

# String Matching

## Komplexe Informationssysteme

---

Fabian Panse

panse@informatik.uni-hamburg.de

Universität Hamburg



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



# Agenda

---

- 1 Problemstellung
- 2 Genauigkeit
  - Mengenbasierte Ähnlichkeitsmaße
  - Sequenzbasierte Ähnlichkeitsmaße
  - Phonetische Ähnlichkeitsmaße
  - Semantische Ähnlichkeitsmaße
  - Hybride Ähnlichkeitsmaße
- 3 Skalierbarkeit

# Einordnung

---

# Einordnung

---

- Identifizierung von Stringwerten die das gleiche Realwelt-Objekt referenzieren

# Einordnung

---

- Identifizierung von Stringwerten die das gleiche Realwelt-Objekt referenzieren
  - Namen von Personen (z.B. 'Christina' vs. 'Kristina')

# Einordnung

---

- Identifizierung von Stringwerten die das gleiche Realwelt-Objekt referenzieren
  - Namen von Personen (z.B. 'Christina' vs. 'Kristina')
  - Namen von Einrichtungen (z.B. 'HPI' vs. 'Hasso-Plattner Institut')

# Einordnung

---

- Identifizierung von Stringwerten die das gleiche Realwelt-Objekt referenzieren
  - Namen von Personen (z.B. 'Christina' vs. 'Kristina')
  - Namen von Einrichtungen (z.B. 'HPI' vs. 'Hasso-Plattner Institut')
  - Adressen  
( '1210 W. Dayton St Madison WI' vs. '1210 West Dayton Madison WI 53706' )

# Einordnung

---

- Identifizierung von Stringwerten die das gleiche Realwelt-Objekt referenzieren
  - Namen von Personen (z.B. 'Christina' vs. 'Kristina')
  - Namen von Einrichtungen (z.B. 'HPI' vs. 'Hasso-Plattner Institut')
  - Adressen  
( '1210 W. Dayton St Madison WI' vs. '1210 West Dayton Madison WI 53706' )
  - Telefonnummern ('040-12345' vs. '(040)12354')

# Einordnung

---

- Identifizierung von Stringwerten die das gleiche Realwelt-Objekt referenzieren
  - Namen von Personen (z.B. 'Christina' vs. 'Kristina')
  - Namen von Einrichtungen (z.B. 'HPI' vs. 'Hasso-Plattner Institut')
  - Adressen  
( '1210 W. Dayton St Madison WI' vs. '1210 West Dayton Madison WI 53706' )
  - Telefonnummern ('040-12345' vs. '(040)12354')
- Spielt eine tragende Rolle in vielen Integrationsaufgaben
  - Schema Matching, Duplicate Detection, etc.

# Einordnung

---

- Identifizierung von Stringwerten die das gleiche Realwelt-Objekt referenzieren
  - Namen von Personen (z.B. 'Christina' vs. 'Kristina')
  - Namen von Einrichtungen (z.B. 'HPI' vs. 'Hasso-Plattner Institut')
  - Adressen  
( '1210 W. Dayton St Madison WI' vs. '1210 West Dayton Madison WI 53706' )
  - Telefonnummern ('040-12345' vs. '(040)12354')
- Spielt eine tragende Rolle in vielen Integrationsaufgaben
  - Schema Matching, Duplicate Detection, etc.
- Ausblick
  - Problemdefinition
  - Maße um die Ähnlichkeit zweier Stringwerte numerisch zu quantifizieren
  - Techniken zur Performanzoptimierung

# Problemdefinition

---

# Problemdefinition

---

- Gegeben sind zwei Mengen  $A$  and  $B$  an Stringwerten

# Problemdefinition

---

- Gegeben sind zwei Mengen  $A$  and  $B$  an Stringwerten
  - Finde alle Paare  $(x, y)$  mit  $x \in A$  und  $y \in B$  die dasselbe Realwelt-Objekt referenzieren

# Problemdefinition

---

- Gegeben sind zwei Mengen  $A$  and  $B$  an Stringwerten
  - Finde alle Paare  $(x, y)$  mit  $x \in A$  und  $y \in B$  die dasselbe Realwelt-Objekt referenzieren
  - Ergebnispaae werden als *Match* bezeichnet

# Problemdefinition

- Gegeben sind zwei Mengen  $A$  and  $B$  an Stringwerten
  - Finde alle Paare  $(x, y)$  mit  $x \in A$  und  $y \in B$  die dasselbe Realwelt-Objekt referenzieren
  - Ergebnispaaire werden als *Match* bezeichnet
  - Beispiel (aus [DHI12])

<b>Menge A</b>	<b>Menge B</b>	<b>Matches</b>
$x_1$ =Dave Smith	$y_1$ =David D. Smith	$(x_1, y_1)$
$x_2$ =Joe Wilson	$y_2$ =Daniel D. Smith	$(x_3, y_2)$
$x_3$ =Dan Smith		

# Problemdefinition

- Gegeben sind zwei Mengen  $A$  and  $B$  an Stringwerten
  - Finde alle Paare  $(x, y)$  mit  $x \in A$  und  $y \in B$  die dasselbe Realwelt-Objekt referenzieren
  - Ergebnispaaire werden als *Match* bezeichnet
  - Beispiel (aus [DHI12])

<b>Menge A</b>	<b>Menge B</b>	<b>Matches</b>
$x_1 = \text{Dave Smith}$	$y_1 = \text{David D. Smith}$	$(x_1, y_1)$
$x_2 = \text{Joe Wilson}$	$y_2 = \text{Daniel D. Smith}$	$(x_3, y_2)$
$x_3 = \text{Dan Smith}$		

- Herausforderungen: Genauigkeit und Skalierbarkeit

# Herausforderungen: Genauigkeit

---

- **Google** meldet 593\* verschiedene Schreibweisen von 'Britney Spears' in Suchanfragen
  - 'Britney Spears' (488941 Suchen)
  - 'Brittany Spears' (40134 Suchen)
  - 'Brittney Spears' (36315 Suchen)
  - 'Britany Spears' (24342 Suchen)
  - 'Britny Spears' (7331 Suchen)
  - ...

\*Quelle: <http://www.netpaths.net/blog/britney-spears-spelling-variations/>

# Herausforderungen: Genauigkeit

---

- **Google** meldet 593\* verschiedene Schreibweisen von 'Britney Spears' in Suchanfragen
  - 'Britney Spears' (488941 Suchen)
  - 'Brittany Spears' (40134 Suchen)
  - 'Brittney Spears' (36315 Suchen)
  - 'Britany Spears' (24342 Suchen)
  - 'Britny Spears' (7331 Suchen)
  - ...
- wieviele verschiedene Schreibweisen für komplexere Namen?

