Duplikaterkennung

Komplexe Informationssysteme

Fabian Panse

panse@informatik.uni-hamburg.de

Universität Hamburg

Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Duplikaterkennung

Duplikaterkennung - Motivation

Daten sind häufig von Qualitätsmängeln betroffen:

- Unvollständigkeit
- Fehlerhaft

Quelle: Felix Naumann, Melanie Herschel [NH10]

Duplikaterkennung - Motivation

Reinigung einer Datenbank/Quelle:

- Duplikate als Qualitätsproblem
- Schwerwiegende Folgen möglich

tid	Name	Alter	Wohnort
t_1	Max Mustermann	27	Hamburg
t_2	Maximilian Musterman	28	Hamburg - Eimsbüttel
t_3	Max Muster	30	Hamburg
t_4	Johannes Frisch	45	Berlin
t_5	Anne Gruber	41	Harburg
t_6	Kieser Klaus	53	Harburg
t_7	Gruber Annemarie	41	Hamburg
t_8	Christian Fabian	21	Berlin

Duplikaterkennung - Motivation

Semantisch korrekte Verknüpfung verschieder Quellen:

Quelle 1

tid	Name	Wohnort
t_{11}	Maximilian Musterman	Eimsbüttel
t_{12}	Max Muster	Hamburg
t_{13}	Johannes Frisch	Berlin
t_{14}	Gruber Annemarie	Hamburg
t_{15}	Christian Fabian	Berlin

Quelle 2

tid	Name	Studiengang
t_{21}	Max Mustermann	Mathematik
t_{22}	Max Muster	Informatik
t_{23}	Hannes Frisch	Biologie
t_{24}	Anne Gruber	Biologie
t_{25}	Kieser Klaus	Chemie

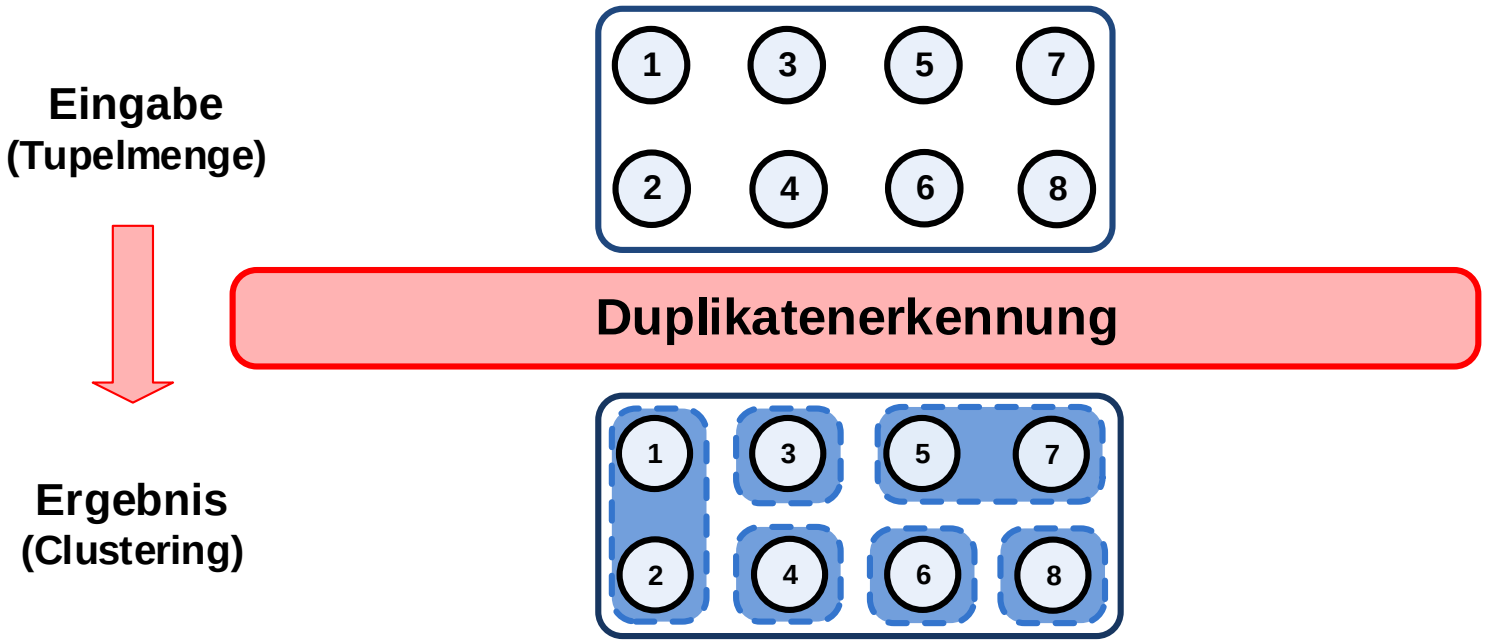
Duplikaterkennung - Synonyme

Auch bekannt als:

- Duplicate Elimination
- Merge-Purge Problem
- Entity Matching
- Entity Resolution
- Record Linkage
- Data Matching
- Deduplication
- Data Linkage
- Object Matching
- Object Reconciliation
- Object Identification
- Reference Reconciliation

Duplikaterkennung - Definition

- Basierend auf paarweisen Tupelvergleichen
- Zwischenergebnisse: Klasse der Matches (M), Klasse der Unmatches (U)
- Endergebnis: Partitionierung (Clustering) der Tupelmenge



Duplikaterkennung - Ziele

Hohe Effektivität:

- Möglichst viele echte Duplikate finden
- Möglichst wenig falsche Duplikate als solche deklarieren

Hohe Effizienz:

- Kurze Laufzeit (speziell bei virtueller Integration)
- Ressourcenschonende Berechnung (Speicher, CPU)
- Skalierbarkeit

Duplikaterkennung - Warum so schwer?

- Zielkonflikt 1: Effektivität vs. Effizienz (offline vs. online)

Duplikaterkennung - Warum so schwer?

- Zielkonflikt 1: Effektivität vs. Effizienz (offline vs. online)
- Zielkonflikt 2: Viele Duplikate finden vs. nur richtige Duplikate als solche deklarieren

Duplikaterkennung - Warum so schwer?

- Zielkonflikt 1: Effektivität vs. Effizienz (offline vs. online)
- Zielkonflikt 2: Viele Duplikate finden vs. nur richtige Duplikate als solche deklarieren
- Domänenspezifische Datenbeschaffenheit (z.B. Datentypen, Ähnlichkeitskonzepte)

Duplikaterkennung - Warum so schwer?

- Zielkonflikt 1: Effektivität vs. Effizienz (offline vs. online)
- Zielkonflikt 2: Viele Duplikate finden vs. nur richtige Duplikate als solche deklarieren
- Domänenspezifische Datenbeschaffenheit (z.B. Datentypen, Ähnlichkeitskonzepte)
- Fallspezifische Datenbeschaffenheit
  - Komplexität (einzelne Tabelle vs. viele Tabellen)
  - Größe (hundert vs. millionen Tupel, wenige vs. viele Attribute)
  - Reinheitsgrad (wenige vs. viele Fehler, einfache vs. massive Fehler, einheitliche vs. uneinheitliche Darstellung)

Duplikaterkennung - Warum so schwer?

- Zielkonflikt 1: Effektivität vs. Effizienz (offline vs. online)
- Zielkonflikt 2: Viele Duplikate finden vs. nur richtige Duplikate als solche deklarieren
- Domänenspezifische Datenbeschaffenheit (z.B. Datentypen, Ähnlichkeitskonzepte)
- Fallspezifische Datenbeschaffenheit
  - Komplexität (einzelne Tabelle vs. viele Tabellen)
  - Größe (hundert vs. millionen Tupel, wenige vs. viele Attribute)
  - Reinheitsgrad (wenige vs. viele Fehler, einfache vs. massive Fehler, einheitliche vs. uneinheitliche Darstellung)
  - Grad der Vollständigkeit (wenige vs. viele Nullwerte)
  - Duplikatmuster (wenige vs. viele Duplikate, kleine Duplikatcluster vs. große Duplikatcluster, Verschiedenartigkeit von Duplikaten bzw. Nichtduplikaten)

Duplikaterkennung - Warum so schwer? (Forts.)

- Semantische Heterogenität (Synonyme, Homonyme, etc.)

Duplikaterkennung - Warum so schwer? (Forts.)

- Semantische Heterogenität (Synonyme, Homonyme, etc.)
- Inhärente Unsicherheit von Duplikaterkennungsprozessen

Duplikaterkennung - Warum so schwer? (Forts.)

- Semantische Heterogenität (Synonyme, Homonyme, etc.)
- Inhärente Unsicherheit von Duplikaterkennungsprozessen
- Anwendungsspezifische Qualitätsanforderungen (welche Fehlerart wiegt schwerer? False Positives oder False Negative)

Duplikaterkennung - Warum so schwer? (Forts.)

- Semantische Heterogenität (Synonyme, Homonyme, etc.)
- Inhärente Unsicherheit von Duplikaterkennungsprozessen
- Anwendungsspezifische Qualitätsanforderungen (welche Fehlerart wiegt schwerer? False Positives oder False Negative)
- Vorhandene Ressourcen (CPU, Speicher, monolitisches vs. verteiltes System)

Duplikaterkennung - Warum so schwer? (Forts.)

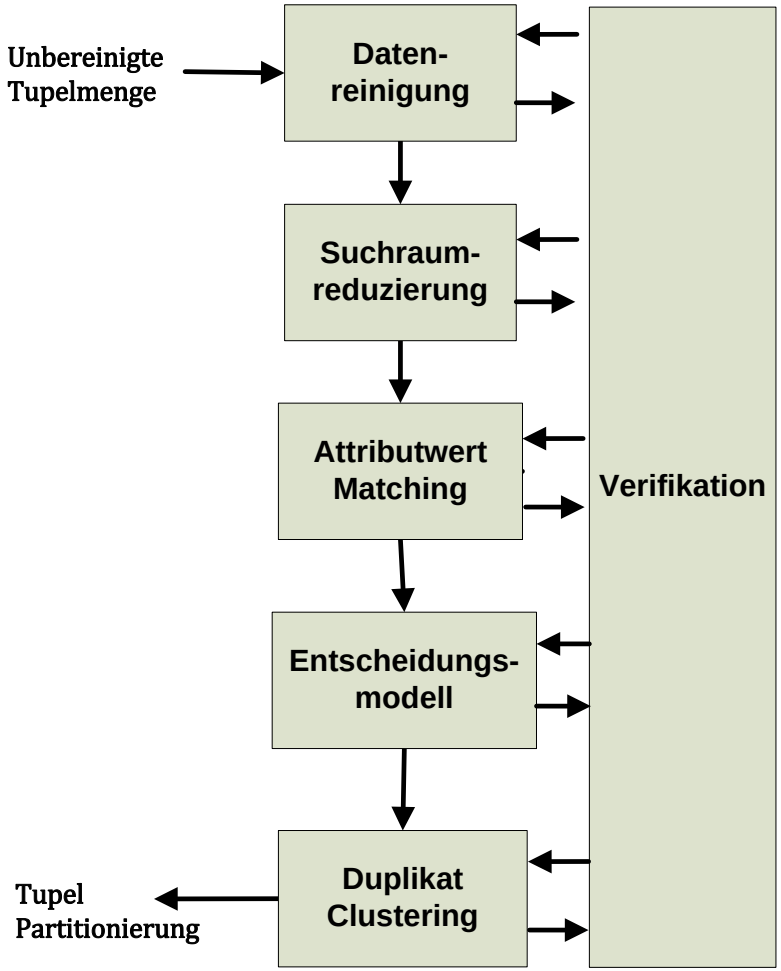
- Semantische Heterogenität (Synonyme, Homonyme, etc.)
- Inhärente Unsicherheit von Duplikaterkennungsprozessen
- Anwendungsspezifische Qualitätsanforderungen (welche Fehlerart wiegt schwerer? False Positives oder False Negative)
- Vorhandene Ressourcen (CPU, Speicher, monolitisches vs. verteiltes System)

Wenn man Datensätze über verschiedene Datenbanken hinweg betrachtet:

- Datenmodell-Heterogenität
- Schematische Heterogenität
  - Relation vs. Attribut, Attribut vs. Wert, Relation vs. Wert
  - Normalisiert vs. denormalisiert

Duplikaterkennung - Prozessablauf

- Datenreinigung (Standardisierung, Entfernung einfacher Fehler)
- Suchraumreduzierung (Effizienzerhöhung)
- Attributwertmatching (Messen von Attributwertähnlichkeiten)
- Entscheidungsmodell (ähnlichkeitsbasierte Entscheidungsfindung)
- Duplikat Clustering (Clustern der paarweisen Entscheidungen)
- Verifikation (Abschätzung der Ergebnisqualität)



Verifikation

Eingabe:

- Gold standard (korrektes Clustering $\mathcal{C}_{\text{gold}}$)
- Duplikat Clustering $\mathcal{C}$, Suchraum, Paarweise Entscheidungen

Ausgabe:

- Qualitätswert $q \in [0, 1]$

Unterscheidung in:

- Paarbasierte Verfahren
- Clusterbasierte Verfahren

Paarbasierte Verifikation

Grundidee:

- Zerlegung des Clusterings in paarweise Duplikatsentscheidungen

$\Rightarrow$ Klassifikation in Matches (M) und Unmatches (U)

- Qualität als die Güte einer binären Klassifikation
  - **True Positives (TP):** zu Recht deklarierten Matches
  - **True Negatives (TN):** zu Recht deklarierten Unmatches
  - **False Positives (FP):** zu Unrecht deklarierten Matches
  - **False Negatives (FN):** zu Unrecht deklarierten Unmatches

Paarbasierte Verifikation - Qualitätsmaße

- **Recall:** Anteil an gefundenen Duplikaten

$$Recall = \frac{|TP|}{|TP| + |FN|}$$

- **Precision:** Anteil an zu Recht deklarierten Matches

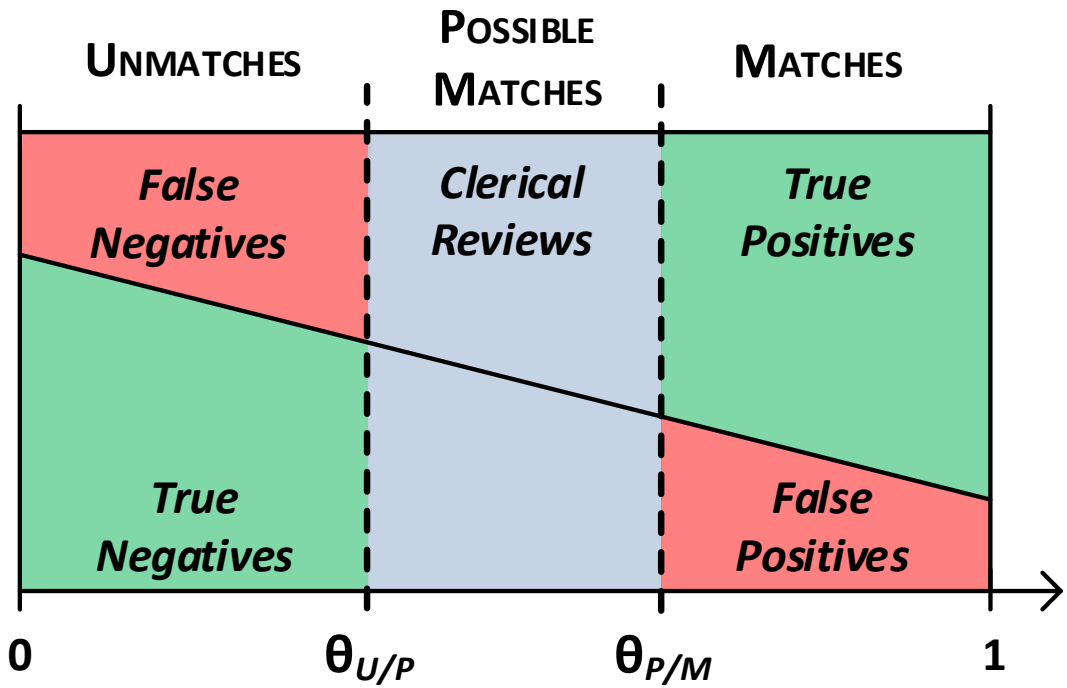
$$Precision = \frac{|TP|}{|TP| + |FP|}$$

- **F_1-score:** Harmonisches Mittel von Recall und Precision

$$F_1\text{-score} = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

Trade-Off: Recall vs. Precision

- Hoher Schwellwert $\Rightarrow$ hohe Precision, kleiner Recall
- Kleiner Schwellwert $\Rightarrow$ kleine Precision, hoher Recall



Clusterbasierte Verifikation [MWGM10]

Grundidee:

- Gold standard und Duplikaterkennungsergebnis sind Clusterings

$\Rightarrow$ Qualität als Ähnlichkeit zwischen zwei Clusterings

Qualitätsmaße:

- **Closest Cluster Recall:**

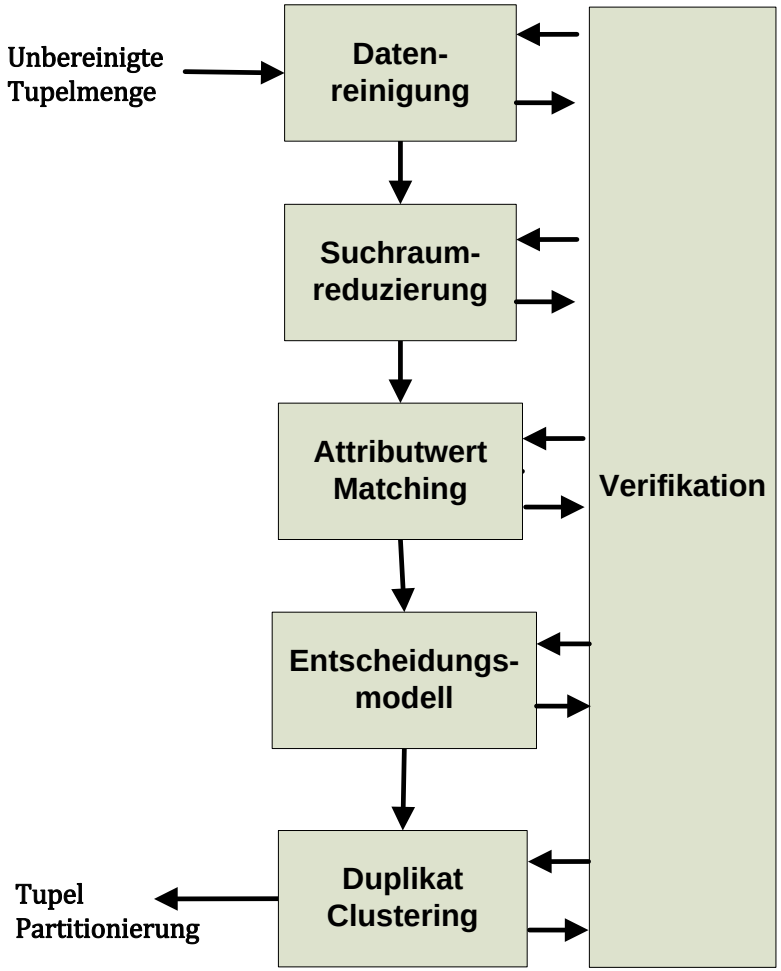
$$\text{ccRec}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \frac{\sum_{C_q \in \mathcal{C}_{\text{gold}}} \mathbf{max}_{C_p \in \mathcal{C}} sim(C_p, C_q)}{|\mathcal{C}_{\text{gold}}|}$$

- **Closest Cluster Precision:**

$$\text{ccPrec}(\mathcal{C}, \mathcal{C}_{\text{gold}}) = \frac{\sum_{C_p \in \mathcal{C}} \mathbf{max}_{C_q \in \mathcal{C}_{\text{gold}}} sim(C_p, C_q)}{|\mathcal{C}|}$$

Duplikaterkennung - Prozessablauf

- Datenreinigung (Standardisierung, Entfernung einfacher Fehler)
- Suchraumreduzierung (Effizienzerhöhung)
- Attributwertmatching (Messen von Attributwertähnlichkeiten)
- Entscheidungsmodell (ähnlichkeitsbasierte Entscheidungsfindung)
- Duplikat Clustering (Clustern der paarweisen Entscheidungen)
- Verifikation (Abschätzung der Ergebnisqualität)



Suchraumreduzierung - Motivation

- Iterative Verfahren haben quadratische Komplexität

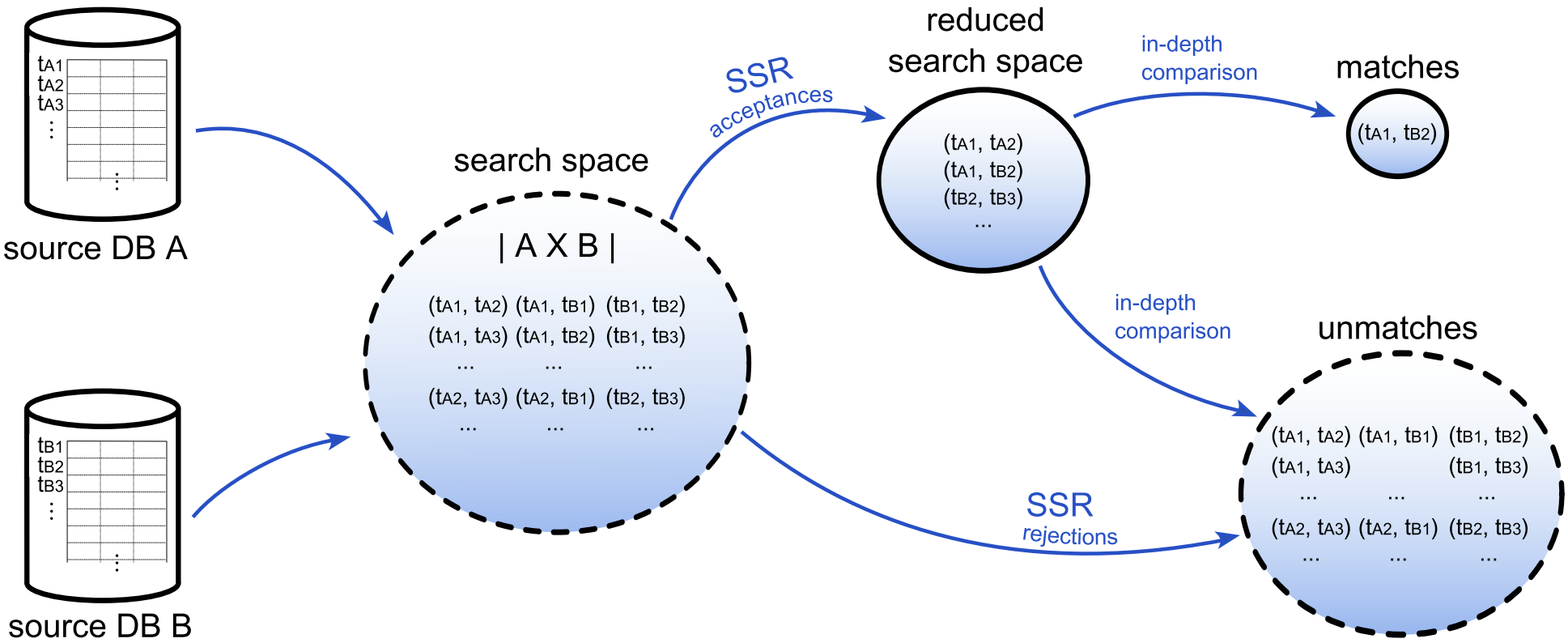
  Gegeben: n Tupel $\Rightarrow \frac{n \times (n-1)}{2}$ paarweise Vergleiche

$\Rightarrow$ Paarweiser Vergleich aller Tupel zu ineffizient

- Herausfiltern klarer *Unmatches* mit wenig Aufwand
- 'Teure' Vergleiche nur für potentielle Duplikatpaare

- *False Negatives* (geminderte Effektivität) schlimmer als *False Positives* (geminderte Effizienz)

Suchraumreduzierung - Grundidee



Schlüsselbasierte Suchraumreduzierung

- Die meisten Reduzierungsverfahren basieren auf der Nutzung von Schlüsseln
- Eine Schlüsseldefinition ist eine Funktion die angibt wie ein Stringwert aus einem Tupel extrahiert wird

Schlüsselbasierte Suchraumreduzierung

- Die meisten Reduzierungsverfahren basieren auf der Nutzung von Schlüsseln
- Eine Schlüsseldefinition ist eine Funktion die angibt wie ein Stringwert aus einem Tupel extrahiert wird

  Beispiel:

  Konkateniere die ersten drei Buchstaben des Vornamens mit den ersten zwei Buchstaben des Nachnames

Schlüsselbasierte Suchraumreduzierung

- Die meisten Reduzierungsverfahren basieren auf der Nutzung von Schlüsseln
- Eine Schlüsseldefinition ist eine Funktion die angibt wie ein Stringwert aus einem Tupel extrahiert wird

  Beispiel:

  Konkateniere die ersten drei Buchstaben des Vornamens mit den ersten zwei Buchstaben des Nachnames
- Mit Hilfe der Schlüsseldefinition wird von jedem Tupel ein Schlüssel extrahiert
- Die Schlüssel werden dann verwendet um zu bestimmen welche Tupelpaare im Suchraum verbleiben

Standard Blocking

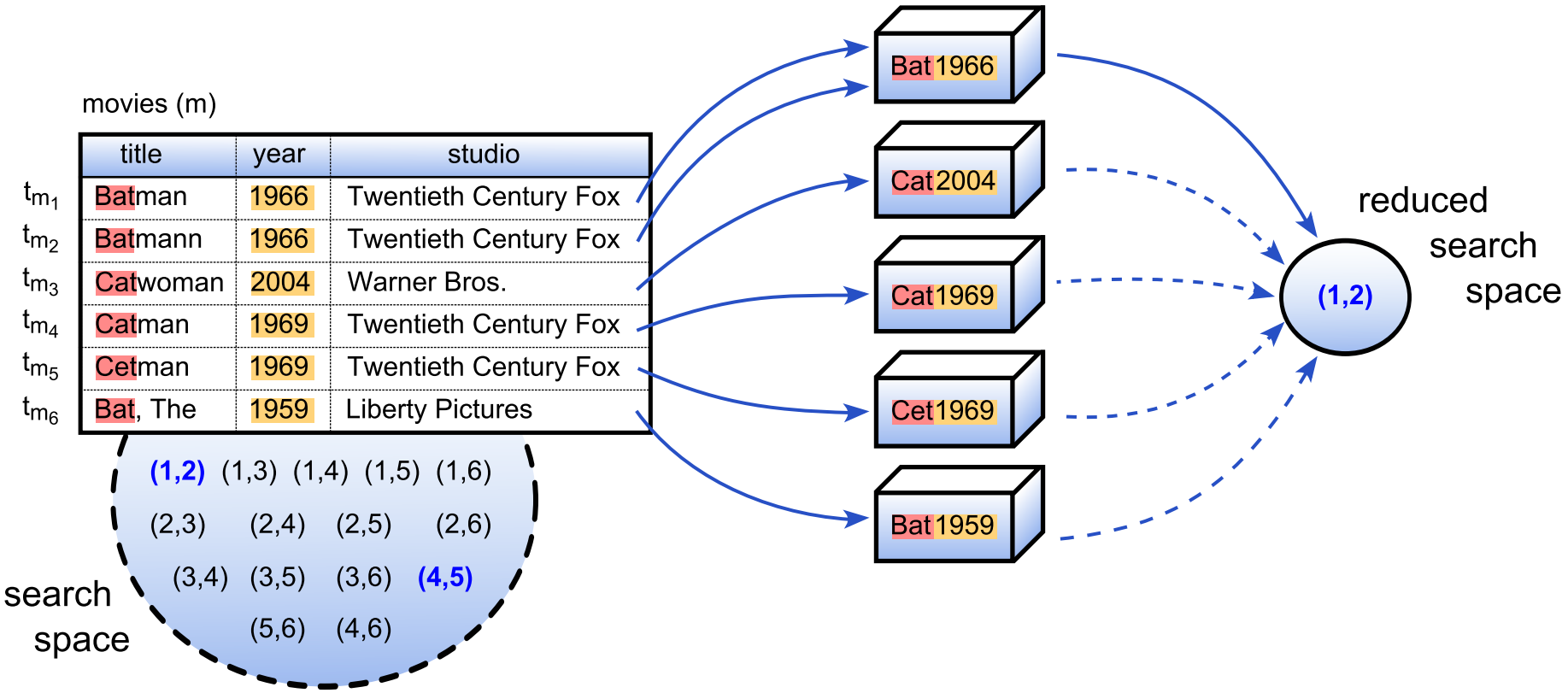
Eingabe:

- Schlüsseldefinition
- [Optional] Codierungsfunktion

Vorgehensweise:

- Bildung eines Schlüssels für jedes Tupel
- [Optional] Codierung des Schlüssels (z.B. Soundex Code)
- Blockbildung: Ein Block pro Schlüssel (Hashing)
- Zwei Tupel im selben Block bilden ein Kandidatenpaar

Standard Blocking (Beispiel)



Standard Blocking

Stärken:

- Einfach zu Implementieren
- Schnell ('billig') auszuführen

Schwächen:

- Größte Block bestimmt Größe des Suchraumes

$\Rightarrow$ Verfahren ineffektiv sobald ein Block groß ist

- Entfernen großer Blöcke aber nicht möglich da jedes Tupel nur einmal vorkommt
- Paarvergleich nur bei Schlüsselgleichheit

$\Rightarrow$ Langer Schlüssel: Kleine Blöcke aber viele *False Negatives*

$\Rightarrow$ Kurzer Schlüssel: Wenige *False Negatives* aber große Blöcke

Sorted Neighborhood Method [HS95]

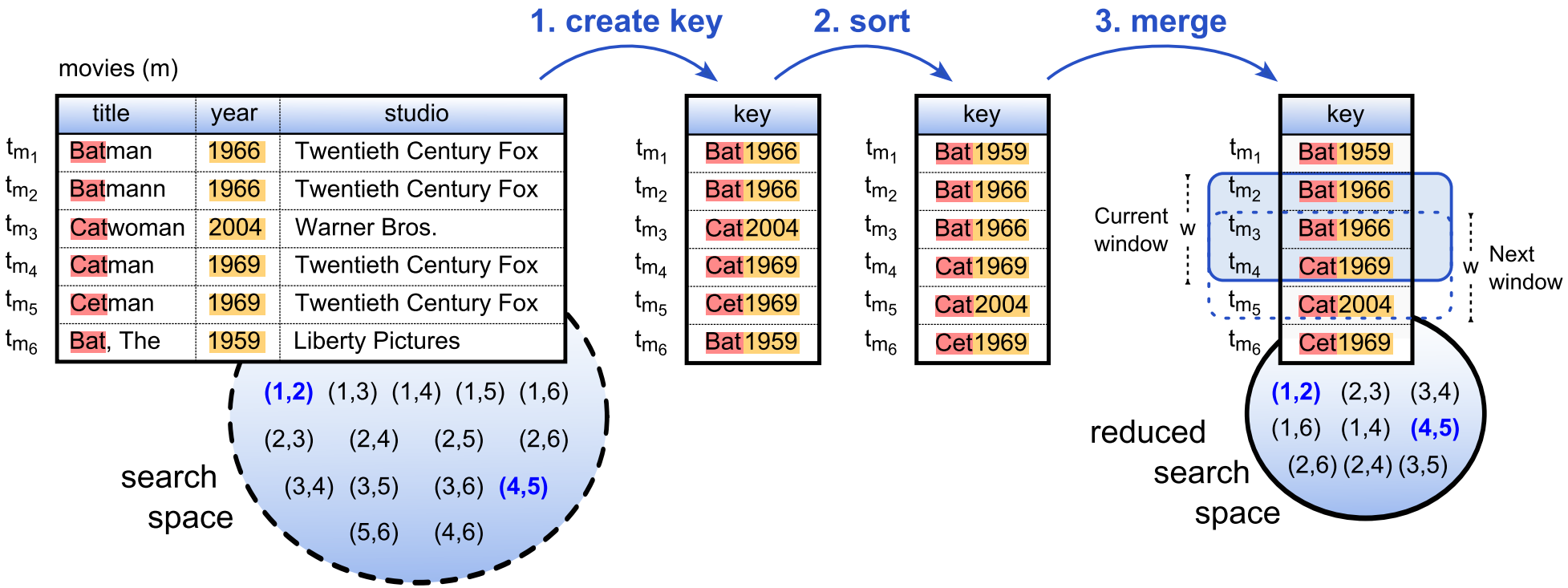
Eingabe:

- Schlüsseldefinition
- Fenstergröße w

Vorgehensweise:

- Bildung eines Schlüssels für jedes Tupel
- Sortieren aller Tupel nach Schlüssel
- Ein 'Fenster' der Größe w läuft über die sortierte Tupel-Liste
- Zwei Tupel im selben Fenster bilden ein Kandidatenpaar

Sorted Neighborhood Method (Beispiel)



Sorted Neighborhood Method

Stärken:

- Überlappende Blöcke (Schlüssel müssen nicht mehr gleich sein)
- Genaue Vorhersage über resultierende Suchraumgröße möglich

Schwächen:

- Fenstergröße vs. Häufigkeit von Schlüsselwerten
- Tupel mit sehr ähnlichen Schlüsseln werden evtl. nicht verglichen (Lsg.: variable Fenstergröße)

Suffix-Array Based Blocking [dVKCC09]

Eingabe:

- Schlüsseldefinition
- Minimale Suffixlänge l_s
- Maximale Blockgröße l_b

Vorgehensweise:

- Bildung eines Schlüssels für jedes Tupel
- Bildung der Suffixe der Länge $l \geq l_s$ von allen Schlüsseln
- Blockbildung: Ein Block pro Suffix
  ($\Rightarrow$ mehrere Schlüssel/Blöcke pro Tupel)
- Entfernung aller Blöcke mit mehr als l_b Einträgen
- Zwei Tupel im selben Block bilden ein Kandidatenpaar

Suffix-Array Based Blocking (Beispiel)

Id	**BKV**	**Suffixe**
R1	Catherine	catherine, atherine, therine, herine, erine, rine
R2	Katherina	katherina, atherina, therina, herina, erina, rina
R3	Catherina	catherina, atherina, therina, herina, erina, rina
R4	Catrina	catrina, trina, rina
R5	Katrina	katrina, trina, rina

Suffix-Array Based Blocking (Beispiel)

Id	BKV	Suffixe
R1	Catherine	catherine, atherine, therine, herine, erine, rine
R2	Katherina	katherina, atherina, therina, herina, erina, rina
R3	Catherina	catherina, atherina, therina, herina, erina, rina
R4	Catrina	catrina, trina, rina
R5	Katrina	katrina, trina, rina

Suffix	Identifier
atherina	R2,R3
atherine	R1
atrina	R4,R5
catherina	R3
catherine	R1
catrina	R4
erina	R2,R3
erine	R1
herina	R2,R3

Suffix	Identifier
herine	R1
katherina	R2
katrina	R5
~~rina~~	~~R2,R3,R4,R5~~
rine	R1
therina	R2,R3
therine	R1
trina	R4,R5

