Phonetic distance
  - Typewriter distance

Equational theory (Gleichungstheorie)

28

Given two records, r_1 and r_2
IF last_name(r_1) = last_name(r_2)
AND edit_distance(first_name(r_1), firstname(r_2)) < 5,
AND address(r_1) = address(r_2)
THEN r_1 is equivalent to r_2

Given two records, r_1 and r_2
IF (ID(r_1) = ID(r_2) OR last_name(r_1) = last_name(r_2))
AND address(r_1) = address(r_2)
AND city(r_1) = city(r_2)
AND (state(r_1) = state(r_2) OR zip(r_1) = zip(r_2))
THEN r_1 is equivalent to r_2

Schwellwerte (threshold)

29

- Schwer zu finden
- Tradeoff zwischen precision und recall
- Viele probieren!
- Idee: Sortieren nach Ähnlichkeit und irgendwo abbrechen

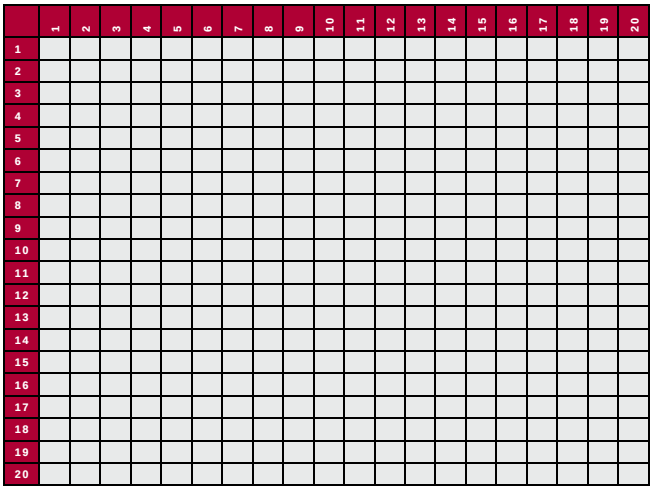
Überblick

30

- Das Problem der Duplikaterkennung
- Ähnlichkeitsmaße
  - Edit Distance et al.
- Algorithmen
  - Naiv & Blocking
  - Sorted-Neighborhood Methode
    - Naive, Multipass
    - Effizient
- Evaluierung

Record Pairs as Matrix

31



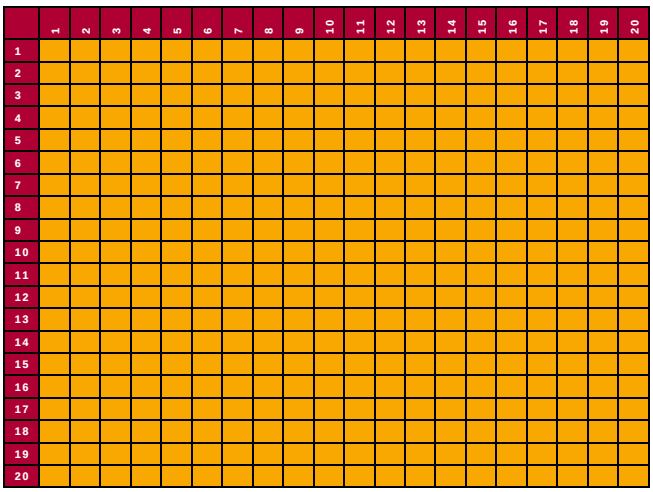
Naiver Algorithmus

32

- Zwei geschachtelte Schleifen
  - Java
  - SQL
    - SELECT C1.*, genID(C1,C2)
      FROM R as C1, R as C2
      WHERE sim(C1,C2) > theta
- Aufwand
  - Nur inter-source Duplikate: 300.000 * 400.000 + 300.000 * 500.000 + 400.000 * 500.000 = 470.000.000.000
  - Mit intra-source Duplikaten: $1.200.000^2$ = 1.440.000.000.000
  - Eigentlich: $(n^2-n)/2$