\*Quelle: <http://www.netpaths.net/blog/britney-spears-spelling-variations/>

# Herausforderungen: Genauigkeit

---

- **Google** meldet 593\* verschiedene Schreibweisen von 'Britney Spears' in Suchanfragen
  - 'Britney Spears' (488941 Suchen)
  - 'Brittany Spears' (40134 Suchen)
  - 'Brittney Spears' (36315 Suchen)
  - 'Britany Spears' (24342 Suchen)
  - 'Britny Spears' (7331 Suchen)
  - ...
- wieviele verschiedene Schreibweisen für komplexere Namen?
  - 'Giannis Antetokounmpo' (griechischer Basketballspieler)

\*Quelle: <http://www.netpaths.net/blog/britney-spears-spelling-variations/>

# Herausforderungen: Genauigkeit

---

- **Google** meldet 593\* verschiedene Schreibweisen von 'Britney Spears' in Suchanfragen
  - 'Britney Spears' (488941 Suchen)
  - 'Brittany Spears' (40134 Suchen)
  - 'Brittney Spears' (36315 Suchen)
  - 'Britany Spears' (24342 Suchen)
  - 'Britny Spears' (7331 Suchen)
  - ...
- wieviele verschiedene Schreibweisen für komplexere Namen?
  - 'Giannis Antetokounmpo' (griechischer Basketballspieler)
  - 'Dharmavarapu Subramanyam' (indischer Schauspieler)

\*Quelle: <http://www.netpaths.net/blog/britney-spears-spelling-variations/>

# Herausforderungen: Genauigkeit

---

- **Google** meldet 593\* verschiedene Schreibweisen von 'Britney Spears' in Suchanfragen
  - 'Britney Spears' (488941 Suchen)
  - 'Brittany Spears' (40134 Suchen)
  - 'Brittney Spears' (36315 Suchen)
  - 'Britany Spears' (24342 Suchen)
  - 'Britny Spears' (7331 Suchen)
  - ...
- wieviele verschiedene Schreibweisen für komplexere Namen?
  - 'Giannis Antetokounmpo' (griechischer Basketballspieler)
  - 'Dharmavarapu Subramanyam' (indischer Schauspieler)
  - 'Janice Keihanaikukauakahihuliheekahaunaele'  
(hawaiianische Frau)

\*Quelle: <http://www.netpaths.net/blog/britney-spears-spelling-variations/>

# Herausforderungen: Genauigkeit

---

- **Google** meldet 593\* verschiedene Schreibweisen von 'Britney Spears' in Suchanfragen
  - 'Britney Spears' (488941 Suchen)
  - 'Brittany Spears' (40134 Suchen)
  - 'Brittney Spears' (36315 Suchen)
  - 'Britany Spears' (24342 Suchen)
  - 'Britny Spears' (7331 Suchen)
  - ...
- wieviele verschiedene Schreibweisen für komplexere Namen?
  - 'Giannis Antetokounmpo' (griechischer Basketballspieler)
  - 'Dharmavarapu Subramanyam' (indischer Schauspieler)
  - 'Janice Keihanaikukauakahihuliheekahaunaele'  
(hawaiianische Frau)
  - 'Venkatanarasimharajuvaripeta railway station'  
(Bahnhof in Indien)

\*Quelle: <http://www.netpaths.net/blog/britney-spears-spelling-variations/>

# Herausforderungen: Genauigkeit

---

Arten von Fehlerquellen:

# Herausforderungen: Genauigkeit

---

Arten von Fehlerquellen:

- Tippfehler (z.B. 'Spears' vs. 'Spear')

# Herausforderungen: Genauigkeit

---

Arten von Fehlerquellen:

- Tippfehler (z.B. 'Spears' vs. 'Speas')
- Unterschiedliche Konventionen  
(z.B. 'Britney Jean Spears' vs. 'Spears, Britney J.')

# Herausforderungen: Genauigkeit

---

## Arten von Fehlerquellen:

- Tippfehler (z.B. 'Spears' vs. 'Speasr')
- Unterschiedliche Konventionen  
(z.B. 'Britney Jean Spears' vs. 'Spears, Britney J.')
- fehlerhafte Transformation vom Gesprochenen ins Geschriebene  
(z.B. 'Spears' vs. 'Speers')

# Herausforderungen: Genauigkeit

---

## Arten von Fehlerquellen:

- Tippfehler (z.B. 'Spears' vs. 'Speasr')
- Unterschiedliche Konventionen  
(z.B. 'Britney Jean Spears' vs. 'Spears, Britney J.')
- fehlerhafte Transformation vom Gesprochenen ins Geschriebene  
(z.B. 'Spears' vs. 'Speers')
- fehlerhafte Transformation vom Handschriftlichen ins Maschinelle (z.B. 'Spears' vs. 'Speans')

# Herausforderungen: Genauigkeit

---

## Arten von Fehlerquellen:

- Tippfehler (z.B. 'Spears' vs. 'Speasr')
- Unterschiedliche Konventionen  
(z.B. 'Britney Jean Spears' vs. 'Spears, Britney J.')
- fehlerhafte Transformation vom Gesprochenen ins Geschriebene  
(z.B. 'Spears' vs. 'Speers')
- fehlerhafte Transformation vom Handschriftlichen ins Maschinelle (z.B. 'Spears' vs. 'Speans')
- Spitznamen, Künstlernamen oder Synonyme  
(z.B. 'Allan Stewart Konigsberg' vs. 'Heywood Allen' vs. 'Woody Allen')

# Herausforderungen: Genauigkeit

---

## Arten von Fehlerquellen:

- Tippfehler (z.B. 'Spears' vs. 'Speasr')
- Unterschiedliche Konventionen  
(z.B. 'Britney Jean Spears' vs. 'Spears, Britney J.')
- fehlerhafte Transformation vom Gesprochenen ins Geschriebene  
(z.B. 'Spears' vs. 'Speers')
- fehlerhafte Transformation vom Handschriftlichen ins Maschinelle (z.B. 'Spears' vs. 'Speans')
- Spitznamen, Künstlernamen oder Synonyme  
(z.B. 'Allan Stewart Konigsberg' vs. 'Heywood Allen' vs. 'Woody Allen')
- Abkürzungen (z.B. 'AStA' vs. 'Allgemeiner Studierendenausschuss')

# Herausforderungen: Genauigkeit

---

## Arten von Fehlerquellen:

- Tippfehler (z.B. 'Spears' vs. 'Speasr')
- Unterschiedliche Konventionen  
(z.B. 'Britney Jean Spears' vs. 'Spears, Britney J.')
- fehlerhafte Transformation vom Gesprochenen ins Geschriebene  
(z.B. 'Spears' vs. 'Speers')
- fehlerhafte Transformation vom Handschriftlichen ins Maschinelle (z.B. 'Spears' vs. 'Speans')
- Spitznamen, Künstlernamen oder Synonyme  
(z.B. 'Allan Stewart Konigsberg' vs. 'Heywood Allen' vs. 'Woody Allen')
- Abkürzungen (z.B. 'AStA' vs. 'Allgemeiner Studierendenausschuss')
- Unterschiedliche Maßeinheiten (z.B. '62 sec' vs. '1:02 h')

# Herausforderungen: Genauigkeit

---

# Herausforderungen: Genauigkeit

---

- **Lösungsansatz:**
  - Verwendung von einem Ähnlichkeitsmaß  $s(x, y) \in [0, 1]$  um die Ähnlichkeit zweier Stringwerte zu quantifizieren
  - Je größer  $s(x, y)$ , desto größer ist die Wahrscheinlichkeit das  $(x, y)$  ein Match ist
  - Klassifiziere  $(x, y)$  als ein Match, wenn  $s(x, y) \geq \theta$