Suffix-Array Based Blocking

Stärken:

- Wie Standard Blocking, aber löst das Problem der Schlüssellänge
  - benutzt pro Tupel den kürzesten Schlüssel der nicht zu einem großen Block führt

Schwächen:

- Fehler am Ende der Schlüssel werden nicht kompensiert

$\Rightarrow$ Verfahren einmal für Suffix und einmal für Präfix (der Präfix eines Strings entspricht dem Suffix des gespiegelten Strings)

$\Rightarrow$ Mergen von Blöcken ähnlicher Suffixe

Schlüsselbasierte Suchraumreduzierung

Stärken:

- Effiziente Berechnung
- Leicht an unterschiedliche Domänen anpassbar

Schwächen:

- Große Abhängigkeit von der Wahl des Schlüssels

$\Rightarrow$ Multi-pass Verfahren

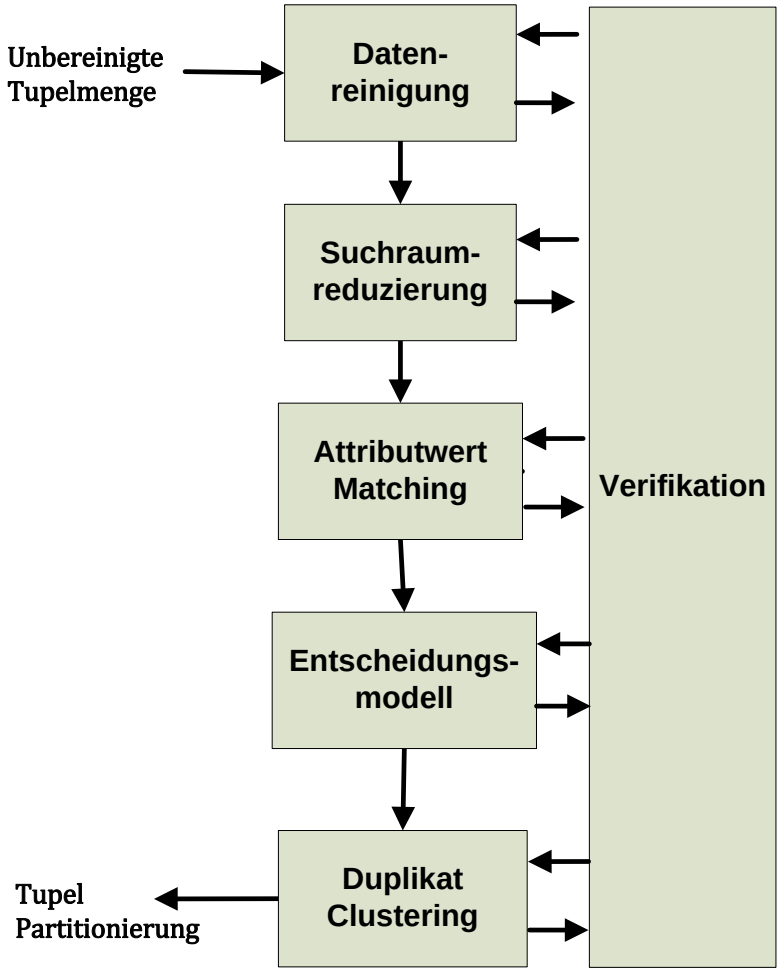
- Mehrere Schlüsseldefinitionen
- Ein Durchlauf pro Schlüsseldefinition
- Vereinigung der produzierten Suchräume
  (Bsp. Füge dem Suchraum alle Paare hinzu die in mindestens zwei Durchläufen als Kandidat deklariert wurden)

weitere Suchraumreduzierungsverfahren

- Q-gram Indexing
- Canopy Clustering
- K-way Sorting
- Similarity-Aware Inverted Indexing
- String-Map basiertes Blocking
- LSH basiertes Blocking
- Sorted Blocks
- Priority Queue
- TI-similarity basiertes Blocking
- Spectral Neighborhood Blocking
- Blocking mit MFIBlocks
- Iteratives Blocking
- Fuzzy Blocking
- Paralleles Blocking mit Map-Reduce

Duplikaterkennung - Prozessablauf

- Datenreinigung (Standardisierung, Entfernung einfacher Fehler)
- Suchraumreduzierung (Effizienzerhöhung)
- Attributwert Matching (Messen von Attributwertähnlichkeiten)
- Entscheidungsmodell (ähnlichkeitsbasierte Entscheidungsfindung)
- Duplikat Clustering (Clustern der paarweisen Entscheidungen)
- Verifikation (Abschätzung der Ergebnisqualität)



Attributwert Matching

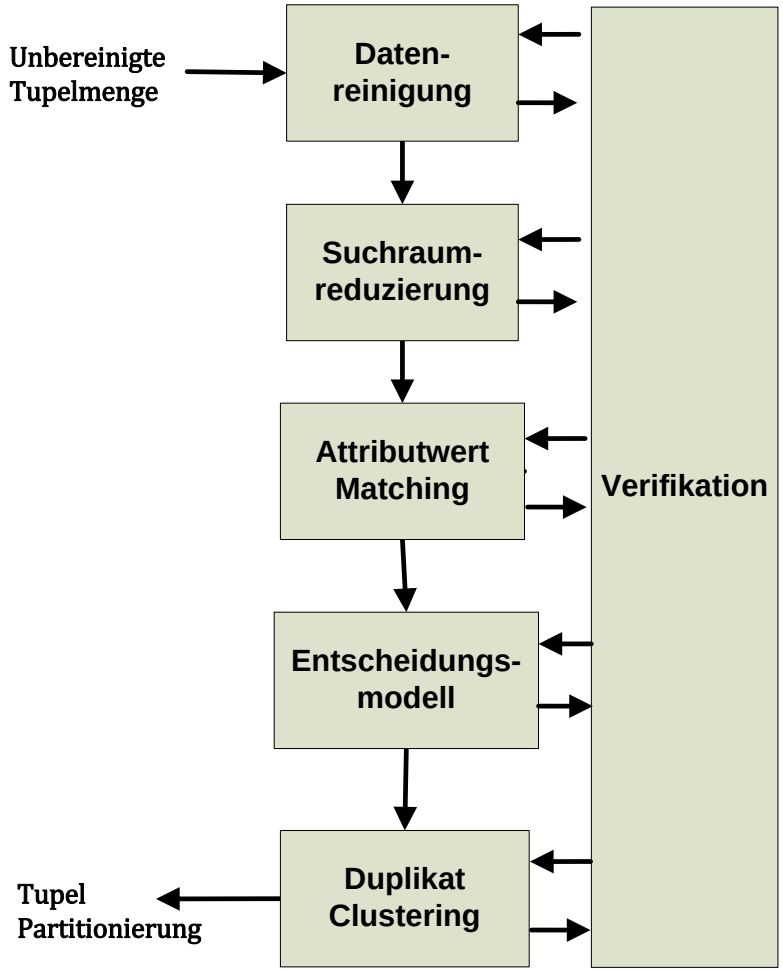
- Für jedes Kandidatenpaar wird ein Vergleichsvektor berechnet

$$\vec{c} = \langle c_1, \ldots, c_n \rangle, \; c_i \in [0, 1]$$

- Der Vektor enthält normierte Ähnlichkeiten zwischen Attributwerten
- Ähnlichkeitsberechnung durch String Matching (z.B. Levenshtein, Monge-Elkan, Kosinus, etc.)
- **Üblich:** Nur Vergleiche von Werten gleicher Attribute
  $\Rightarrow$ Eine Vektorposition pro Attribut
- Attributübergreifende Vergleiche können nützlich sein, wenn:
  - Attributwerte vertauscht sein können (z.B. Vor- und Nachname)
  - der erste Wert in dem zweiten codiert sein kann (z.B. Name in E-Mail)

Duplikaterkennung - Prozessablauf

- Datenreinigung (Standardisierung, Entfernung einfacher Fehler)
- Suchraumreduzierung (Effizienzerhöhung)
- Attributwertmatching (Messen von Attributwertähnlichkeiten)
- Entscheidungsmodell (ähnlichkeitsbasierte Entscheidungsfindung)
- Duplikat Clustering (Clustern der paarweisen Entscheidungen)
- Verifikation (Abschätzung der Ergebnisqualität)



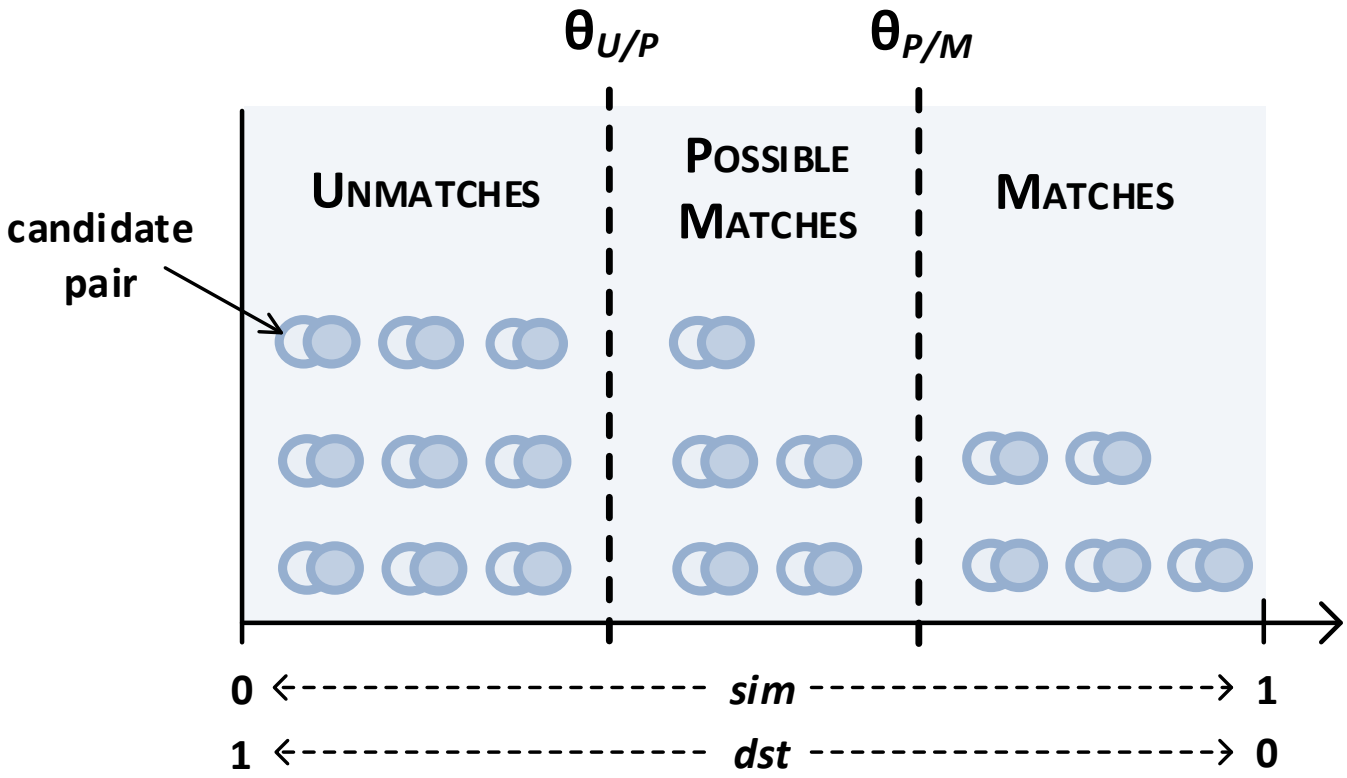
Entscheidungsmodell

Eingabe: Ein Tupelpaar $\{t_i, t_j\}$, Vergleichsvektor $\vec{c}(t_i, t_j)$
Ausgabe: Entscheidung ob t_i und t_j Duplikate sind (samt Ähnlichkeit)

Grundidee:

- Berechnung einer Ähnlichkeit/Distanz zwischen beiden Tupeln
- Nutzen eines Schwellwertes um Tupelpaar als Match oder Unmatch zu klassifizieren
- Häufig Verwendung einer dritten Klasse: Possible Matches
- Possible Matches müssen später von einem Domänexperten manuell begutachtet werden
  - Geht natürlich nur offline (d.h. materialisierte Integration)
  - Alternativ: Crowdsourcing, probabilistische Modellierung

Entscheidungsmodell



Distanzbasierte Ansätze [EIV07]

Ähnlichkeitsberechnung:

- Jedes Tupel als langen String betrachten und Stringähnlichkeitsmaß (z.B. Levenshtein) verwenden (in diesem Fall wird der Vergleichsvektor nicht genutzt)
- Gewichteter Durchschnitt aller Attributähnlichkeiten
- Maximum Weighted Matching zwischen den Attributwerten beider Tupel (wenn attributübergreifende Ähnlichkeiten existieren)
- Distanz im Vektorraum

Distanz im Vektorraum

- Vergleichsvektor $\vec{c}(t_i, t_j) = \langle c_1, \ldots, c_n \rangle$ wird als Punkt P in einem n dimensionalen Vektorraum betrachtet
- Der Punkt $\mathbf{0} = (0, 0, \ldots, 0)$ steht für sichere Nichtduplikate
- Der Punkt $\mathbf{1} = (1, 1, \ldots, 1)$ steht für sichere Duplikate
- Berechnung der Ähnlichkeit zwischen t_i und t_j mit Hilfe eines Distanzmaßes *dst*

$$sim = 1 - \frac{dst(P, \mathbf{1})}{dst(\mathbf{0}, \mathbf{1})} = \frac{dst(\mathbf{0}, P)}{dst(\mathbf{0}, \mathbf{1})}$$

- Wahl des Distanzmaßes *dst*:
  - Euklidische Distanz
  - Manhattan Distanz
  - etc.

Regelbasierte Ansätze [WM89]

Grundidee:

- Domänenexperte definiert Regeln, die bestimmen wann zwei Tupel Duplikate oder Nichtduplikate sind
- Formale Beschreibung:

$$\textit{Prämisse} \Rightarrow \textit{Schlussfolgerung}$$

- Die Prämisse ist gewöhnlich ein Prädikat in konjunktiver Normalform und ist entweder über den Ähnlichkeitsvektor oder die Attributwerte selber definiert:

$$\textit{Prämisse} = (term_{1,1} \vee term_{1,2} \vee \ldots) \wedge \ldots \wedge (term_{n,1} \vee term_{n,2} \vee \ldots)$$

- Die Schlussfolgerung ist Match oder Unmatch

Regelbasierte Ansätze

Einfacher Ansatz:

- Menge an positiven Regeln (Schlussfolgerung ist Match)
- Wenn eine Regel feuert wird das Tupelpaar als Match klassifiziert
- Wenn keine Regel feuert wird das Tupelpaar als Unmatch klassifiziert
- Reihenfolge in der die Regeln geprüft werden ist irrelevant

Profilbasierter Ansatz:

- Positive Regeln (Schlussfolgerung ist Match) und negative Regeln (Schlussfolgerung ist Unmatch)
- Regeln werden der Reihe nach geprüft bis eine feuert
- Reihenfolge in der die Regeln geprüft werden ist relevant

Regelbasierte Ansätze (Beispiel)

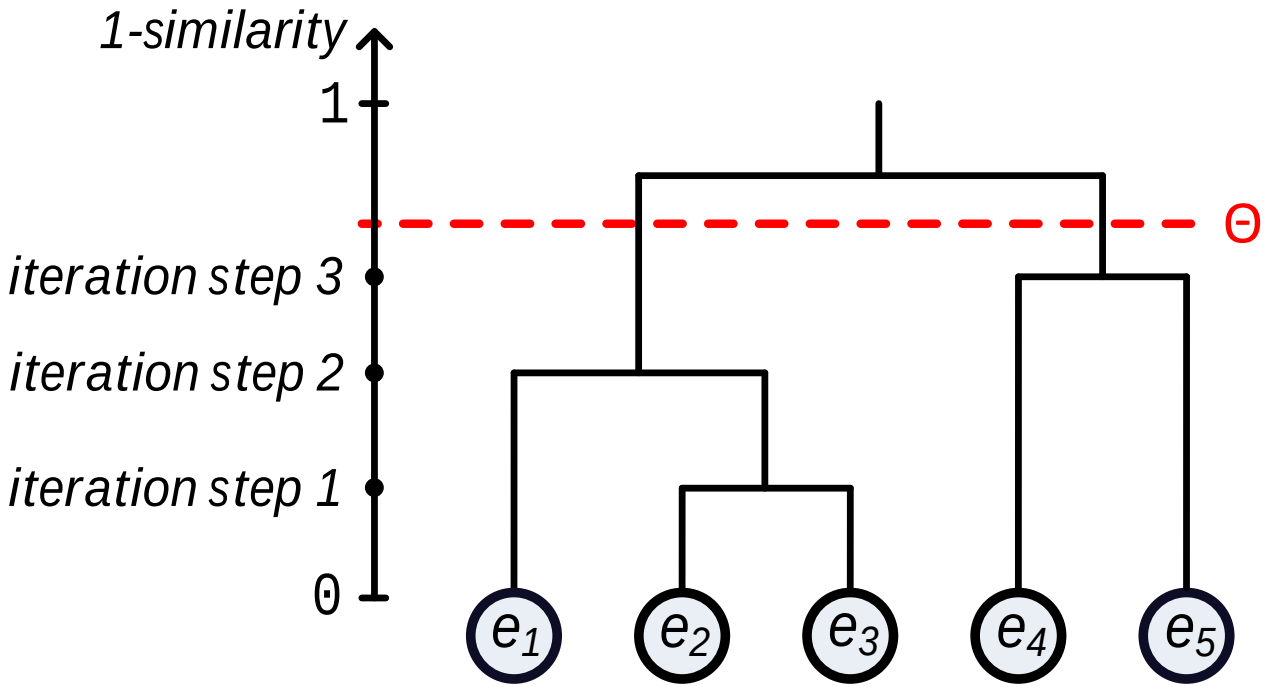
$$
\begin{aligned}
\mathbf{R}_1: &\quad (\vec{c}(\text{fname}, \text{fname}) > 0.6) \wedge (\vec{c}(\text{lname}, \text{lname}) > 0.8) \wedge (\vec{c}(\text{DoB}, \text{DoB}) = 1.0) &&\Rightarrow \textit{Match}\\
\mathbf{R}_2: &\quad (\vec{c}(\text{fname}, \text{lname}) > 0.6) \wedge (\vec{c}(\text{lname}, \text{fname}) > 0.8) \wedge (\vec{c}(\text{DoB}, \text{DoB}) = 1.0) &&\Rightarrow \textit{Match}\\
\mathbf{R}_3: &\quad (\vec{c}(\text{lname}, \text{fname}) > 0.6) \wedge (\vec{c}(\text{fname}, \text{lname}) > 0.8) \wedge (\vec{c}(\text{DoB}, \text{DoB}) = 1.0) &&\Rightarrow \textit{Match}\\
\mathbf{R}_4: &\quad (\vec{c}(\text{lname}, \text{lname}) \leq 0.5) \wedge (\vec{c}(\text{country}, \text{country}) \neq 1.0) &&\Rightarrow \textit{Unmatch}\\
\mathbf{R}_5: &\quad (\vec{c}(\text{lname}, \text{email}) > 0.5) \wedge (\vec{c}(\text{fname}, \text{email}) > 0.5) &&\Rightarrow \textit{Match}\\
\mathbf{R}_{\text{default}}: &\quad \text{True} &&\Rightarrow \textit{Unmatch}
\end{aligned}
$$

Clusterbasierte Ansätze [DHI12]

Grundidee:

- Verwendung von hierarchischen Clusteringverfahren (gegeben Ähnlichkeitsmaß und Schwellwert θ)
- Initialisiere alle Tupel als einelementige Cluster
- Bis die größte gefundene Ähnlichkeit kleiner als θ ist, tue
  - Berechne die Ähnlichkeit zwischen allen Clustern
  - Ist die größte gefundene Ähnlichkeit größer als θ
    $\Rightarrow$ vereine die beiden betreffenden Cluster

Clusterbasierte Ansätze



Clusterbasierte Ansätze

Wie definiert man die Ähnlichkeit zwischen zwei Clustern?