Number of comparisons: All pairs

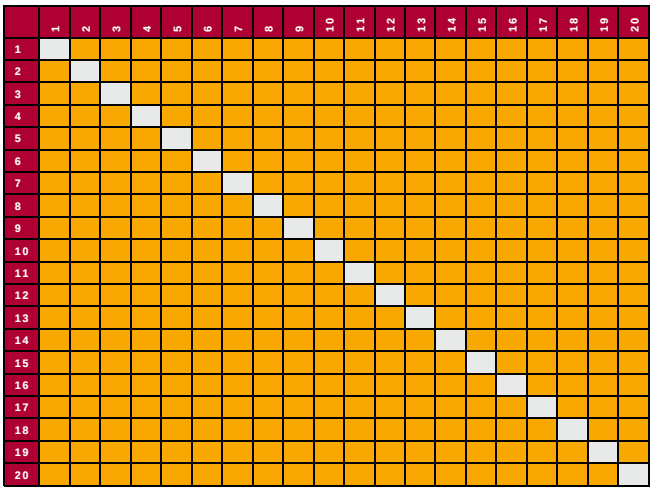
33



400 comparisons

Reflexivity of Similarity

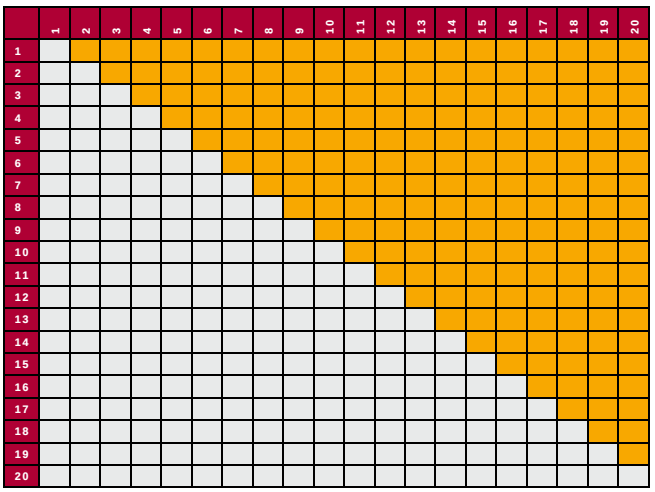
34



380 comparisons

Symmetry of Similarity

35



190 comparisons

Complexity

36

Still: Too many comparisons

- 10.000 customers => 49.995.000 comparisons
  - $(n^2 - n) / 2$
  - Each comparison is expensive (complex similarity measures).

Idea: Avoid comparisons by heuristics

- Filtering of records
- Partitionierung

Partitioning / Blocking

37

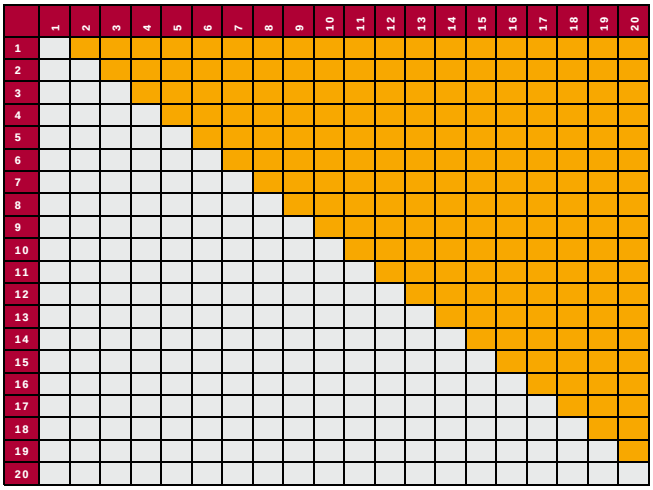
- Partition the records (horizontally) and compare pairs of records only within a partition.
  - Partitioning by first two zip-digits
    - Ca. 100 partitions in Germany
    - Ca. 100 customers per partition
    - => 495.000 comparisons
  - Partition by first letter of surname
  - …

Source: wikipedia.de

- Idea: Partition multiple times by different criteria.
  - Then apply transitive closure on discovered duplicates.