# Herausforderungen: Genauigkeit

- **Lösungsansatz:**
  - Verwendung von einem Ähnlichkeitsmaß  $s(x, y) \in [0, 1]$  um die Ähnlichkeit zweier Stringwerte zu quantifizieren
  - Je größer  $s(x, y)$ , desto größer ist die Wahrscheinlichkeit das  $(x, y)$  ein Match ist
  - Klassifiziere  $(x, y)$  als ein Match, wenn  $s(x, y) \geq \theta$
  
- Oft werden Distanz- oder Kostenmaße verwendet
  - Je geringer die Distanz (Kosten), desto größer die Ähnlichkeit
  - Umrechnung für normiertes Distanzmaß  $d$ :  
$$s(x, y) = 1 - d(x, y)$$

# Herausforderungen: Skalierbarkeit

---

- Vergleichen jedes Stringwertes  $x \in A$  mit jedem Stringwert  $y \in B$  ist bei großen Mengen zu ineffizient (quadratische Komplexität)

# Herausforderungen: Skalierbarkeit

---

- Vergleichen jedes Stringwertes  $x \in A$  mit jedem Stringwert  $y \in B$  ist bei großen Mengen zu ineffizient (quadratische Komplexität)
- Verwendung einer Methode *FindCands* welche eine Menge an möglichen Matches  $C \subseteq B$  für jedes  $x \in A$  effizient bestimmt

# Herausforderungen: Skalierbarkeit

---

- Vergleichen jedes Stringwertes  $x \in A$  mit jedem Stringwert  $y \in B$  ist bei großen Mengen zu ineffizient (quadratische Komplexität)
- Verwendung einer Methode *FindCands* welche eine Menge an möglichen Matches  $C \subseteq B$  für jedes  $x \in A$  effizient bestimmt

Für jeden Stringwert  $x \in A$

- verwende *FindCands* um eine Menge  $C \subseteq B$  zu bestimmen
- für jeden Stringwert  $y \in C$ 
  - wenn  $s(x, y) \geq \theta$ , dann klassifiziere  $(x, y)$  als Match

# Agenda

---

- 1 Problemstellung
- 2 Genauigkeit
  - Mengenbasierte Ähnlichkeitsmaße
  - Sequenzbasierte Ähnlichkeitsmaße
  - Phonetische Ähnlichkeitsmaße
  - Semantische Ähnlichkeitsmaße
  - Hybride Ähnlichkeitsmaße
- 3 Skalierbarkeit

# Klassifizierung von Ähnlichkeitsmaßen

---

# Klassifizierung von Ähnlichkeitsmaßen

---

- Mengenbasiert / Tokenbasiert
  - Overlap Measure, Jaccard, Adamic, Kosinus mit TF/IDF

# Klassifizierung von Ähnlichkeitsmaßen

---

- Mengenbasiert / Tokenbasiert
  - Overlap Measure, Jaccard, Adamic, Kosinus mit TF/IDF
- Sequenzbasiert
  - Levenshtein, Affine Gap, Jaro

# Klassifizierung von Ähnlichkeitsmaßen

---

- **Mengenbasiert / Tokenbasiert**
  - Overlap Measure, Jaccard, Adamic, Kosinus mit TF/IDF
- **Sequenzbasiert**
  - Levenshtein, Affine Gap, Jaro
- **Phonetisch**
  - Soundex, Kölner Phonetik

# Klassifizierung von Ähnlichkeitsmaßen

---

- Mengenbasiert / Tokenbasiert
  - Overlap Measure, Jaccard, Adamic, Kosinus mit TF/IDF
- Sequenzbasiert
  - Levenshtein, Affine Gap, Jaro
- Phonetisch
  - Soundex, Kölner Phonetik
- Semantisch
  - Thesauri, Ontologien

# Klassifizierung von Ähnlichkeitsmaßen

---

- **Mengenbasiert / Tokenbasiert**
  - Overlap Measure, Jaccard, Adamic, Kosinus mit TF/IDF
- **Sequenzbasiert**
  - Levenshtein, Affine Gap, Jaro
- **Phonetisch**
  - Soundex, Kölner Phonetik
- **Semantisch**
  - Thesauri, Ontologien
- **Hybrid**
  - Extended/Generalized Jaccard, Kosinus mit Soft-TF/IDF, Monge-Elkan

# Agenda

---

- 1 Problemstellung
- 2 Genauigkeit
  - Mengenbasierte Ähnlichkeitsmaße
  - Sequenzbasierte Ähnlichkeitsmaße
  - Phonetische Ähnlichkeitsmaße
  - Semantische Ähnlichkeitsmaße
  - Hybride Ähnlichkeitsmaße
- 3 Skalierbarkeit

# Mengenbasierte Ähnlichkeitsmaße

---

# Mengenbasierte Ähnlichkeitsmaße

---

- betrachten Stringwerte als (Multi-)Mengen von Tokens

# Mengenbasierte Ähnlichkeitsmaße

---

- betrachten Stringwerte als (Multi-)Mengen von Tokens
- verwenden Mengeneigenschaften um Ähnlichkeit zu bestimmen

# Mengenbasierte Ähnlichkeitsmaße

---

- betrachten Stringwerte als (Multi-)Mengen von Tokens
- verwenden Mengeneigenschaften um Ähnlichkeit zu bestimmen
- verschiedene Möglichkeiten um einen Stringwert in Tokens zu zerlegen

# Mengenbasierte Ähnlichkeitsmaße

---

- betrachten Stringwerte als (Multi-)Mengen von Tokens
- verwenden Mengeneigenschaften um Ähnlichkeit zu bestimmen
- verschiedene Möglichkeiten um einen Stringwert in Tokens zu zerlegen
  - Wörter des Stringwertes
    - Trennung durch Leerzeichen oder Trennsymbole
    - Entfernung von Stopp-Wörtern wie 'der', 'und', oder 'von'
    - Bsp.: 'david smith' zu {'david', 'smith'}

# Mengenbasierte Ähnlichkeitsmaße

- betrachten Stringwerte als (Multi-)Mengen von Tokens
- verwenden Mengeneigenschaften um Ähnlichkeit zu bestimmen
- verschiedene Möglichkeiten um einen Stringwert in Tokens zu zerlegen
  - Wörter des Stringwertes
    - Trennung durch Leerzeichen oder Trennsymbole
    - Entfernung von Stopp-Wörtern wie 'der', 'und', oder 'von'
    - Bsp.: 'david smith' zu {'david', 'smith'}
  - q-grams (Substrings der Länge q)
    - Spezielles Zeichen # an Anfang und Ende des Stringwertes damit jeder Buchstabe gleich häufig vorkommt
    - Bsp.(q=3): 'david smith' zu {'##d', '#da', 'dav', 'avi', 'vid', 'id#', 'd##'}

# Overlap Measure & Common Neighbor Score

---

# Overlap Measure & Common Neighbor Score

---

- Input: Zwei Stringwerte  $x$  und  $y$ , Tokenfunktion  $tok$

# Overlap Measure & Common Neighbor Score

---

- Input: Zwei Stringwerte  $x$  und  $y$ , Tokenfunktion  $tok$
- Genierung der zwei Tokenmengen  $X = tok(x)$  und  $Y = tok(y)$

# Overlap Measure & Common Neighbor Score

---

- Input: Zwei Stringwerte  $x$  und  $y$ , Tokenfunktion  $tok$
- Genierung der zwei Tokenmengen  $X = tok(x)$  und  $Y = tok(y)$
- **Overlap Measure:**  $O(X, Y) = |X \cap Y|$   
(= Anzahl gemeinsamer Tokens)