- **Average Link:** durchschnittliche Ähnlichkeit der Cluster-Tupel
- **Complete Link:** maximale Ähnlichkeit der Cluster-Tupel
- **Single Link:** minimale Ähnlichkeit der Cluster-Tupel
- **Canonical Entity:** bilde ein kanonisches Tupel per Cluster und nehme die Ähnlichkeit der kanonischen Tupel as Clusterähnlichkeit

Statistische Ansätze [EIV07, Tal11]

Grundidee:

- Berechne die Wahrscheinlichkeiten, dass der Vergleichsvektor typisch für ein Match (kurz M) bzw. Unmatch (kurz U) ist
- Klassifiziere das Tupelpaar als Match wenn die Wahrscheinlichkeit dafür größer ist als für ein Unmatch

$$classify(t_i, t_j) = \begin{cases} M, & \text{if } Pr(M \mid \vec{c}) \geq Pr(U \mid \vec{c}) \\ U, & \text{sonst.} \end{cases}$$

Statistische Ansätze [EIV07, Tal11]

Grundidee:

- Berechne die Wahrscheinlichkeiten, dass der Vergleichsvektor typisch für ein Match (kurz M) bzw. Unmatch (kurz U) ist
- Klassifiziere das Tupelpaar als Match wenn die Wahrscheinlichkeit dafür größer ist als für ein Unmatch

$$classify(t_i, t_j) = \begin{cases} M, & \text{if } Pr(M \mid \vec{c}) \geq Pr(U \mid \vec{c}) \\ U, & \text{sonst.} \end{cases}$$

- Umformung gemäß Satz des Bayes (Logarithmus hält die Wertebereiche klein):

$$classify(t_i, t_j) = \begin{cases} M, & \text{if } \log_2\left(\frac{Pr(\vec{c}|M)}{Pr(\vec{c}|U)}\right) \geq \log_2\left(\frac{Pr(U)}{Pr(M)}\right), \\ U, & \text{sonst.} \end{cases}$$

Statistische Ansätze

Vorraussetzungen:

- Schätzungen oder statistische Erhebungen für die einzelnen Wahrscheinlichkeiten $Pr(\vec{c} \mid M)$
- Da es unendliche viele Vergleichsvektoren gibt, lassen sich die Wahrscheinlichkeiten nicht statistisch bestimmen

$\Rightarrow$ Benutzung eines Matching Pattern

Statistische Ansätze

Vorraussetzungen:

- Schätzungen oder statistische Erhebungen für die einzelnen Wahrscheinlichkeiten $Pr(\vec{c} \mid M)$
- Da es unendliche viele Vergleichsvektoren gibt, lassen sich die Wahrscheinlichkeiten nicht statistisch bestimmen

$\Rightarrow$ Benutzung eines Matching Pattern

Vorgehensweise:

- Transformiere den Vergleichsvektor in ein Matching Pattern
- Ein Matching Pattern ist ein n-dimensionaler Vektor $\vec{p}$ auf einem endlichen Wertebereich

$$\vec{p} = \langle p_1, \ldots, p_n \rangle, \; p_i \in \{1, 2, \ldots, k\}$$

Statistische Ansätze

- Berechne die Wahrscheinlichkeiten, dass $\vec{p}$ typisch für einen Match bzw. Unmatch ist
- Annahme von Unabhängigkeit zwischen verschiedenen Attributpaaren (Naive Bayes)

$$Pr(\vec{p} \mid M) = \prod_{i=1}^{|\vec{p}|} Pr(\vec{p}[i] \mid M)$$

$$Pr(\vec{p} \mid U) = \prod_{i=1}^{|\vec{p}|} Pr(\vec{p}[i] \mid U)$$

Statistische Ansätze

- Berechne die Wahrscheinlichkeiten, dass $\vec{p}$ typisch für einen Match bzw. Unmatch ist
- Annahme von Unabhängigkeit zwischen verschiedenen Attributpaaren (Naive Bayes)

$$Pr(\vec{p} \mid M) = \prod_{i=1}^{|\vec{p}|} Pr(\vec{p}[i] \mid M)$$

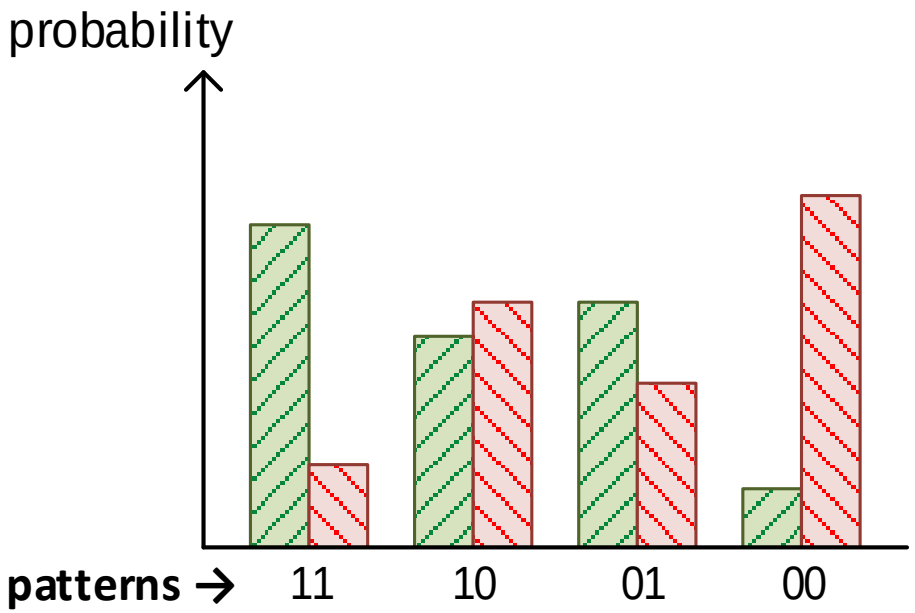
$$Pr(\vec{p} \mid U) = \prod_{i=1}^{|\vec{p}|} Pr(\vec{p}[i] \mid U)$$

Vorraussetzungen:

- Schätzungen oder statistische Erhebungen für die einzelnen Wahrscheinlichkeiten $Pr(\vec{p}[i] \mid M)$ und $Pr(\vec{p}[i] \mid U)$

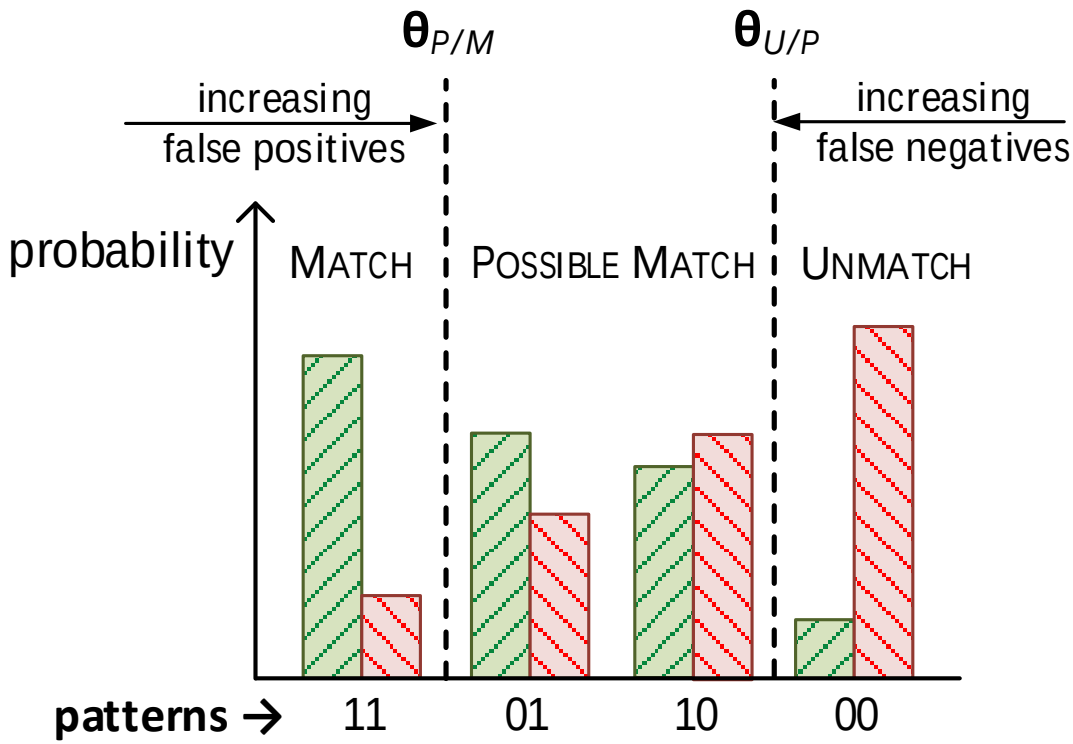
Statistische Ansätze (Beispiel)

Binäres Pattern:



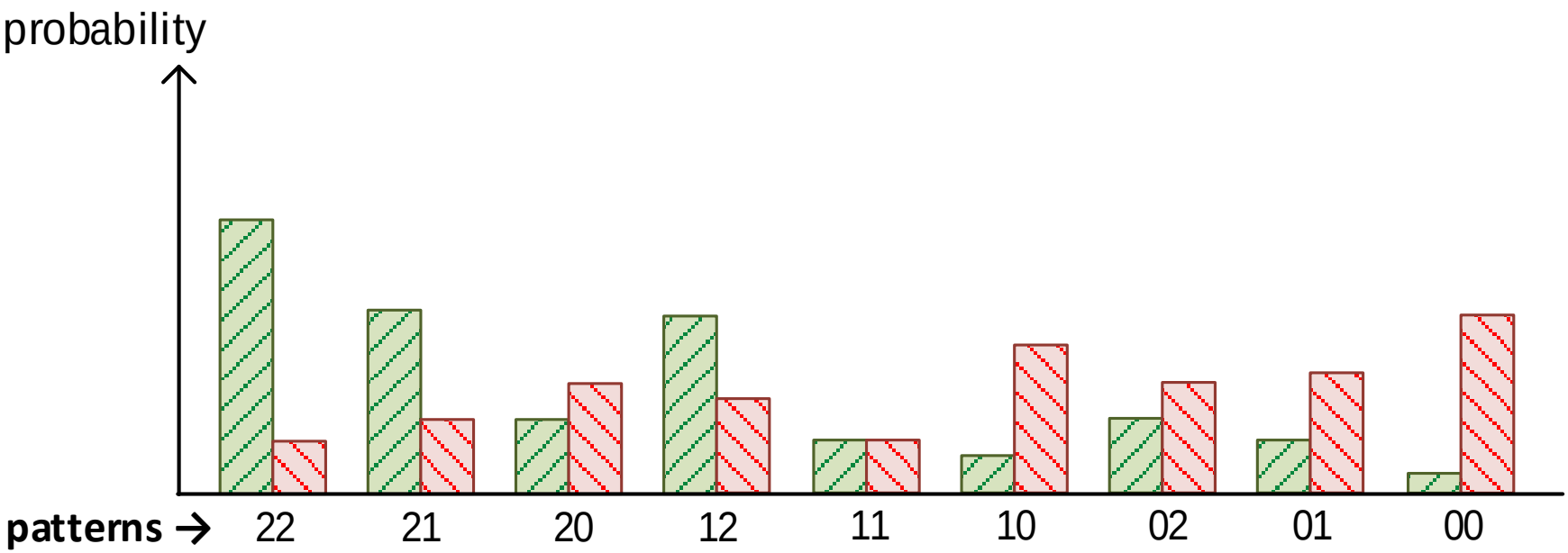
Statistische Ansätze (Beispiel)

Binäres Pattern:



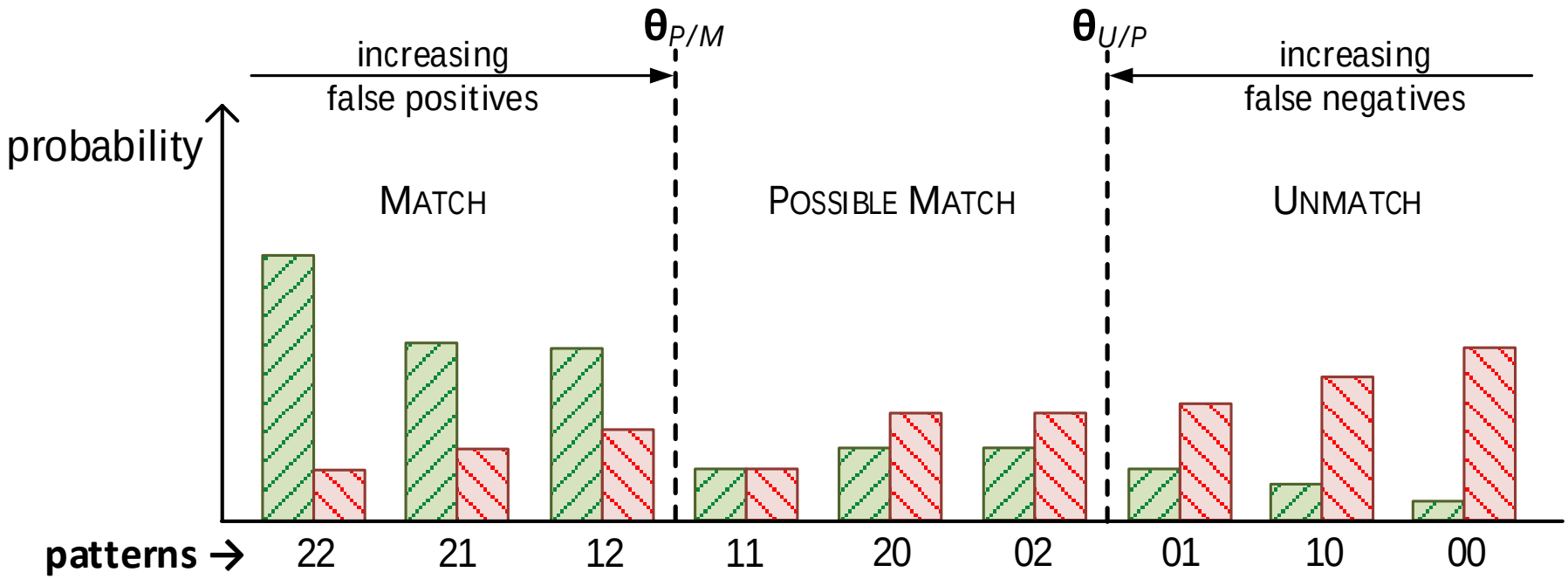
Statistische Ansätze (Beispiel)

Ternäres Pattern:



Statistische Ansätze (Beispiel)

Ternäres Pattern:



Statistische Ansätze (Beispiel)