Records sorted by ZIP

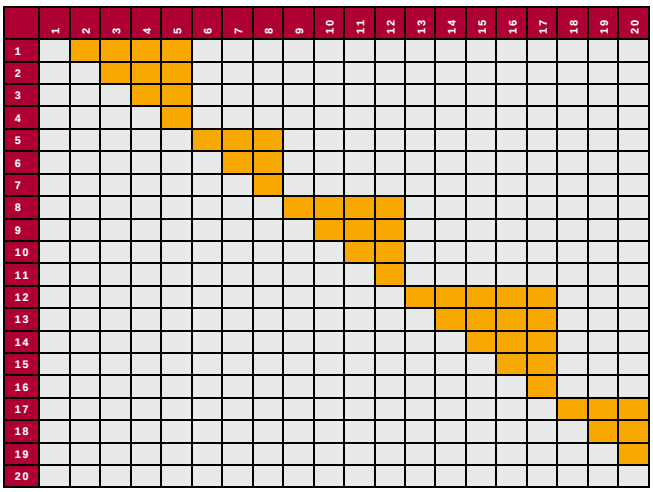
38



190 comparisons

Blocking by ZIP

39



47 comparisons

Blocking / Gruppierung / Partitionierung

40

- Die Kunst: Geeignete Kriterien finden
  - Gruppiere nach Stadt
  - Gruppiere nach erstem Buchstaben des Nachnamen
  - Gruppiere nach PLZ und Vorname
  - Jede Gruppe sollte in den Hauptspeicher passen
  - Gruppierungskriterium sollten möglichst fehlerfreie Attribute sein
- Praktisch:
  - Sortierung mit SQL (nicht GROUP BY)
- Weiter: Mehrere Durchläufe mit unterschiedlichen Kriterien

Mehrfache Partitionierung

41

- Probleme
  - Datenfehler in PLZ
  - Umzug
  - => Duplikat nicht erkannt
- Idee:
  - Partitioniere mehrfach nach unterschiedlichen Kriterien
    - PLZ, Nachname, zusammengesetzter Schlüssel
  - Bilde Transitive Hülle
    - Durchgang 1: A = B
    - Durchgang 2: B = C
    - Transitive Hülle: A = C

Überblick

42

- Das Problem der Duplikaterkennung
- Ähnlichkeitsmaße
  - Edit Distance et al.
- Algorithmen
  - Naiv & Blocking
  - Sorted-Neighborhood Methode
    - Naive, Multipass
    - Effizient
- Evaluierung

Die Sorted Neighborhood Methode

43

- Input:
  - Tabelle mit N Tuplen
  - Ähnlichkeitsmaß (basierend auf Edit distance)
- Output:
  - Klassen (clusters) der äquivalenten Tupel (= Duplikate)
- Problem: Viele Tupel
  - Vergleich eines jeden Tupelpaares zu aufwendig (Effizienz).
  - Tabelle passt nicht in den Speicher (Skalierbarkeit).

Sorted Neighborhood

44

Idee

- Daten geschickt partitionieren.
- Nur innerhalb dieser Partitionen Duplikate suchen.

Algorithmus nach [HS98]

1. Create Key:
   - Schlüssel mittels relevanter Feldern erzeugen.
     - ◇ Sequenz einer Teilmenge von Attributen oder der Teilketten innerhalb der Attribute.
     - ◇ Effektivität des Algorithmus ist von Schlüsselauswahl abhängig.
     - ◇ Schlüssel ist nur virtuell und nicht eindeutig.
       - Wird nur für Sortierung benutzt.
2. Sort:
   - Daten nach dem Schlüssel sortieren.
3. Merge:
   - Fenster (der Größe *w*) über sortierte Tupel schieben.
   - Nur Tupel innerhalb des Fensters miteinander vergleichen.