# Overlap Measure & Common Neighbor Score

---

- Input: Zwei Stringwerte  $x$  und  $y$ , Tokenfunktion  $tok$
- Genierung der zwei Tokenmengen  $X = tok(x)$  und  $Y = tok(y)$
- **Overlap Measure:**  $O(X, Y) = |X \cap Y|$   
(= Anzahl gemeinsamer Tokens)
- Nachteil: nicht normalisiert (Werte daher schwer vergleichbar)

# Overlap Measure & Common Neighbor Score

- Input: Zwei Stringwerte  $x$  und  $y$ , Tokenfunktion  $tok$
  - Genierung der zwei Tokenmengen  $X = tok(x)$  und  $Y = tok(y)$
  - **Overlap Measure**:  $O(X, Y) = |X \cap Y|$   
(= Anzahl gemeinsamer Tokens)
  - Nachteil: nicht normalisiert (Werte daher schwer vergleichbar)
- ⇒ **Common Neighbor Score** [BG07]:  $CNS(X, Y) = O(X, Y)/k$
- Normalisierung des Overlap Measures mit geeignet großer Konstante  $k$

# Overlap Measure & Common Neighbor Score (Beispiel)

---

# Overlap Measure & Common Neighbor Score (Beispiel)

---

- $x = \text{'dave'}$ ,  $y = \text{'dav'}$
- Tokenbildung durch 2-grams

# Overlap Measure & Common Neighbor Score (Beispiel)

- $x = \text{'dave'}$ ,  $y = \text{'dav'}$
- Tokenbildung durch 2-grams

$$X = \{\text{'#d'}$$

$$Y = \{\text{'#d'}$$

# Overlap Measure & Common Neighbor Score (Beispiel)

- $x = \text{'dave'}$ ,  $y = \text{'dav'}$
- Tokenbildung durch 2-grams

$$X = \{\text{'#d'}$$

$$Y = \{\text{'#d'}$$

$$\Rightarrow X \cap Y = \{\text{'#d'}$$

# Overlap Measure & Common Neighbor Score (Beispiel)

- $x = \text{'dave'}$ ,  $y = \text{'dav'}$
- Tokenbildung durch 2-grams

$$X = \{\text{'#d'}$$

$$Y = \{\text{'#d'}$$

$$\Rightarrow X \cap Y = \{\text{'#d'}$$

$$\Rightarrow O(X, Y) = 3$$

# Overlap Measure & Common Neighbor Score (Beispiel)

- $x = \text{'dave'}$ ,  $y = \text{'dav'}$
- Tokenbildung durch 2-grams

$$X = \{\text{'#d'}$$

$$Y = \{\text{'#d'}$$

$$\Rightarrow X \cap Y = \{\text{'#d'}$$

$$\Rightarrow O(X, Y) = 3$$

$$\Rightarrow CNS(X, Y) = 0.3 \text{ mit } k = 10$$

# Jaccard Coefficient

---

# Jaccard Coefficient

---

- Normalisierung durch feste Konstante unrepräsentativ wenn Größe der insgesamt betrachteten Tokenmengen stark variiert

# Jaccard Coefficient

---

- Normalisierung durch feste Konstante unrepräsentativ wenn Größe der insgesamt betrachteten Tokenmengen stark variiert
- ⇒ Normalisierung durch Anzahl aller betrachteter Tokens

$$Jacc(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

(= Anteil gemeinsamer Tokens)

# Jaccard Coefficient (Beispiel)

---

# Jaccard Coefficient (Beispiel)

---

- $x='dave', y='dav'$
- Tokenbildung durch 2-grams

# Jaccard Coefficient (Beispiel)

---

- $x = \text{'dave'}$ ,  $y = \text{'dav'}$
- Tokenbildung durch 2-grams

$$X = \{\text{'#d'}$$

$$Y = \{\text{'#d'}$$

# Jaccard Coefficient (Beispiel)

- $x = \text{'dave'}$ ,  $y = \text{'dav'}$
- Tokenbildung durch 2-grams

$$X = \{\text{'#d'}$$

$$Y = \{\text{'#d'}$$

$$\Rightarrow X \cap Y = \{\text{'#d'}$$

$$X \cup Y = \{\text{'#d'}$$

# Jaccard Coefficient (Beispiel)

- $x = \text{'dave'}$ ,  $y = \text{'dav'}$
- Tokenbildung durch 2-grams

$$X = \{\text{'#d'}$$

$$Y = \{\text{'#d'}$$

$$\Rightarrow X \cap Y = \{\text{'#d'}$$

$$X \cup Y = \{\text{'#d'}$$

$$\Rightarrow \text{Jacc}(X, Y) = 3/6 = 0.5$$

# Adamic/Adar Ähnlichkeit

---

# Adamic/Adar Ähnlichkeit

---

- Einige Tokens sind bessere Unterscheidungsmerkmale als andere

# Adamic/Adar Ähnlichkeit

---

- Einige Tokens sind bessere Unterscheidungsmerkmale als andere
- ⇒ zwei Stringwerte sind umso ähnlicher je mehr seltene Token sie gemeinsam haben

# Adamic/Adar Ähnlichkeit

- Einige Tokens sind bessere Unterscheidungsmerkmale als andere
- ⇒ zwei Stringwerte sind umso ähnlicher je mehr seltene Token sie gemeinsam haben
- ⇒ Berücksichtigung der *uniqueness*  $u(t)$  eines Tokens  $t$

$$\text{Adar}(X, Y) = \frac{\sum_{t \in |X \cap Y|} u(t)}{\sum_{t \in |X \cup Y|} u(t)}$$

# Adamic/Adar Ähnlichkeit (Beispiel)

---

# Adamic/Adar Ähnlichkeit (Beispiel)

---

- $x = \text{'Apple Corporation, CA'}$ ,  $y = \text{'IBM Corporation, CA'}$ ,  
 $z = \text{'Apple Corp'}$
- Tokenbildung durch Trennzeichen

# Adamic/Adar Ähnlichkeit (Beispiel)

- $x = \text{'Apple Corporation, CA'}$ ,  $y = \text{'IBM Corporation, CA'}$ ,  
 $z = \text{'Apple Corp'}$
- Tokenbildung durch Trennzeichen
  - $X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\}$
  - $Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\}$
  - $Z = \{\text{'Apple'}, \text{'Corp'}\}$

# Adamic/Adar Ähnlichkeit (Beispiel)

- $x = \text{'Apple Corporation, CA'}$ ,  $y = \text{'IBM Corporation, CA'}$ ,  
 $z = \text{'Apple Corp'}$
- Tokenbildung durch Trennzeichen
$$X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\}$$
$$Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\}$$
$$Z = \{\text{'Apple'}, \text{'Corp'}\}$$
- Eigentlich  $x$  wahrscheinlicher ein Match mit  $z$  als mit  $y$ ,  
aber  $Jacc(X, Y) > Jacc(X, Z)$