Vergleich: Binär vs Ternär

Tupelpaar	Vergleichsvektor	Binäres Pattern	Ternäres Pattern
$\{t_1, t_2\}$	$\langle 0.4, 1.0 \rangle$	$\langle 0, 1 \rangle$	$\langle 1, 2 \rangle$
$\{t_1, t_3\}$	$\langle 0.7, 0.7 \rangle$	$\langle 1, 1 \rangle$	$\langle 2, 1 \rangle$
$\{t_1, t_4\}$	$\langle 0.9, 1.0 \rangle$	$\langle 1, 1 \rangle$	$\langle 2, 2 \rangle$
$\{t_1, t_5\}$	$\langle 0.1, 0.6 \rangle$	$\langle 0, 1 \rangle$	$\langle 0, 1 \rangle$
$\{t_1, t_6\}$	$\langle 0.6, 0.6 \rangle$	$\langle 0, 1 \rangle$	$\langle 1, 1 \rangle$

$\Rightarrow$ Ternäres Pattern deutlich diskriminierender als binäres

Lernbasierte Ansätze

Unüberwachte Lernverfahren:

- keine Trainingsdaten erforderlich

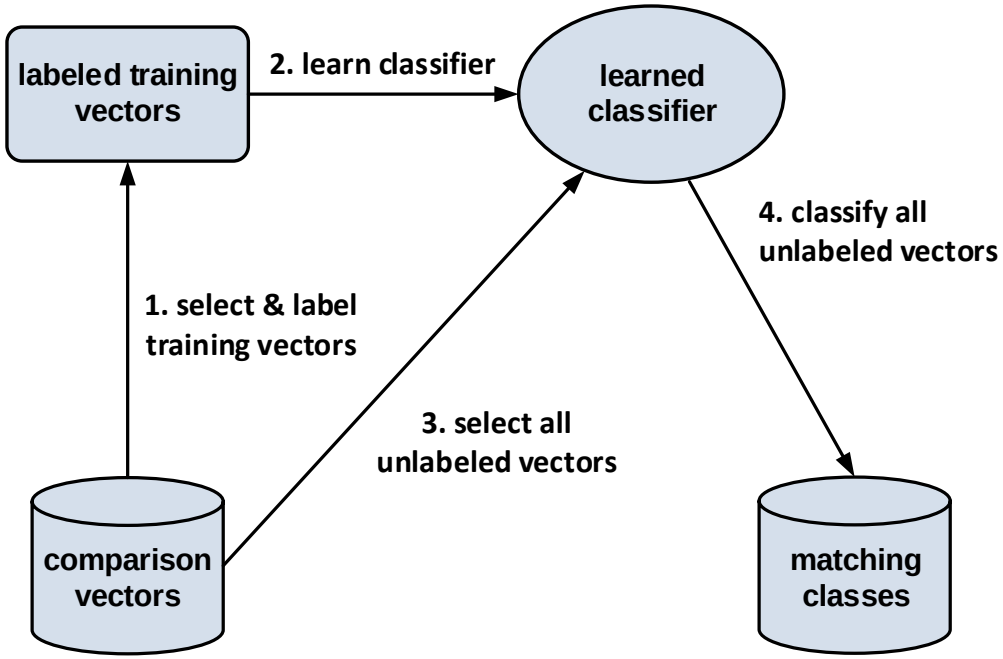
Lernbasierte Ansätze

Unüberwachte Lernverfahren:

- keine Trainingsdaten erforderlich

Überwachte Lernverfahren:

- Trainingsdaten erforderlich
- Trainingsdaten sind die Vergleichsvektoren von Tupelpaaren von denen wir wissen, ob sie Duplikate sind oder nicht



Unüberwachte Lernverfahren [EIV07]

Constrained Clustering:

- Clustern der Vergleichsvektoren in zwei Cluster (z.B. 2-Means)
- Markiere das Cluster das sich näher an Punkt $\mathbf{1} = (1, 1, \ldots, 1)$ befindet als Menge der Matches (und das andere als Menge der Unmatches)
- Vorteil: voll automatisch
- Nachteil: oft keine klare Grenze zwischen den Clustern

Unüberwachte Lernverfahren [EIV07]

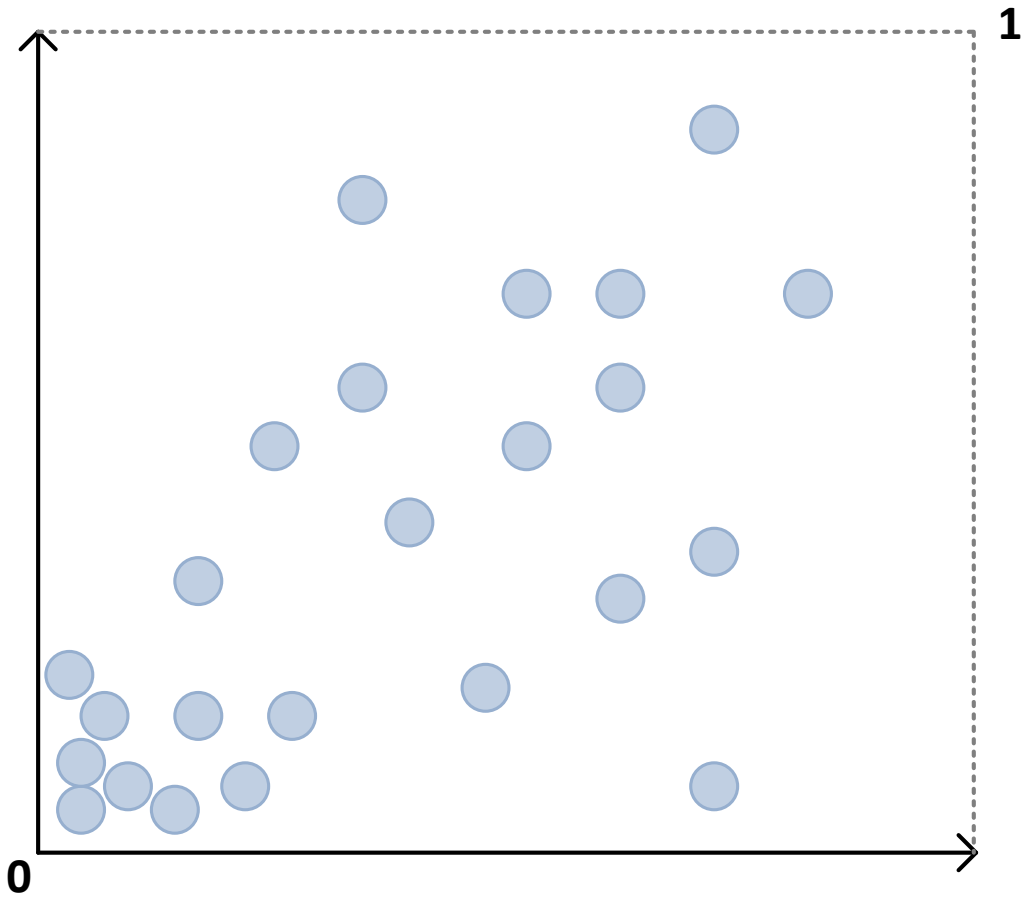
Constrained Clustering:

- Clustern der Vergleichsvektoren in zwei Cluster (z.B. 2-Means)
- Markiere das Cluster das sich näher an Punkt $\mathbf{1} = (1, 1, \ldots, 1)$ befindet als Menge der Matches (und das andere als Menge der Unmatches)
- Vorteil: voll automatisch
- Nachteil: oft keine klare Grenze zwischen den Clustern

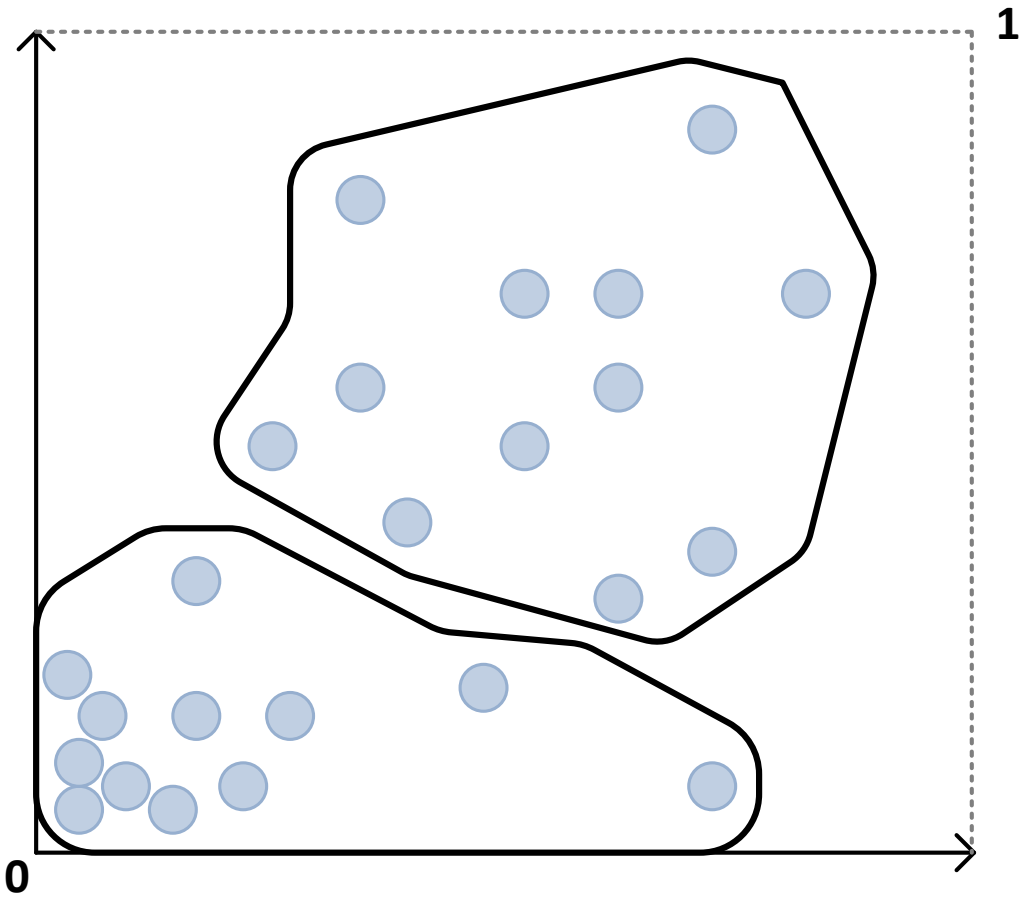
Unconstrained Clustering:

- Verwendung eines Verfahren das keine Clusterzahl vorgibt
- Domänexperte klassifiziert pro Cluster ein Tupelpaar per Hand und adoptiert das Ergebnis für alle Paare des Clusters
- Vorteil: klare Clustergrenzen
- Nachteil: semi-automatisch

Constrained Clustering - Beispiel

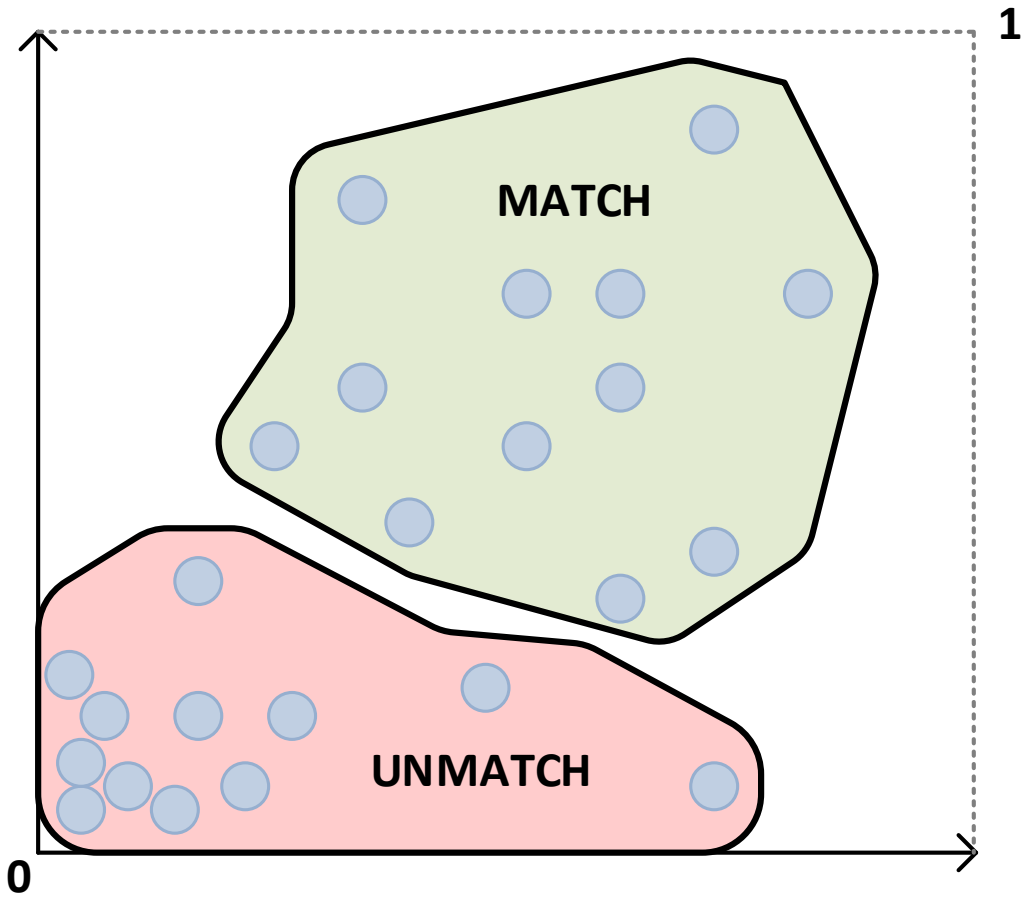


Constrained Clustering - Beispiel



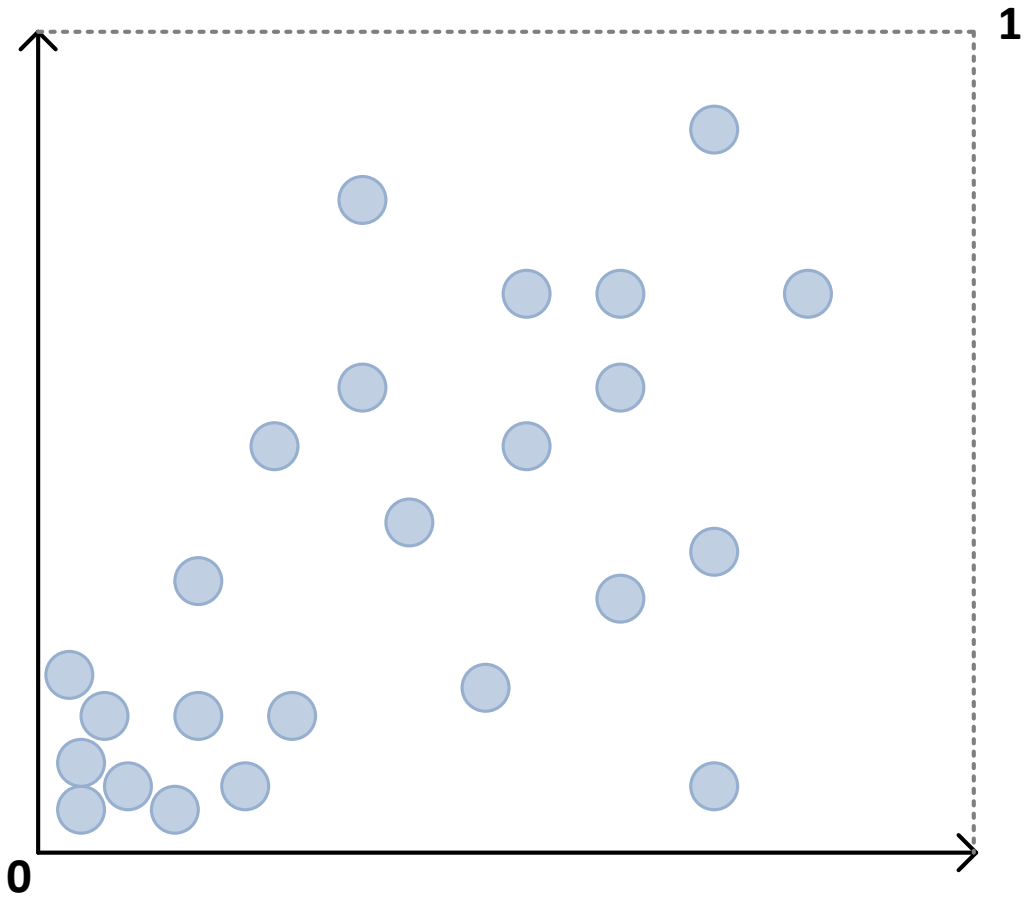
1. Clustern aller Vergleichsvektoren in 2 Cluster