Sorted Neighborhood Methode [HS98]

45

ID	Title	Year	Genre
17	Mask of Zorro	1998	Adventure
18	Addams Family	1991	Comedy
25	Rush Hour	1998	Comedy
31	Matrix	1999	Sci-Fi
52	Return of Dschafar	1994	Children
113	Adams Family	1991	Comedie
207	Return of Djaffar	1995	Children

Create key

1.

ID	Key
17	MSKAD98
18	DDMCO91
25	RSHCO98
31	MTRSC99
52	RTRCH94
113	DMSCO91
207	RTRCH95

2. Sort

ID	Key
18	DDMCO91
113	DMSCO91
17	MSKAD98
31	MTRSC99
25	RSHCO98
52	RTRCH94
207	RTRCH95

Merge

3.

ID	Key
18	DDMCO91
113	DMSCO91
17	MSKAD98
31	MTRSC99
25	RSHCO98
52	RTRCH94
207	RTRCH95

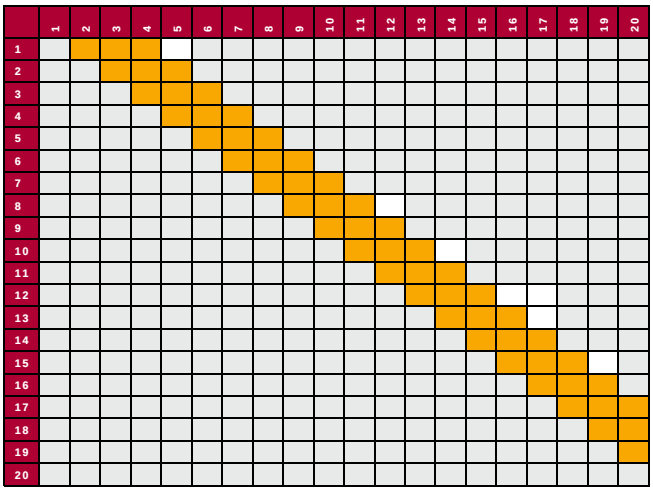
classify(18,113) → duplicates

classify(52,207) → duplicates

SNM by ZIP (window size 4)

HPI Hasso Plattner Institut

46



54 comparisons

Sorted Neighborhood – Aufwand

47

Aufwand

- N : Anzahl der Tupel, w: Fenstergröße (window)
- Theoretisch:
  - O(N) + O(N logN) + O(w N) = O(N logN)
    - ◇ bei w < logN; O(wN) sonst
- Praktisch:
  - Drei Läufe über die Daten auf der Festplatte

Sorted Neighborhood – Aufwand

48

Kommentare

- Wahl des Schlüssels
  - Formulierung durch Experten
  - Aufwändig
  - Schwer vergleichbare Ergebnisse
  - Für Effektivität entscheidend
- Wahl der Fenstergröße
  - w = N : $O(N^2) \Rightarrow$ max. accuracy & max. Zeit
  - w = 2 : $O(N) \Rightarrow$ min. accuracy & min. Zeit
- Entscheidung ob ein Duplikat vorliegt, ist eine komplexe Berechnung (edit distance).

Sorted Neighborhood – Multipass Verfahren

49

- Problematische Schlüsselwahl
  - Beispiel: Schlüssel beginnt mit ID
    - r_1: 193456782 und r_2: 913456782
- Problemlösung 1:
  - Vergrößerung des Fensters: $w \rightarrow N$
- Problemlösung 2:
  - Multipass Verfahren

Sorted Neighborhood – Multipass Verfahren

50

- Mehrmalige Durchführung von Sorted Neighborhood Methode mit verschiedenen Schlüsseln
- *w* relativ klein
- Transitive Hülle auf Ergebnissen jedes Durchgangs:
  - Equivalent(a, b) && Equivalent (b, c)
    $\Rightarrow$ Equivalent (a, c)
  - Dadurch werde neue Duplikate gefunden.