# Adamic/Adar Ähnlichkeit (Beispiel)

- $x = \text{'Apple Corporation, CA'}$ ,  $y = \text{'IBM Corporation, CA'}$ ,  
 $z = \text{'Apple Corp'}$
- Tokenbildung durch Trennzeichen
$$X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\}$$
$$Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\}$$
$$Z = \{\text{'Apple'}, \text{'Corp'}\}$$
- Eigentlich  $x$  wahrscheinlicher ein Match mit  $z$  als mit  $y$ ,  
aber  $Jacc(X, Y) > Jacc(X, Z)$
- aber wenn  $u(\text{'Apple'}) \gg u(\text{'Corporation'})$ , dann gilt  
 $Adar(X, Z) > Adar(X, Y)$

# TF/IDF

---

# TF/IDF

---

- Verbreitetes Maß für *uniqueness* (Information Retrieval)

# TF/IDF

---

- Verbreitetes Maß für *uniqueness* (Information Retrieval)
- **term frequency**  $tf(t, d)$ :
  - Häufigkeit des Terms  $t$  in Dokument  $d$

# TF/IDF

---

- Verbreitetes Maß für *uniqueness* (Information Retrieval)
- **term frequency**  $tf(t, d)$ :
  - Häufigkeit des Terms  $t$  in Dokument  $d$
- **document frequency**  $df(t)$ :
  - Anteil aller Dokumente (Menge  $N$ ) die Term  $t$  beinhalten (Menge  $N_t$ ), d.h.  $df(t) = N_t/N$

# TF/IDF

---

- Verbreitetes Maß für *uniqueness* (Information Retrieval)
- **term frequency**  $tf(t, d)$ :
  - Häufigkeit des Terms  $t$  in Dokument  $d$
- **document frequency**  $df(t)$ :
  - Anteil aller Dokumente (Menge  $N$ ) die Term  $t$  beinhalten (Menge  $N_t$ ), d.h.  $df(t) = N_t/N$
- **inverse document frequency**  $idf(t)$ :  $idf(t) = N/N_t$

# TF/IDF

---

- Verbreitetes Maß für *uniqueness* (Information Retrieval)
- **term frequency**  $tf(t, d)$ :
  - Häufigkeit des Terms  $t$  in Dokument  $d$
- **document frequency**  $df(t)$ :
  - Anteil aller Dokumente (Menge  $N$ ) die Term  $t$  beinhalten (Menge  $N_t$ ), d.h.  $df(t) = N_t/N$
- **inverse document frequency**  $idf(t)$ :  $idf(t) = N/N_t$
- In String Matching: Term  $\simeq$  Token und Dokument  $\simeq$  Multimenge aller Tokens eines Stringwerts

# TF/IDF

- Verbreitetes Maß für *uniqueness* (Information Retrieval)
- **term frequency**  $tf(t, d)$ :
  - Häufigkeit des Terms  $t$  in Dokument  $d$
- **document frequency**  $df(t)$ :
  - Anteil aller Dokumente (Menge  $N$ ) die Term  $t$  beinhalten (Menge  $N_t$ ), d.h.  $df(t) = N_t/N$
- **inverse document frequency**  $idf(t)$ :  $idf(t) = N/N_t$
- In String Matching: Term  $\simeq$  Token und Dokument  $\simeq$  Multimenge aller Tokens eines Stringwerts
- verschiedene Varianten um  $tf$  und  $idf$  zu  $TF/IDF$  vereinen
- Gängige Varianten [DHI12]:
  - $TF/IDF(t, d) = tf(t, d) \times idf(t)$
  - $TF/IDF(t, d) = \log(tf(t, d) + 1) \times \log(idf(t))$

# TF/IDF: Beispiel

---

# TF/IDF: Beispiel

---

$X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\}$

$Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\}$

$Z = \{\text{'Apple'}, \text{'Corp'}\}$

# TF/IDF: Beispiel

---

$X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\}$

$Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\}$

$Z = \{\text{'Apple'}, \text{'Corp'}\}$

$N = 100$        $N_{\text{'Apple'}} = 5$

$N_{\text{'CA'}} = 25$        $N_{\text{'Corporation'}} = 40$

$N_{\text{'IBM'}} = 1$        $N_{\text{'Corp'}} = 20$

# TF/IDF: Beispiel

$X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\}$

$Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\}$

$Z = \{\text{'Apple'}, \text{'Corp'}\}$

$N = 100$

$N_{\text{'Apple'}} = 5$

$N_{\text{'CA'}} = 25$

$N_{\text{'Corporation'}} = 40$

$N_{\text{'IBM'}} = 1$

$N_{\text{'Corp'}} = 20$

<u>tf</u>	X	Y	Z
'Apple'	1	0	1
'CA'	1	1	0
'Corp'	0	0	1
'Corporation'	1	1	0
'IBM'	0	1	0

# TF/IDF: Beispiel

$X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\}$

$Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\}$

$Z = \{\text{'Apple'}, \text{'Corp'}\}$

$N = 100$        $N_{\text{'Apple'}} = 5$

$N_{\text{'CA'}} = 25$        $N_{\text{'Corporation'}} = 40$

$N_{\text{'IBM'}} = 1$        $N_{\text{'Corp'}} = 20$

<u>tf</u>	X	Y	Z
'Apple'	1	0	1
'CA'	1	1	0
'Corp'	0	0	1
'Corporation'	1	1	0
'IBM'	0	1	0

<u>idf</u>	
'Apple'	$100/5 = 20$
'CA'	$100/25 = 4$
'Corp'	$100/20 = 5$
'Corporation'	$100/40 = 2.5$
'IBM'	$100/1 = 100$

# Kosinus Ähnlichkeit

---

# Kosinus Ähnlichkeit

---

- Für jede Tokenmenge (Dokument)  $X$  wird ein mehrdimensionaler Feature-Vektor  $v_X$  erstellt

# Kosinus Ähnlichkeit

- Für jede Tokenmenge (Dokument)  $X$  wird ein mehrdimensionaler Feature-Vektor  $v_X$  erstellt
- Die Ähnlichkeit zwischen zwei Tokenmengen  $X$  und  $Y$  entspricht dem Kosinus des Winkels zwischen ihren Vektoren  $v_X$  und  $v_Y$

$$\begin{aligned} \text{CosSim}(X, Y) &= \frac{v_X \cdot v_Y}{\|v_X\| \times \|v_Y\|} \\ &= \frac{\sum_{i=1}^k v_X(i) \times v_Y(i)}{\sqrt{\sum_{i=1}^k v_X(i)^2} \times \sqrt{\sum_{i=1}^k v_Y(i)^2}} \end{aligned}$$