Constrained Clustering - Beispiel

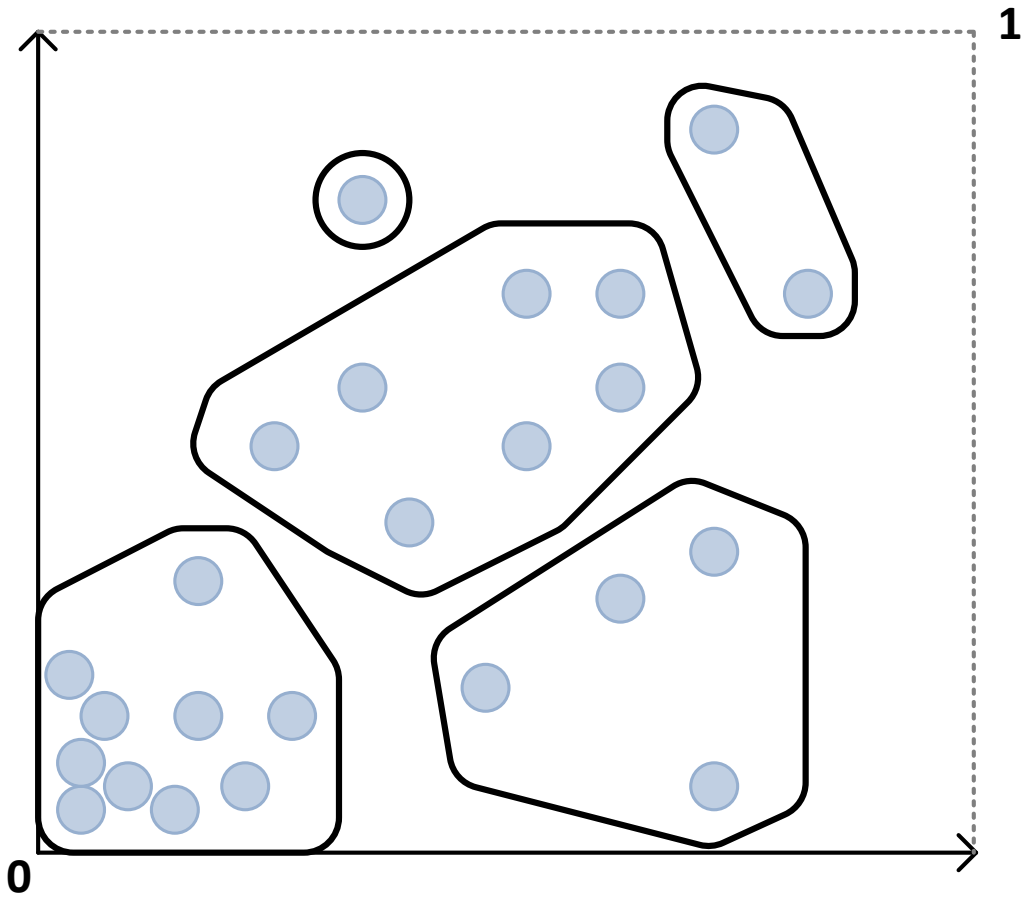


2. Automatische Klassifikation basierend auf den Abständen der Clusterzentren zu Punkt **1**

Unconstrained Clustering - Beispiel

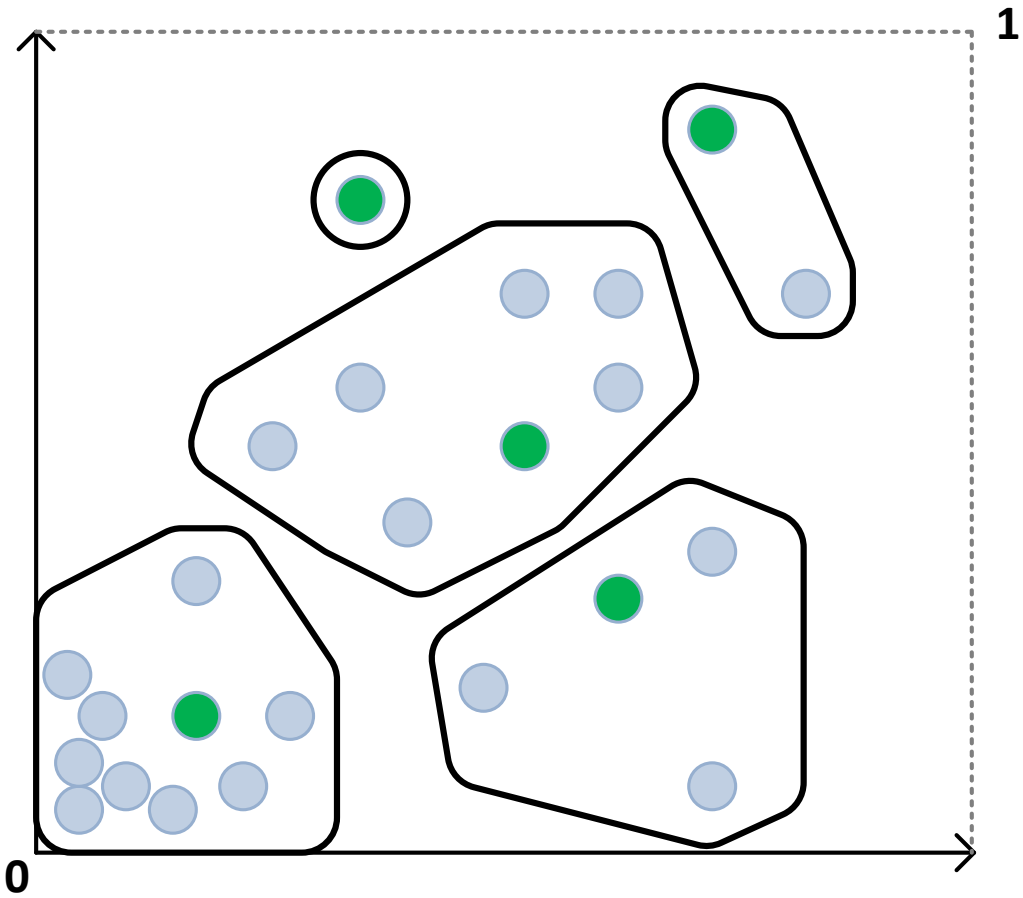


Unconstrained Clustering - Beispiel



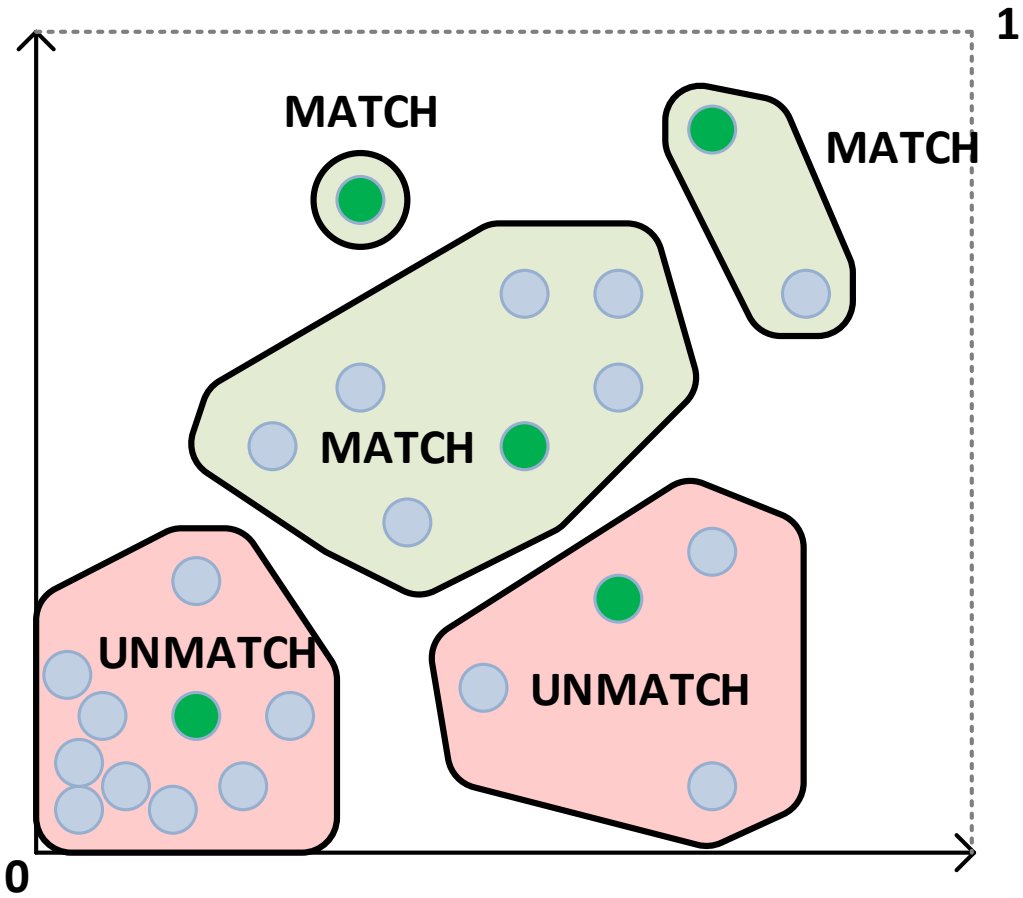
1. Cluster Vergleichsvektoren

Unconstrained Clustering - Beispiel



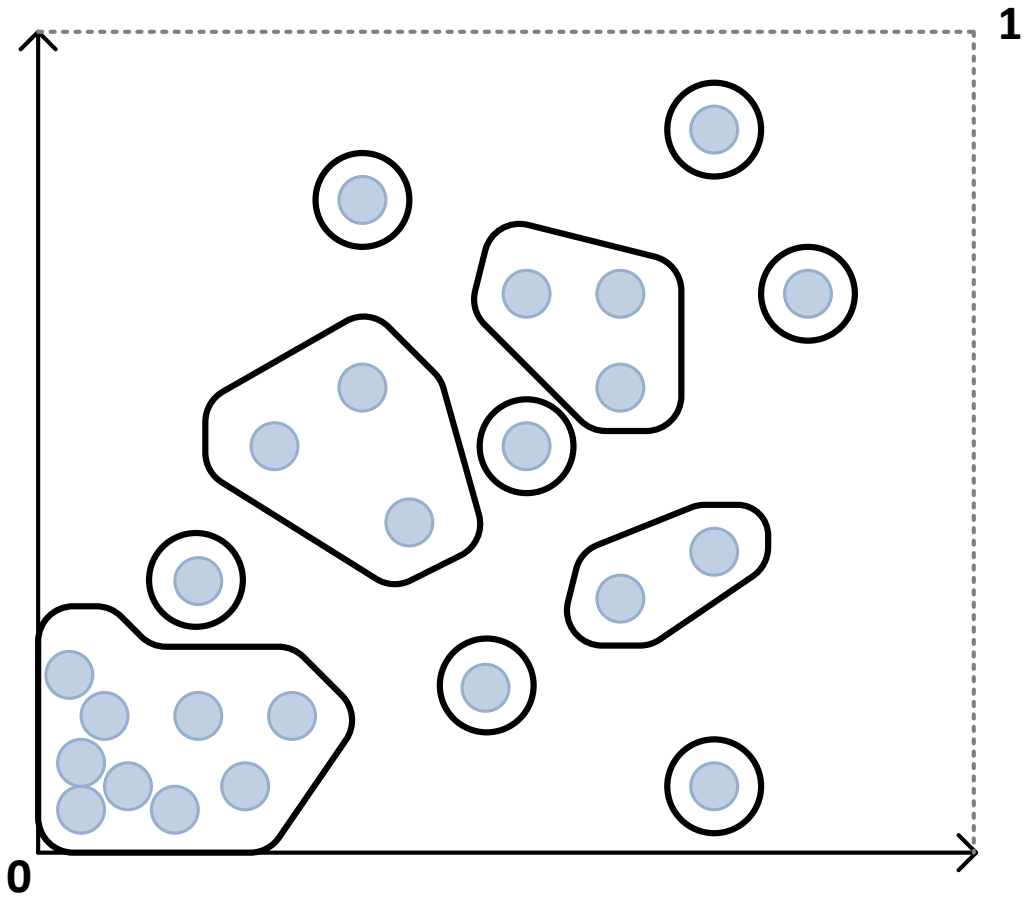
2. Manuelle Klassifikation einiger Vektoren

Unconstrained Clustering - Beispiel



3. Automatische Klassifikation aller anderen Vektoren

Unconstrained Clustering - Beispiel



Problem: Großer manueller Aufwand bei vielen Clustern

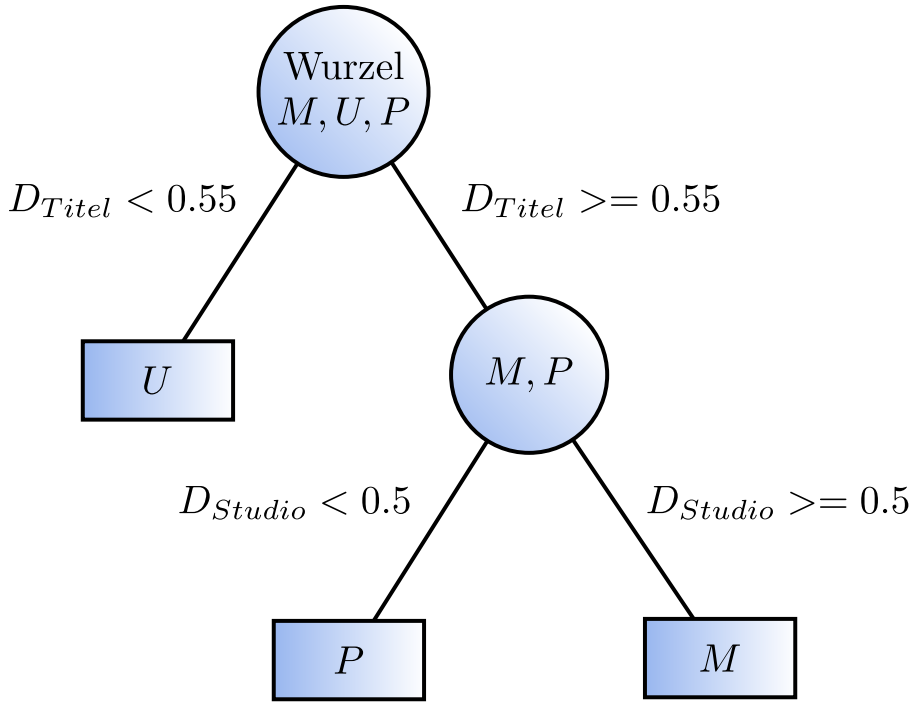
Überwachte Lernverfahren

Entscheidungsbäume:

- Startet mit der Menge aller Trainingsvektoren
- Splittet diese Menge Schritt für Schritt in zwei Untermengen bis alle Vektoren einer Menge gleich gelabelt sind
- In jeder Runde wird versucht die Vektoreneigenschaft für den 'besten' Split zu finden
- Als Splitkriterium (Berwertung von 'besten') wird häufig der *Information Gain* oder der *Giny Index* verwendet
- Das Ergebnis kann als eine Menge von Regeln betrachtet werden (siehe regelbasierte Entscheidungsmodelle)

Entscheidungsbäume (Beispiel)

D_{Titel}	D_{Jahr}	D_{Studio}	Label
1.0	0.8	0.0	P
0.9	1.0	0.0	P
0.2	0.0	0.0	U
0.9	0.8	1.0	M
0.2	0.0	0.3	U
0.2	0.0	0.3	U



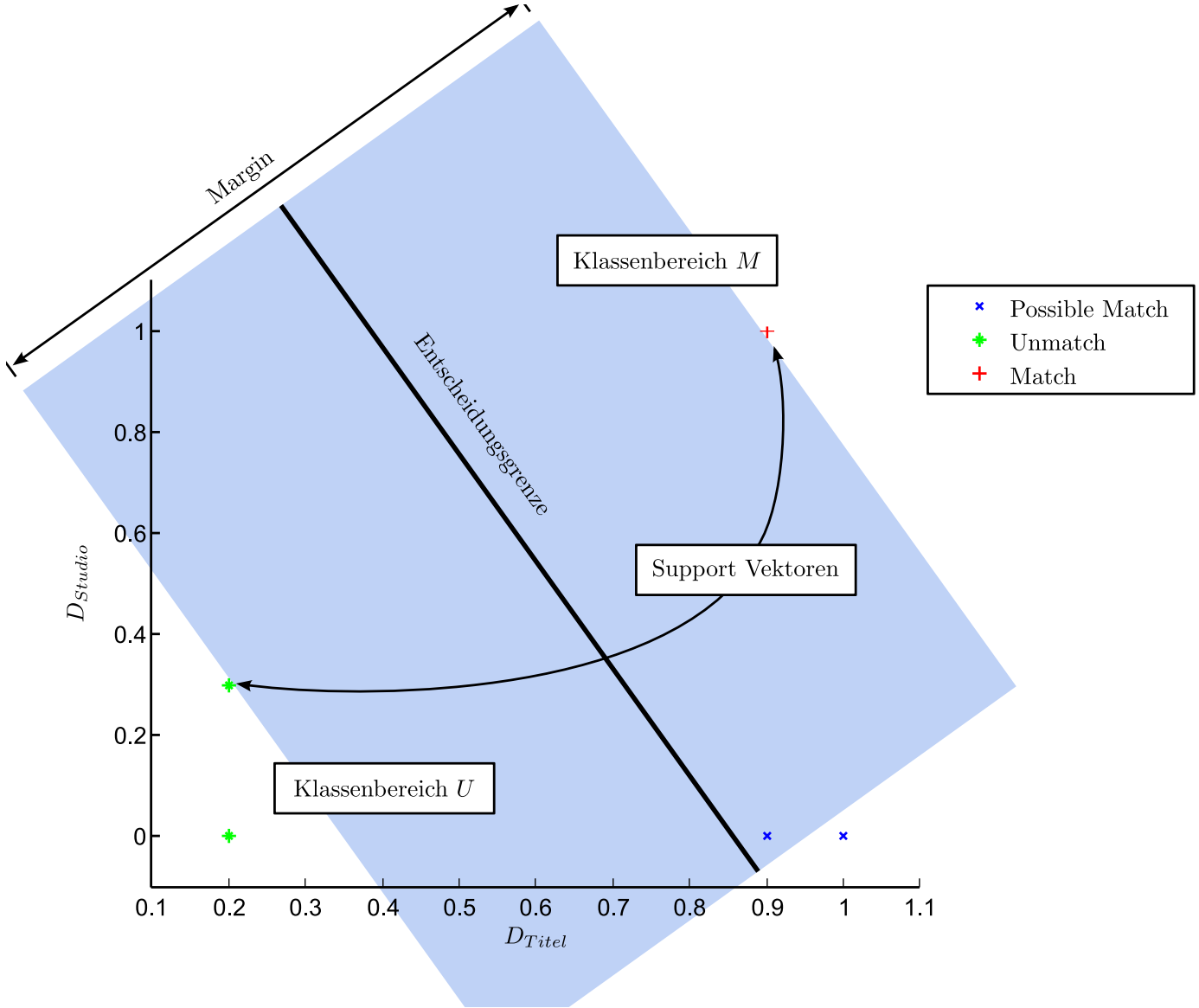
IF $D_{Titel} >= 0.55$ **AND** $D_{Studio} >= 0.5$ **THEN MATCHED.**