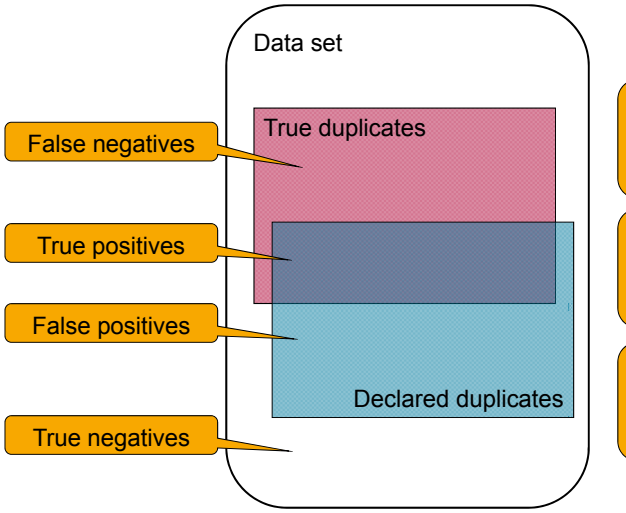
Überblick

51

- Das Problem der Duplikaterkennung
- Ähnlichkeitsmaße
  - Edit Distance et al.
- Algorithmen
  - Naiv & Blocking
  - Sorted-Neighborhood Methode
    - Naive, Multipass
    - Effizient
- Evaluierung

Precision & Recall

52

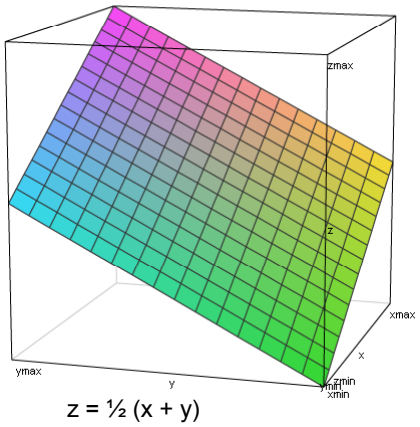


$$\text{Precision} = \frac{\text{True positives}}{\text{Declared duplicates}}$$

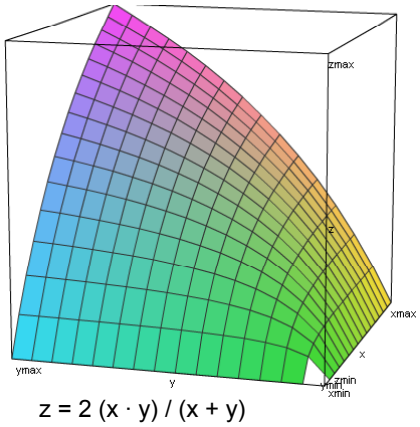
$$\text{Recall} = \frac{\text{True positives}}{\text{True duplicates}}$$

$$\text{F-Measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Arithmetisches Mittel („Durchschnitt“) vs. Harmonisches Mittel („F-Maß“)