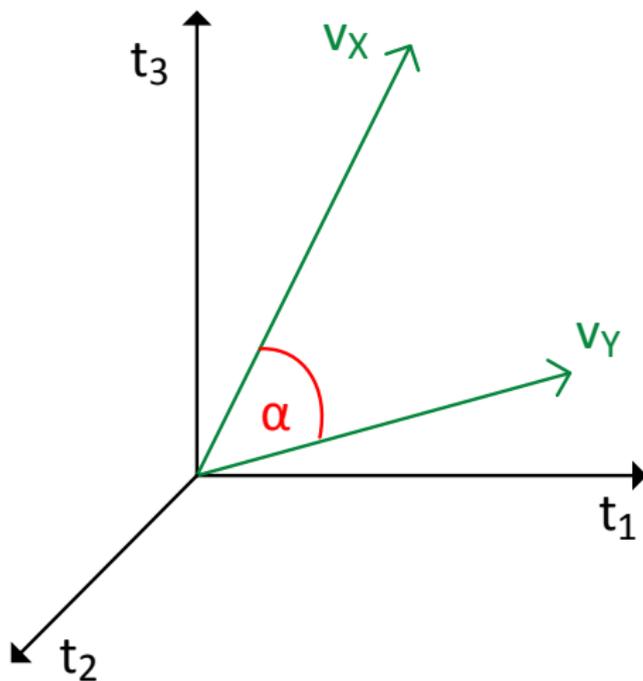
# Kosinus Ähnlichkeit

- Für jede Tokenmenge (Dokument)  $X$  wird ein mehrdimensionaler Feature-Vektor  $v_X$  erstellt
- Die Ähnlichkeit zwischen zwei Tokenmengen  $X$  und  $Y$  entspricht dem Kosinus des Winkels zwischen ihren Vektoren  $v_X$  und  $v_Y$

$$\begin{aligned} \text{CosSim}(X, Y) &= \frac{v_X \cdot v_Y}{\|v_X\| \times \|v_Y\|} \\ &= \frac{\sum_{i=1}^k v_X(i) \times v_Y(i)}{\sqrt{\sum_{i=1}^k v_X(i)^2} \times \sqrt{\sum_{i=1}^k v_Y(i)^2}} \end{aligned}$$

- Tokenmengen ähnlich  $\Rightarrow$  Winkel  $\rightarrow 0 \Rightarrow$  Kosinus  $\rightarrow 1$

# Kosinus Ähnlichkeit



# Kosinus Ähnlichkeit

---

# Kosinus Ähnlichkeit

---

- Eine Dimension per vorkommendem Token (Term)

# Kosinus Ähnlichkeit

---

- Eine Dimension per vorkommendem Token (Term)
- Jeder Vektor  $v_X$  hat einen Featurewert  $v_X(t)$  für jedes Token  $t$

# Kosinus Ähnlichkeit

---

- Eine Dimension per vorkommendem Token (Term)
- Jeder Vektor  $v_X$  hat einen Featurewert  $v_X(t)$  für jedes Token  $t$
- Der Featurewert  $v_X(t)$  wird gewöhnlich mit  $TF/IDF(t, X)$  berechnet

# Kosinus Ähnlichkeit

---

- Eine Dimension per vorkommendem Token (Term)
- Jeder Vektor  $v_X$  hat einen Featurewert  $v_X(t)$  für jedes Token  $t$
- Der Featurewert  $v_X(t)$  wird gewöhnlich mit  $TF/IDF(t, X)$  berechnet
- Je mehr sich die Vektoren in den einzelnen Featurewerten ähneln, desto geringer wird der Winkel zwischen ihnen

# Kosinus Ähnlichkeit (Beispiel)

$X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\}$

$Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\}$

$Z = \{\text{'Apple'}, \text{'Corp'}\}$

$N = 100$

$N_{\text{'Apple'}} = 5$

$N_{\text{'CA'}} = 25$

$N_{\text{'Corporation'}} = 40$

$N_{\text{'IBM'}} = 1$

$N_{\text{'Corp'}} = 20$

# Kosinus Ähnlichkeit (Beispiel)

$$\begin{array}{lll} X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\} & N = 100 & N_{\text{'Apple'}} = 5 \\ Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\} & N_{\text{'CA'}} = 25 & N_{\text{'Corporation'}} = 40 \\ Z = \{\text{'Apple'}, \text{'Corp'}\} & N_{\text{'IBM'}} = 1 & N_{\text{'Corp'}} = 20 \end{array}$$

- Gegeben  $TF/IDF(t, X) = tf(t, X) \times idf(t)$

# Kosinus Ähnlichkeit (Beispiel)

$$\begin{array}{lll} X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\} & N = 100 & N_{\text{'Apple'}} = 5 \\ Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\} & N_{\text{'CA'}} = 25 & N_{\text{'Corporation'}} = 40 \\ Z = \{\text{'Apple'}, \text{'Corp'}\} & N_{\text{'IBM'}} = 1 & N_{\text{'Corp'}} = 20 \end{array}$$

- Gegeben  $TF/IDF(t, X) = tf(t, X) \times idf(t)$

$$\Rightarrow v_X = \langle 20, 4, 0, 2.5, 0 \rangle$$

$$v_Y = \langle 0, 4, 0, 2.5, 100 \rangle$$

$$v_Z = \langle 20, 0, 5, 0, 0 \rangle$$

# Kosinus Ähnlichkeit (Beispiel)

$$\begin{array}{lll}
 X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\} & N = 100 & N_{\text{'Apple'}} = 5 \\
 Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\} & N_{\text{'CA'}} = 25 & N_{\text{'Corporation'}} = 40 \\
 Z = \{\text{'Apple'}, \text{'Corp'}\} & N_{\text{'IBM'}} = 1 & N_{\text{'Corp'}} = 20
 \end{array}$$

- Gegeben  $TF/IDF(t, X) = tf(t, X) \times idf(t)$

$$\Rightarrow v_X = \langle 20, 4, 0, 2.5, 0 \rangle$$

$$v_Y = \langle 0, 4, 0, 2.5, 100 \rangle$$

$$v_Z = \langle 20, 0, 5, 0, 0 \rangle$$

$$\Rightarrow \text{CosSim}(X, Y) = (22.25) / (\sqrt{422.25} \times \sqrt{10022.25}) = 0.011$$

# Kosinus Ähnlichkeit (Beispiel)

$$\begin{array}{lll}
 X = \{\text{'Apple'}, \text{'Corporation'}, \text{'CA'}\} & N = 100 & N_{\text{'Apple'}} = 5 \\
 Y = \{\text{'IBM'}, \text{'Corporation'}, \text{'CA'}\} & N_{\text{'CA'}} = 25 & N_{\text{'Corporation'}} = 40 \\
 Z = \{\text{'Apple'}, \text{'Corp'}\} & N_{\text{'IBM'}} = 1 & N_{\text{'Corp'}} = 20
 \end{array}$$

- Gegeben  $TF/IDF(t, X) = tf(t, X) \times idf(t)$

$$\Rightarrow v_X = \langle 20, 4, 0, 2.5, 0 \rangle$$

$$v_Y = \langle 0, 4, 0, 2.5, 100 \rangle$$

$$v_Z = \langle 20, 0, 5, 0, 0 \rangle$$

$$\Rightarrow \text{CosSim}(X, Y) = (22.25) / (\sqrt{422.25} \times \sqrt{10022.25}) = 0.011$$

$$\Rightarrow \text{CosSim}(X, Z) = (400) / (\sqrt{422.25} \times \sqrt{425}) = 0.944$$

# Agenda

---

- 1 Problemstellung
- 2 Genauigkeit
  - Mengenbasierte Ähnlichkeitsmaße
  - **Sequenzbasierte Ähnlichkeitsmaße**
  - Phonetische Ähnlichkeitsmaße
  - Semantische Ähnlichkeitsmaße
  - Hybride Ähnlichkeitsmaße
- 3 Skalierbarkeit

# Sequenzbasierte Ähnlichkeitsmaße

---

# Sequenzbasierte Ähnlichkeitsmaße

---

- betrachten Stringwerte als Sequenzen von Schriftzeichen

# Sequenzbasierte Ähnlichkeitsmaße

---

- betrachten Stringwerte als Sequenzen von Schriftzeichen
- Edit-Distanz berechnet Ähnlichkeit basierend auf den minimalen Kosten die notwendig sind um die erste Sequenz in die zweite zu transformieren