Überwachte Lernverfahren

Support-Vector Machine:

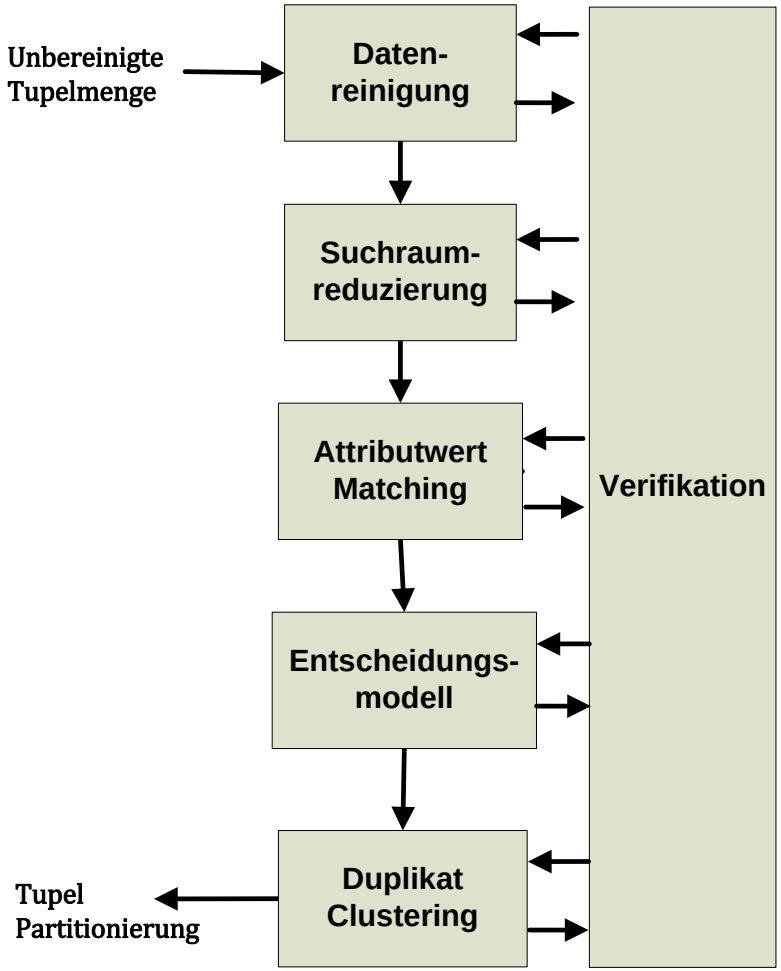
- Betrachtet Vergleichsvektoren als Punkte in mehrdimensionalen Vektorraum
- Erzeugt eine Hyperebene welche die Trainingspunkte in zwei Klassen teilt
- Dabei gilt:
  - Alle gleichgelabelten Vektoren befinden sich in der selben Klasse
  - Die Distanz zwischen dem nächsten Vektor und der Hyperebene ist maximiert
- **Problem:** Lineare Separierung der Vektoren oft nicht möglich
- **Kernel trick:** Transformation in einen höher dimensionalen Raum und Rücktransformation der Hyperebene in eine nicht lineare Funktion

Support-Vector Machine (Beispiel)



Duplikaterkennung - Prozessablauf

- Datenreinigung (Standardisierung, Entfernung einfacher Fehler)
- Suchraumreduzierung (Effizienzerhöhung)
- Attributwertmatching (Messen von Attributwertähnlichkeiten)
- Entscheidungsmodell (ähnlichkeitsbasierte Entscheidungsfindung)
- Duplikat Clustering (Clustern der paarweisen Entscheidungen)
- Verifikation (Abschätzung der Ergebnisqualität)



Duplikat Clustering - Motivation

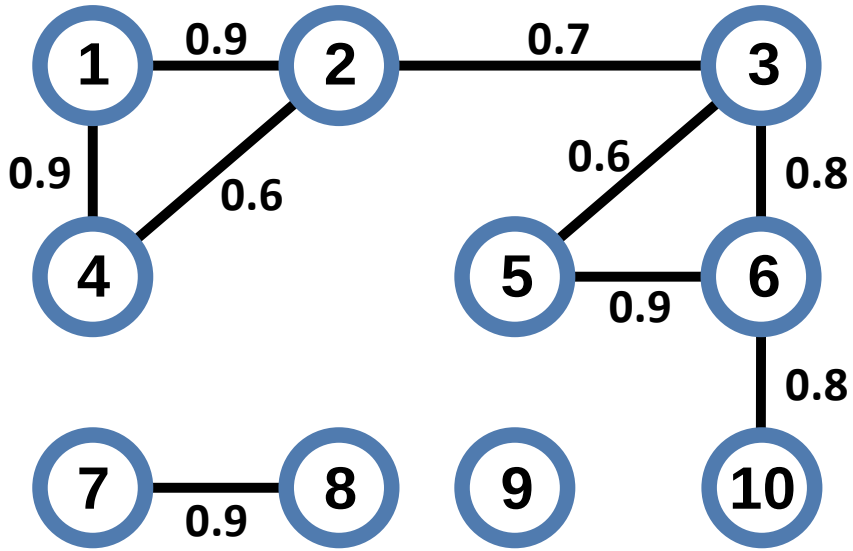
- Input: Eine Menge an paarweisen Duplikatsentscheidungen
- Diese Entscheidungen sind unabhängig voneinander getroffen worden und können in Konflikt stehen

  Beispiel: $\{A, B\}$ und $\{A, C\}$ sind Matches aber $\{B, C\}$ ist ein Unmatch. Da Identität aber transitiv ist, kann das nicht sein

- Ziel: Finden eines Duplikatclusterings das möglichst wenige der paarweise Entscheidungen revidiert

Duplicate-Pair Graph [HCML09, NH10]

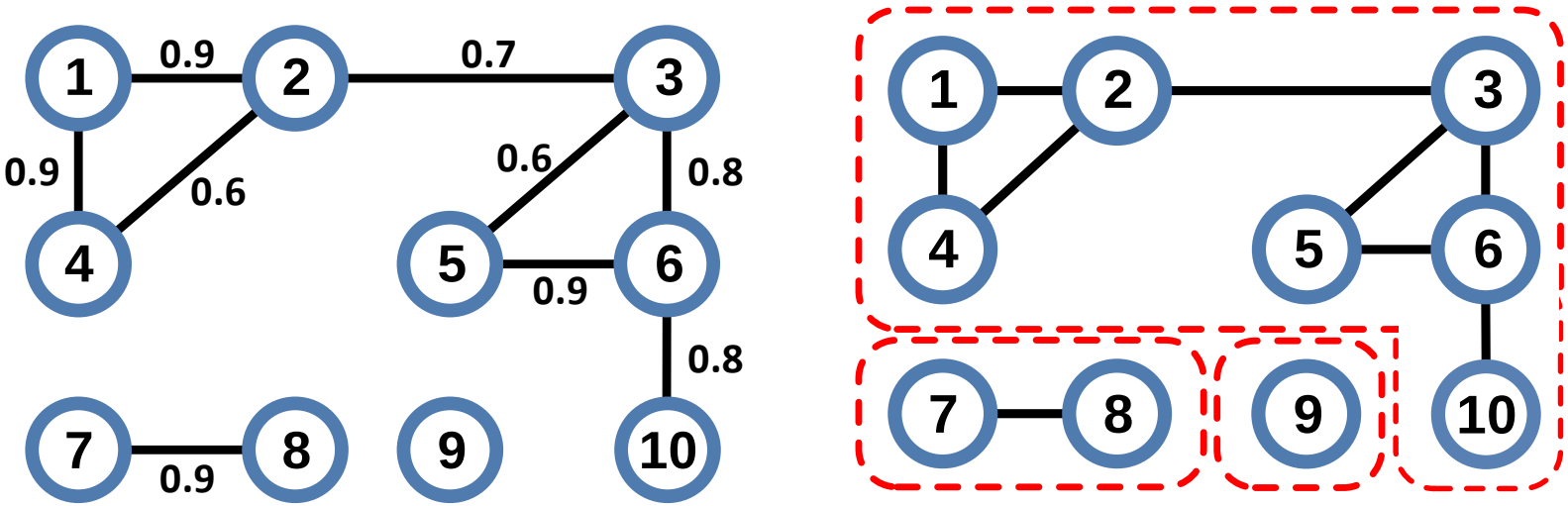
- Viele Verfahren basieren auf dem Duplicate-Pair Graph
- Graph besitzt Tuple als Knoten
- Zwei Knoten sind mit einer Kante verbunden, wenn die zugehörigen Knoten als Match klassifiziert wurden
- Die Ähnlichkeit beider Tupel wird als Kantengewicht genutzt



Partitioning based on Connected Components

- Berechnet die Komponenten des Duplicate-Pair Graphs
- Entspricht der transitive Hülle aller gefundenen Matches
- **Problem:** Produziert große Cluster

$\Rightarrow$ Produziert häufig eine große Zahl an False Positives



Partitioning based on Centers [HCML09, NH10]

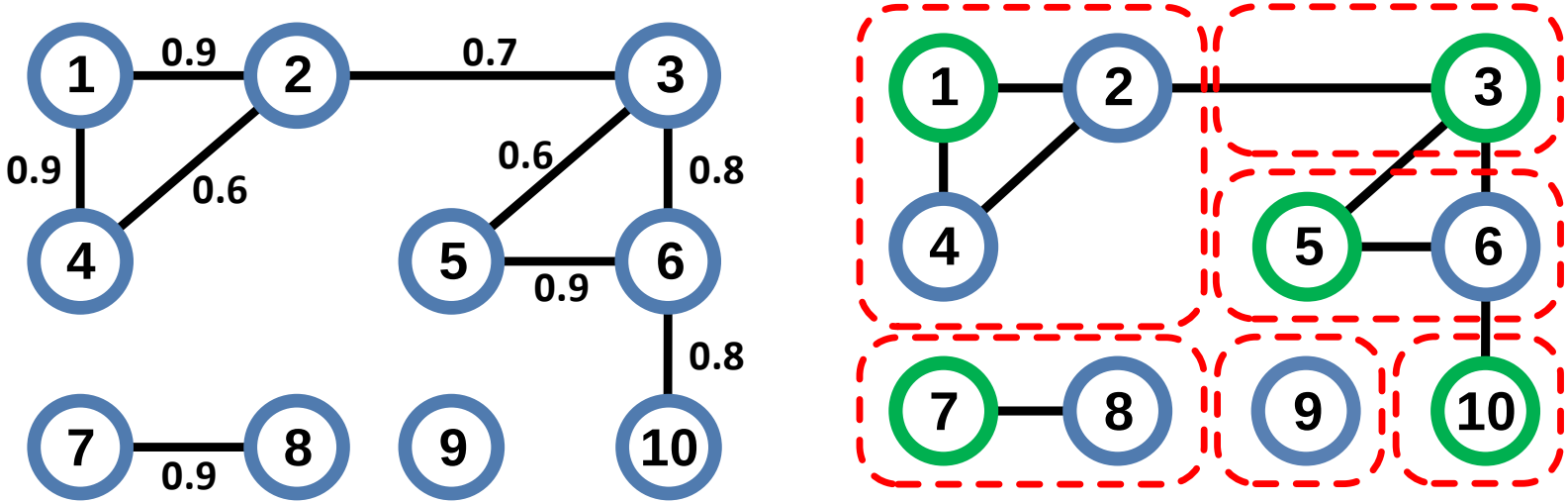
Grundidee:

- Wählt einige Knoten des Duplicate-Pair Graphs als Center
- Weist jeden Knoten seinem nächsten Center zu

Problem:

- Produziert kleine Cluster

$\Rightarrow$ Produziert häufig eine große Zahl an False Negatives

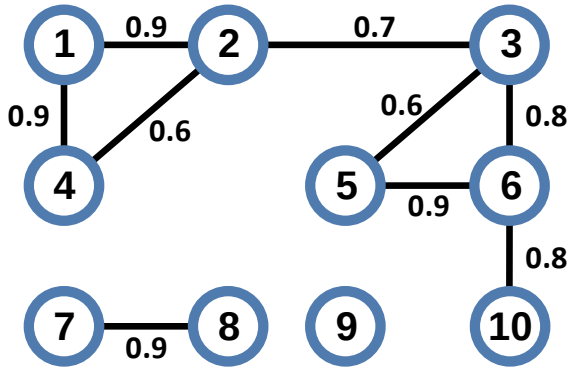


Partitioning based on Centers

Vorgehensweise:

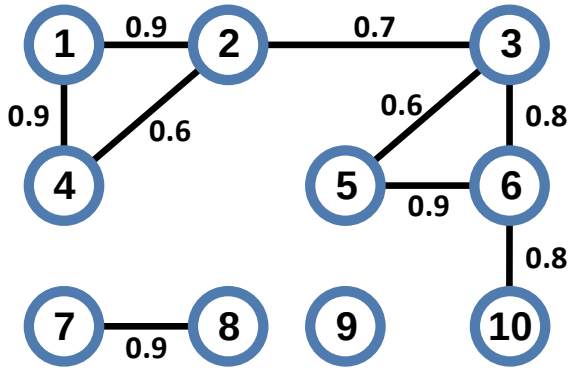
- Sortiere alle Kanten absteigend nach Gewicht
- Itertiere über die sortierte Liste und wählt eine Kante
  1. Beide Knoten gehören schon einem Cluster an
     $\Rightarrow$ Es passiert nichts
  2. Nur ein Knoten gehört einem Cluster an, aber ist kein Center
     $\Rightarrow$ Der clusterlose Knoten wird als neues Center gewählt
  3. Nur ein Knoten gehört einem Cluster an und ist ein Center
     $\Rightarrow$ Der clusterlose Knoten wird dem anderen Cluster hinzugefügt
  4. Keiner der Knoten gehört bereits zu einem Cluster
     $\Rightarrow$ Einer der beiden Knoten wird als neues Center gewählt und der andere Knoten wird dessem Cluster hinzugefügt

Partitioning based on Centers - Beispiel



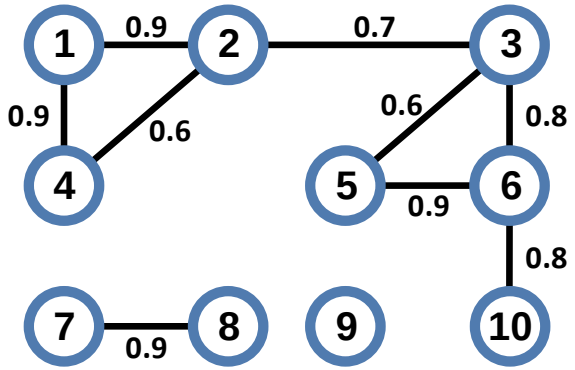
$\{1, 2\}$
$\{1, 4\}$
$\{5, 6\}$
$\{7, 8\}$
$\{3, 6\}$
$\{6, 10\}$
$\{2, 3\}$
$\{2, 4\}$
$\{3, 5\}$

Partitioning based on Centers - Beispiel



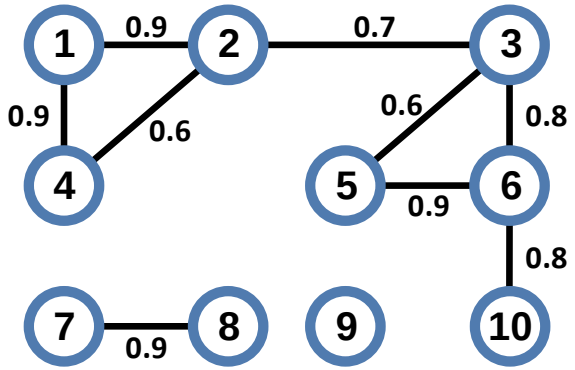
$\{1,2\}$ Fall 4 1 neuer Center $\{\langle 1,2\rangle\}$
$\{1,4\}$
$\{5,6\}$
$\{7,8\}$
$\{3,6\}$
$\{6,10\}$
$\{2,3\}$
$\{2,4\}$
$\{3,5\}$

Partitioning based on Centers - Beispiel



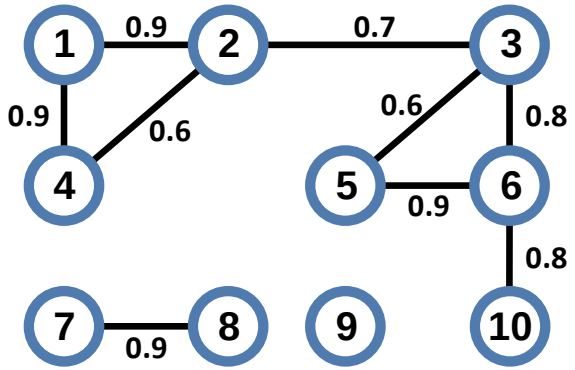
$\{1,2\}$	Fall 4	1 neuer Center	$\{\langle 1,2\rangle\}$
$\{1,4\}$	Fall 3	-	$\{\langle 1,2,4\rangle\}$
$\{5,6\}$			
$\{7,8\}$			
$\{3,6\}$			
$\{6,10\}$			
$\{2,3\}$			
$\{2,4\}$			
$\{3,5\}$			