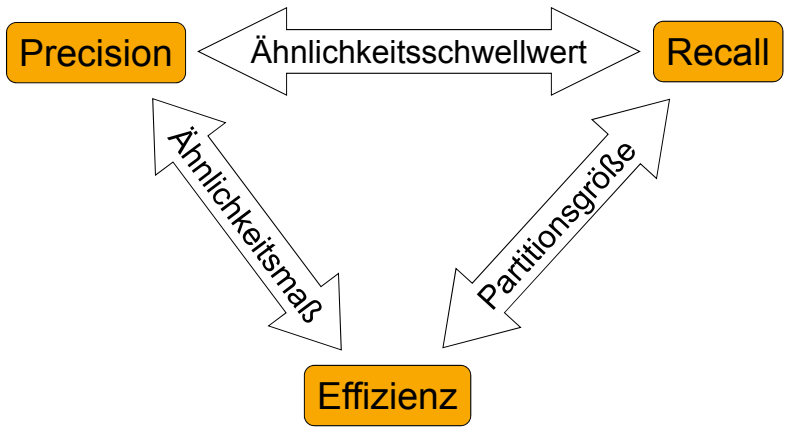


$z = ½ (x + y)$



$z = 2 (x \cdot y) / (x + y)$

Duplikaterkennung – Zielkonflikte



Zusammenfassung des Vorgehens

55

Gegeben zwei Tupelmengen A und B

Kernidee:

- Bilde Kreuzprodukt aller Tupel.
- Für jedes Paar berechne Ähnlichkeit
  - Z.B. bzgl. Attributwerte
  - Z.B. bzgl. Beziehungen zu anderen Tabellen (Fremdschlüssel)
  - usw.
- Wähle Duplikatpaare aus
  - Ähnlichste Paare bis Schwellwert
  - Nebenbedingungen
- Bilde Duplikatcluster
  - Transitive Hülle

Edit Distance

Containment metric

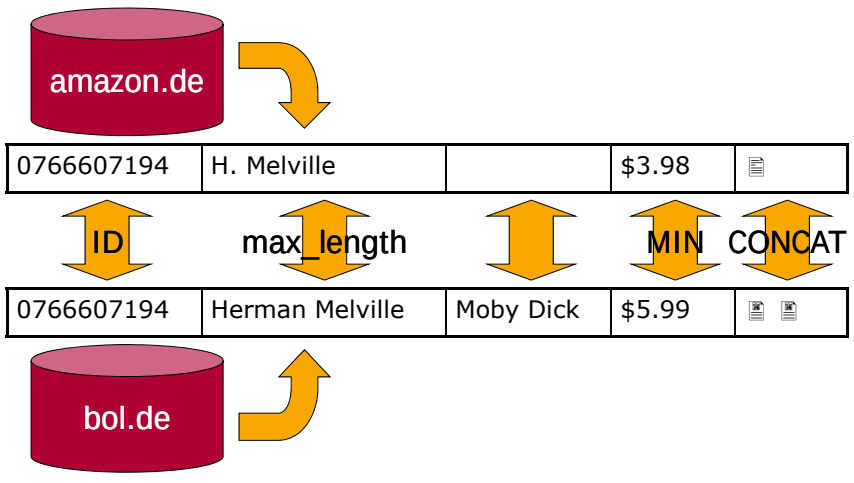
Sorted Neighborhood Methode

Probleme

- Anzahl und Komplexität der Vergleiche (Effizienz)
- Güte des Ähnlichkeitsmaßes (Effektivität)
- Große Datenmengen (Skalierbarkeit)

Datenfusion

56



Abgabe bis 18 Uhr

- Liste mit IDs der Duplikatpaare
- Kleinere ID eines Paares zuerst
  - Numerisch!
- Sonstige Sortierung ist egal
- Ein Paar pro Zeile
- Trennung der IDs durch Komma
  - 3, 25
  - 5, 11
  - 2, 39
  - 45, 123
  - …

- Trick 1
  - Leere Liste abgeben
  - Precision = 1
  - Aber Recall schlecht
- Trick 2
  - Alle Paare abgeben
    - Passt nicht in E-Mail
  - Recall = 1
  - Aber precision schlecht

Literatur

- [RD00] Data Cleaning: Problems and Current Approaches, E. Rahm and H.H. Do, IEEE Bulletin 23(4), 2000.
- [Kuk92] Technique for automatically correcting words in text, ACM Computing Survey 24(4), 1992, Karen Kukich
- [HS98] M. Hernandez and S. Stolfo Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery, 2(1): 9-37.
- [ME97] Alvaro E. Monge, Charles Elkan: An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. In Proceedings of the Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'97)