# Sequenzbasierte Ähnlichkeitsmaße

---

- betrachten Stringwerte als Sequenzen von Schriftzeichen
- Edit-Distanz berechnet Ähnlichkeit basierend auf den minimalen Kosten die notwendig sind um die erste Sequenz in die zweite zu transformieren
- Mehrere Varianten mit unterschiedlichen Mengen an erlaubten Operationen und deren Kosten

# Sequenzbasierte Ähnlichkeitsmaße

---

- betrachten Stringwerte als Sequenzen von Schriftzeichen
- Edit-Distanz berechnet Ähnlichkeit basierend auf den minimalen Kosten die notwendig sind um die erste Sequenz in die zweite zu transformieren
- Mehrere Varianten mit unterschiedlichen Mengen an erlaubten Operationen und deren Kosten
- Grundlegende Variante: **Levenshtein Distanz**

# Levenshtein Distanz/Ähnlichkeit

---

# Levenshtein Distanz/Ähnlichkeit

---

- Erlaubte Operationen: Einfügen, Löschen und Ersetzen

# Levenshtein Distanz/Ähnlichkeit

---

- Erlaubte Operationen: Einfügen, Löschen und Ersetzen
- Jede Operation hat die Kosten 1

# Levenshtein Distanz/Ähnlichkeit

---

- Erlaubte Operationen: Einfügen, Löschen und Ersetzen
- Jede Operation hat die Kosten 1
- Normalisierung durch die Länge der größeren Zeichenkette

$$\text{LevSim}(x, y) = 1 - \frac{\text{LevDst}(x, y)}{\max(|x|, |y|)}$$

# Levenshtein Distanz/Ähnlichkeit (Beispiel)

---

# Levenshtein Distanz/Ähnlichkeit (Beispiel)

---

- $x = \text{'hase'}$ ,  $y = \text{'rasen'}$

# Levenshtein Distanz/Ähnlichkeit (Beispiel)

---

- $x='hase'$ ,  $y='rasen'$
- **Schritt 1:** Ersetze 'h' durch 'r'

# Levenshtein Distanz/Ähnlichkeit (Beispiel)

---

- $x='hase'$ ,  $y='rasen'$
- **Schritt 1:** Ersetze 'h' durch 'r'
- **Schritt 2:** Füge ein 'n' ans Ende der Sequenz

# Levenshtein Distanz/Ähnlichkeit (Beispiel)

---

- $x='hase'$ ,  $y='rasen'$
  - **Schritt 1:** Ersetze 'h' durch 'r'
  - **Schritt 2:** Füge ein 'n' ans Ende der Sequenz
- ⇒ Gesamtkosten sind 2

# Levenshtein Distanz/Ähnlichkeit (Beispiel)

---

- $x = \text{'hase'}$ ,  $y = \text{'rasen'}$
  - **Schritt 1:** Ersetze 'h' durch 'r'
  - **Schritt 2:** Füge ein 'n' ans Ende der Sequenz
- ⇒ Gesamtkosten sind 2
- Da gilt:  $|x| = 4$  und  $|y| = 5$

# Levenshtein Distanz/Ähnlichkeit (Beispiel)

- $x = \text{'hase'}$ ,  $y = \text{'rasen'}$
  - **Schritt 1:** Ersetze 'h' durch 'r'
  - **Schritt 2:** Füge ein 'n' ans Ende der Sequenz
- ⇒ Gesamtkosten sind 2
- Da gilt:  $|x| = 4$  und  $|y| = 5$
- ⇒ Levenshtein Ähnlichkeit ist  $1 - 0.4 = 0.6$

# Levenshtein Distanz/Ähnlichkeit

---

# Levenshtein Distanz/Ähnlichkeit

---

- Berechnung durch dynamische Programmierung

# Levenshtein Distanz/Ähnlichkeit

- Berechnung durch dynamische Programmierung

- Edit-Distance-Matrix  
(Berechnung)

		<b>H</b>	<b>A</b>	<b>S</b>	<b>E</b>
	0	1	2	3	4
<b>R</b>	1	1	2		
<b>A</b>	2				
<b>S</b>	3				
<b>E</b>	4				
<b>N</b>	5				

$$D(i, 0) = i$$

$$D(0, j) = j$$

$$D(i, j) = \min \left\{ \begin{array}{l} D(i-1, j) + 1, \\ D(i, j-1) + 1, \\ D(i-1, j-1) + d(i, j) \end{array} \right\}$$

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

# Levenshtein Distanz/Ähnlichkeit

---

# Levenshtein Distanz/Ähnlichkeit

---

- Ermittlung der zugehörigen Transformation

# Levenshtein Distanz/Ähnlichkeit

- Ermittlung der zugehörigen Transformation

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 kleinstmöglichen Wert nach
  - ▶ links = DELETE
  - ▶ oben = INSERT
  - ▶ diagonal = MATCH oder REPLACE

# Levenshtein Distanz/Ähnlichkeit

---

# Levenshtein Distanz/Ähnlichkeit

---

- Beispiel: 'dva' vs. 'dave'

# Levenshtein Distanz/Ähnlichkeit

- Beispiel: 'dva' vs. 'dave'

	$y_0$	$y_1$	$y_2$	$y_3$	$y_4$
		<b>d</b>	<b>a</b>	<b>v</b>	<b>e</b>
$x_0$	0	1	2	3	4
$x_1$	<b>d</b>	1	0	1	
$x_2$	<b>v</b>	2			
$x_3$	<b>a</b>	3			

	$y_0$	$y_1$	$y_2$	$y_3$	$y_4$	
		<b>d</b>	<b>a</b>	<b>v</b>	<b>e</b>	
$x_0$	0	1	2	3	4	
$x_1$	<b>d</b>	1	0	1	2	3
$x_2$	<b>v</b>	2	1	1	1	2
$x_3$	<b>a</b>	3	2	1	2	2

$x = d - v a$

||||

$y = d a v e$

Substitute a with e  
Insert a (after d)

Quelle: Doan, Halevy and Ives. Principles of data Integration, 2012 [DHI12]

# Affine Gap Distanz

---

# Affine Gap Distanz

---

- Erweitert die Levenshtein Distanz um das Einfügen und Löschen von kompletten Substrings

# Affine Gap Distanz

---

- Erweitert die Levenshtein Distanz um das Einfügen und Löschen von kompletten Substrings
- Modifizieren von ganzen Blöcken ist günstiger als alle Zeichen einzeln zu modifizieren  
(Ziel: Robustheit gegen fehlende Mittelnamen)

# Affine Gap Distanz

---

- Erweitert die Levenshtein Distanz um das Einfügen und Löschen von kompletten Substrings
- Modifizieren von ganzen Blöcken ist günstiger als alle Zeichen einzeln zu modifizieren  
(Ziel: Robustheit gegen fehlende Mittelnamen)
- Kosten  $w_g$  für das Öffnen einer Lücke

# Affine Gap Distanz

---

- Erweitert die Levenshtein Distanz um das Einfügen und Löschen von kompletten Substrings
- Modifizieren von ganzen Blöcken ist günstiger als alle Zeichen einzeln zu modifizieren  
(Ziel: Robustheit gegen fehlende Mittelnamen)
- Kosten  $w_g$  für das Öffnen einer Lücke
- Kosten  $w_s$  für jedes weitere Zeichen in einer Lücke