Partitioning based on Centers - Beispiel



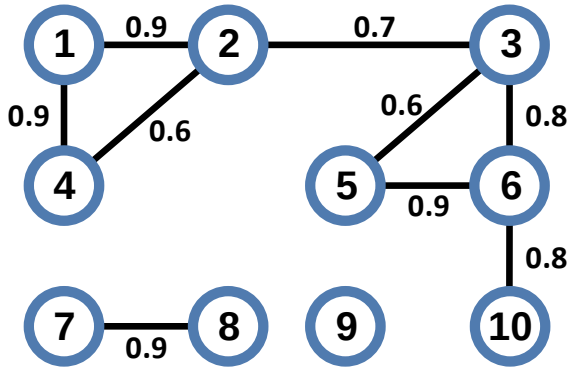
$\{1,2\}$	Fall 4	1 neuer Center	$\{\langle 1,2\rangle\}$
$\{1,4\}$	Fall 3	-	$\{\langle 1,2,4\rangle\}$
$\{5,6\}$	Fall 4	5 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle\}$
$\{7,8\}$			
$\{3,6\}$			
$\{6,10\}$			
$\{2,3\}$			
$\{2,4\}$			
$\{3,5\}$			

Partitioning based on Centers - Beispiel



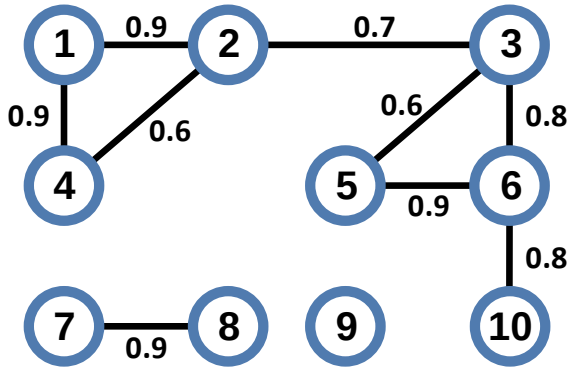
$\{1,2\}$	Fall 4	1 neuer Center	$\{\langle 1,2\rangle\}$
$\{1,4\}$	Fall 3	-	$\{\langle 1,2,4\rangle\}$
$\{5,6\}$	Fall 4	5 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle\}$
$\{7,8\}$	Fall 4	7 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle, \langle 7,8\rangle\}$
$\{3,6\}$			
$\{6,10\}$			
$\{2,3\}$			
$\{2,4\}$			
$\{3,5\}$			

Partitioning based on Centers - Beispiel



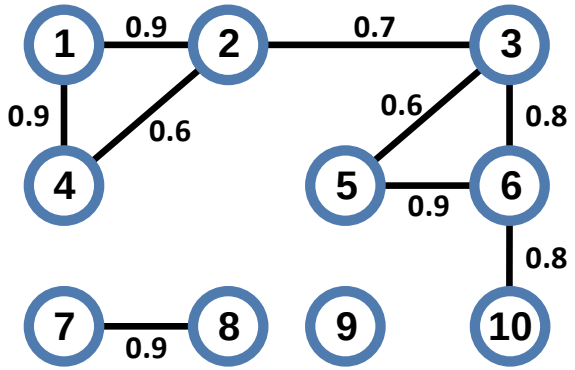
$\{1,2\}$	Fall 4	1 neuer Center	$\{\langle 1,2\rangle\}$
$\{1,4\}$	Fall 3	-	$\{\langle 1,2,4\rangle\}$
$\{5,6\}$	Fall 4	5 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle\}$
$\{7,8\}$	Fall 4	7 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle, \langle 7,8\rangle\}$
$\{3,6\}$	Fall 2	3 neuer Center	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle\}$
$\{6,10\}$			
$\{2,3\}$			
$\{2,4\}$			
$\{3,5\}$			

Partitioning based on Centers - Beispiel



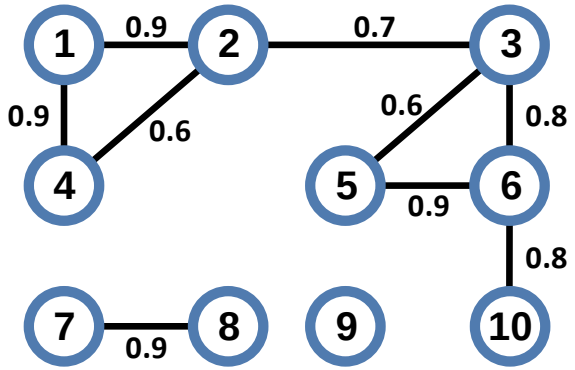
$\{1,2\}$	Fall 4	1 neuer Center	$\{\langle 1,2\rangle\}$
$\{1,4\}$	Fall 3	-	$\{\langle 1,2,4\rangle\}$
$\{5,6\}$	Fall 4	5 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle\}$
$\{7,8\}$	Fall 4	7 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle, \langle 7,8\rangle\}$
$\{3,6\}$	Fall 2	3 neuer Center	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle\}$
$\{6,10\}$	Fall 2	10 neuer Center	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle, \langle 10\rangle\}$
$\{2,3\}$			
$\{2,4\}$			
$\{3,5\}$			

Partitioning based on Centers - Beispiel



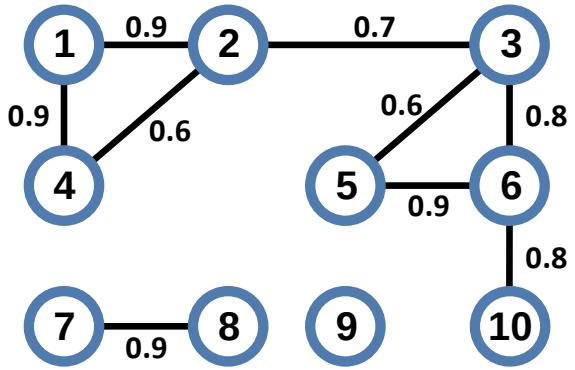
$\{1,2\}$	Fall 4	1 neuer Center	$\{\langle 1,2\rangle\}$
$\{1,4\}$	Fall 3	-	$\{\langle 1,2,4\rangle\}$
$\{5,6\}$	Fall 4	5 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle\}$
$\{7,8\}$	Fall 4	7 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle, \langle 7,8\rangle\}$
$\{3,6\}$	Fall 2	3 neuer Center	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle\}$
$\{6,10\}$	Fall 2	10 neuer Center	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle, \langle 10\rangle\}$
$\{2,3\}$	Fall 1	-	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle, \langle 10\rangle\}$
$\{2,4\}$			
$\{3,5\}$			

Partitioning based on Centers - Beispiel



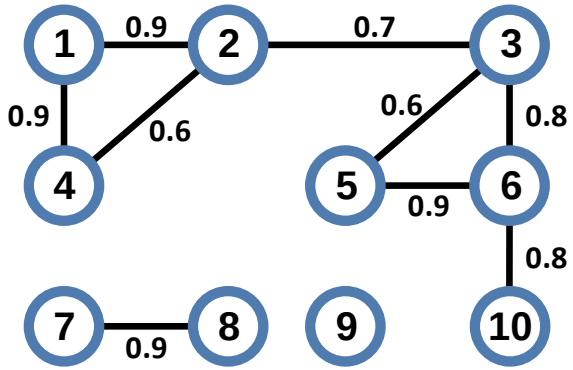
$\{1,2\}$	Fall 4	1 neuer Center	$\{\langle 1,2\rangle\}$
$\{1,4\}$	Fall 3	-	$\{\langle 1,2,4\rangle\}$
$\{5,6\}$	Fall 4	5 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle\}$
$\{7,8\}$	Fall 4	7 neuer Center	$\{\langle 1,2,4\rangle, \langle 5,6\rangle, \langle 7,8\rangle\}$
$\{3,6\}$	Fall 2	3 neuer Center	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle\}$
$\{6,10\}$	Fall 2	10 neuer Center	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle, \langle 10\rangle\}$
$\{2,3\}$	Fall 1	-	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle, \langle 10\rangle\}$
$\{2,4\}$	Fall 1	-	$\{\langle 1,2,4\rangle, \langle 3\rangle, \langle 5,6\rangle, \langle 7,8\rangle, \langle 10\rangle\}$
$\{3,5\}$			

Partitioning based on Centers - Beispiel



$\{1,2\}$	Fall 4	1 neuer Center	$\{\langle 1,2\rangle\}$
$\{1,4\}$	Fall 3	-	$\{\langle 1,2,4\rangle\}$
$\{5,6\}$	Fall 4	5 neuer Center	$\{\langle 1,2,4\rangle,\langle 5,6\rangle\}$
$\{7,8\}$	Fall 4	7 neuer Center	$\{\langle 1,2,4\rangle,\langle 5,6\rangle,\langle 7,8\rangle\}$
$\{3,6\}$	Fall 2	3 neuer Center	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle\}$
$\{6,10\}$	Fall 2	10 neuer Center	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle,\langle 10\rangle\}$
$\{2,3\}$	Fall 1	-	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle,\langle 10\rangle\}$
$\{2,4\}$	Fall 1	-	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle,\langle 10\rangle\}$
$\{3,5\}$	Fall 1	-	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle,\langle 10\rangle\}$

Partitioning based on Centers - Beispiel



$\{1,2\}$	Fall 4	1 neuer Center	$\{\langle 1,2\rangle\}$
$\{1,4\}$	Fall 3	-	$\{\langle 1,2,4\rangle\}$
$\{5,6\}$	Fall 4	5 neuer Center	$\{\langle 1,2,4\rangle,\langle 5,6\rangle\}$
$\{7,8\}$	Fall 4	7 neuer Center	$\{\langle 1,2,4\rangle,\langle 5,6\rangle,\langle 7,8\rangle\}$
$\{3,6\}$	Fall 2	3 neuer Center	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle\}$
$\{6,10\}$	Fall 2	10 neuer Center	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle,\langle 10\rangle\}$
$\{2,3\}$	Fall 1	-	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle,\langle 10\rangle\}$
$\{2,4\}$	Fall 1	-	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle,\langle 10\rangle\}$
$\{3,5\}$	Fall 1	-	$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle,\langle 10\rangle\}$
Knoten 9 hat keine Kante			$\{\langle 1,2,4\rangle,\langle 3\rangle,\langle 5,6\rangle,\langle 7,8\rangle,\langle 9\rangle,\langle 10\rangle\}$

Edge Removal [NH10]

Grundidee:

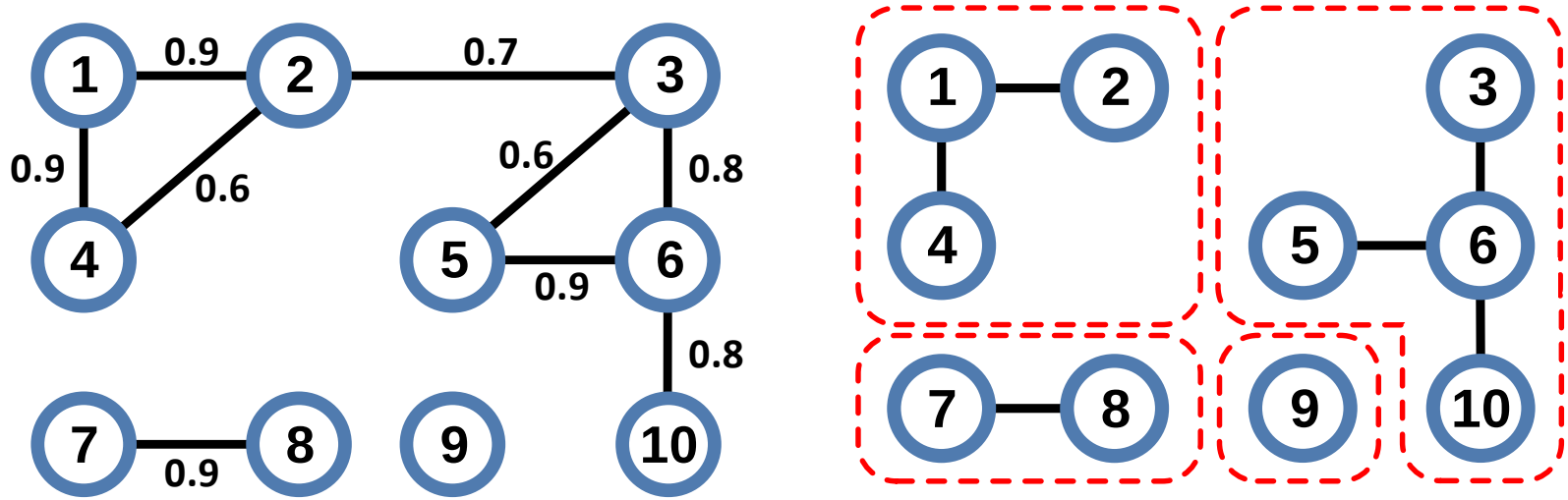
- Entfernt solange die Kante mit dem kleinsten Gewicht vom Duplicate-Pair Graph bis ein Stopp Kriterium erfüllt ist
- Verwendet die Komponenten des modifizierten Graphen als Cluster
- **Typisches Stopp Kriterium:** Der kürzeste Pfad zwischen jedem Knotenpaar ist kleiner als n
- Kann für jede Komponente separat ausgeführt werden (entfernt weniger Kanten)

Problem:

- Kann berechnungsintensiv sein
- Entfernte Kanten können neue längste kürzeste Pfade erzeugen

$\Rightarrow$ Eine große Zahl von Kanten wird entfernt werden

Edge Removal - Beispiel



Einfaches Beispiel: Entferne alle Kanten mit Gewicht kleiner als 0.8

Nutzen von Beziehungsinformationen in komplexen DB

- Nachbarschaftsähnlichkeit:
  - Ähnlichkeit der korrelierenden Datenobjekte
  - Beispiel: Anteil gleicher Akteure zweier Filme

Nutzen von Beziehungsinformationen in komplexen DB

- Nachbarschaftsähnlichkeit:
  - Ähnlichkeit der korrelierenden Datenobjekte
  - Beispiel: Anteil gleicher Akteure zweier Filme
- Kollektive Duplikaterkennung:
  - Duplikatsentscheidung zweier Objekte hat Einfluß auf Duplikatsentscheidungen benachbarter Objekte
  - Beispiel: Zwei als Duplikat identifizierte Schauspieler erhöhen die Wahrscheinlichkeit, dass deren Filme auch Duplikate sind

Nutzen von Beziehungsinformationen in komplexen DB

- Nachbarschaftsähnlichkeit:
  - Ähnlichkeit der korrelierenden Datenobjekte
  - Beispiel: Anteil gleicher Akteure zweier Filme
- Kollektive Duplikaterkennung:
  - Duplikatsentscheidung zweier Objekte hat Einfluß auf Duplikatsentscheidungen benachbarter Objekte
  - Beispiel: Zwei als Duplikat identifizierte Schauspieler erhöhen die Wahrscheinlichkeit, dass deren Filme auch Duplikate sind
  - Iterativ über vollständige Duplikatclusterings
  - Queue-basiert für einzelne Objektpaare

Nutzen von Beziehungsinformationen in komplexen DB

- Nachbarschaftsähnlichkeit:
  - Ähnlichkeit der korrelierenden Datenobjekte
  - Beispiel: Anteil gleicher Akteure zweier Filme
- Kollektive Duplikaterkennung:
  - Duplikatsentscheidung zweier Objekte hat Einfluß auf Duplikatsentscheidungen benachbarter Objekte
  - Beispiel: Zwei als Duplikat identifizierte Schauspieler erhöhen die Wahrscheinlichkeit, dass deren Filme auch Duplikate sind
  - Iterativ über vollständige Duplikatclusterings
  - Queue-basiert für einzelne Objektpaare
- Negative Regeln:
  - Objektbeziehungen die auf Nichtduplikate hinweisen
  - Beispiel: Personen die gemeinsam Autor eines Artikels sind

Literatur I

[DHI12] Anhai Doan, Alon Halevy, and Zachary Ives.
Principles of Data Integration.
Morgan Kaufmann, 2012.

[DN09] Luna Dong and Felix Naumann.
Data Fusion.
VLDB Tutorial, 2009.

[dVKCC09] Timothy de Vries, Hui Ke, Sanjay Chawla, and Peter Christen.
Robust record linkage blocking using suffix arrays.
In *CIKM*, pages 305-314, 2009.

[EIV07] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios.
Duplicate Record Detection: A Survey.
IEEE Trans. Knowl. Data Eng., 19(1):1-16, 2007.

[HCML09] Oktie Hassanzadeh, Fei Chiang, Renee J. Miller, and Hyun Chul Lee.
Framework for evaluating clustering algorithms in duplicate detection.
PVLDB,2(1):1282-1293, 2009.

Literatur II

[HS95] Mauricio A. Hernandez and Salvatore J. Stolfo.
The Merge/Purge Problem for Large Databases.
In *SIGMOD Conference*, pages 127-138, 1995.

[LN06] Ulf Leser and Felix Naumann.
Informationsintegration.
dpunkt.verlag, 2006.
In German.

[MWGM10] David Menestrina, Steven Whang, and Hector Garcia-Molina.
Evaluating entity resolution results.
PVLDB, 3(1):208-219, 2010.

[NH10] Felix Naumann and Melanie Herschel.
An Introduction to Duplicate Detection.
Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

Literatur III

[Tal11] John R. Talburt.
Entity Resolution and Information Quality.
Morgan Kaufman Publ. Inc., 2011.

[WM89] Y. Richard Wang and Stuart E. Madnick.
The Inter-Database Instance Identification Problem in Integrating Autonomous Systems.
In *Proceedings of the 5th International Conference on Data Engineering (ICDE)*, pages 46-55, 1989.