# Affine Gap Distanz

---

- Erweitert die Levenshtein Distanz um das Einfügen und Löschen von kompletten Substrings
- Modifizieren von ganzen Blöcken ist günstiger als alle Zeichen einzeln zu modifizieren  
(Ziel: Robustheit gegen fehlende Mittelnamen)
- Kosten  $w_g$  für das Öffnen einer Lücke
- Kosten  $w_s$  für jedes weitere Zeichen in einer Lücke
- Kosten einer Lücke der Länge  $l$ :  $w(l) = w_g + (l - 1) \times w_s$

# Affine Gap Distanz

---

- Erweitert die Levenshtein Distanz um das Einfügen und Löschen von kompletten Substrings
- Modifizieren von ganzen Blöcken ist günstiger als alle Zeichen einzeln zu modifizieren  
(Ziel: Robustheit gegen fehlende Mittelnamen)
- Kosten  $w_g$  für das Öffnen einer Lücke
- Kosten  $w_s$  für jedes weitere Zeichen in einer Lücke
- Kosten einer Lücke der Länge  $l$ :  $w(l) = w_g + (l - 1) \times w_s$
- Kann mit Hilfe dynamischer Programmierung berechnet werden

# Affine Gap Distanz (Beispiel)

---

# Affine Gap Distanz (Beispiel)

---

- $x = \text{'Hans J Wurst'}$  und  $y = \text{'Hans Jürgen Wurst'}$

# Affine Gap Distanz (Beispiel)

---

- $x = \text{'Hans J Wurst'}$  und  $y = \text{'Hans Jürgen Wurst'}$
- Kosten für Öffnen einer Lücke:  $w_g = 1$

## Affine Gap Distanz (Beispiel)

---

- $x = \text{'Hans J Wurst'}$  und  $y = \text{'Hans Jürgen Wurst'}$
- Kosten für Öffnen einer Lücke:  $w_g = 1$
- Kosten für Weiterführen einer Lücke:  $w_s = 0.1$

## Affine Gap Distanz (Beispiel)

---

- $x = \text{'Hans J Wurst'}$  und  $y = \text{'Hans Jürgen Wurst'}$
- Kosten für Öffnen einer Lücke:  $w_g = 1$
- Kosten für Weiterführen einer Lücke:  $w_s = 0.1$
- Die Lücke umfasst den substring 'ürgen'

## Affine Gap Distanz (Beispiel)

---

- $x = \text{'Hans J Wurst'}$  und  $y = \text{'Hans Jürgen Wurst'}$
- Kosten für Öffnen einer Lücke:  $w_g = 1$
- Kosten für Weiterführen einer Lücke:  $w_s = 0.1$
- Die Lücke umfasst den substring 'ürgen'
- Gesamtkosten der Lücke:  $w(l) = 1 + 4 \times 0.1 = 1.4$

## Affine Gap Distanz (Beispiel)

---

- $x = \text{'Hans J Wurst'}$  und  $y = \text{'Hans Jürgen Wurst'}$
- Kosten für Öffnen einer Lücke:  $w_g = 1$
- Kosten für Weiterführen einer Lücke:  $w_s = 0.1$
- Die Lücke umfasst den substring 'ürgen'
- Gesamtkosten der Lücke:  $w(l) = 1 + 4 \times 0.1 = 1.4$
- Die Levenshtein Distanz zwischen  $x$  und  $y$  beträgt hingegen 5

# Jaro Ähnlichkeit

---

# Jaro Ähnlichkeit

---

- Konstruiert zum Vergleich kurzer Stringwerte wie Namen

# Jaro Ähnlichkeit

---

- Konstruiert zum Vergleich kurzer Stringwerte wie Namen
- Berechnung in 3 Schritten:

# Jaro Ähnlichkeit

---

- Konstruiert zum Vergleich kurzer Stringwerte wie Namen
- Berechnung in 3 Schritten:
  - Finde allen gemeinsamen Buchstaben  $x_i$  und  $y_j$  ( $x_i = y_j$ ) so dass  $|i - j| \leq \min(|x|, |y|)/2$

# Jaro Ähnlichkeit

- Konstruiert zum Vergleich kurzer Stringwerte wie Namen
- Berechnung in 3 Schritten:
  - Finde allen gemeinsamen Buchstaben  $x_i$  und  $y_j$  ( $x_i = y_j$ ) so dass  $|i - j| \leq \min(|x|, |y|)/2$
  - Vergleiche den  $i$ ten gemeinsamen Buchstaben von  $x$  mit dem  $i$ ten gemeinsamen Buchstaben von  $y$ . Sind sie verschieden, liegt eine Transposition vor.

# Jaro Ähnlichkeit

- Konstruiert zum Vergleich kurzer Stringwerte wie Namen
- Berechnung in 3 Schritten:
  - Finde allen gemeinsamen Buchstaben  $x_i$  und  $y_j$  ( $x_i = y_j$ ) so dass  $|i - j| \leq \min(|x|, |y|)/2$
  - Vergleiche den  $i$ ten gemeinsamen Buchstaben von  $x$  mit dem  $i$ ten gemeinsamen Buchstaben von  $y$ . Sind sie verschieden, liegt eine Transposition vor.
  - Sei  $c$  die Anzahl der gemeinsamen Buchstaben und sei  $t$  die Anzahl der Transpositionen, die Jaro Ähnlichkeit zwischen  $x$  und  $y$  ist definiert als:

$$Jaro(x, y) = \frac{1}{3} \left( \frac{c}{|x|} + \frac{c}{|y|} + \frac{c - t/2}{c} \right)$$

# Jaro Ähnlichkeit (Beispiel)

---

# Jaro Ähnlichkeit (Beispiel)

---

- $x = \text{'jon'}$  und  $y = \text{'ojhn'}$

# Jaro Ähnlichkeit (Beispiel)

---

- $x = \text{'jon'}$  und  $y = \text{'ojhn'}$
- Die maximale Distanz zwischen zwei gemeinsamen Buchstaben beträgt  $\lfloor \min(3, 4)/2 \rfloor = \lfloor 1.5 \rfloor = 1$

# Jaro Ähnlichkeit (Beispiel)

---

- $x = \text{'jon'}$  und  $y = \text{'ojhn'}$
- Die maximale Distanz zwischen zwei gemeinsamen Buchstaben beträgt  $\lfloor \min(3, 4)/2 \rfloor = \lfloor 1.5 \rfloor = 1$
- Die Sequenz gemeinsamer Buchstaben in  $x$  ist 'jon'

# Jaro Ähnlichkeit (Beispiel)

---

- $x = \text{'jon'}$  und  $y = \text{'ojhn'}$
- Die maximale Distanz zwischen zwei gemeinsamen Buchstaben beträgt  $\lfloor \min(3, 4)/2 \rfloor = \lfloor 1.5 \rfloor = 1$
- Die Sequenz gemeinsamer Buchstaben in  $x$  ist 'jon'
- Die Sequenz gemeinsamer Buchstaben in  $y$  ist 'ojn'

# Jaro Ähnlichkeit (Beispiel)

---

- $x = \text{'jon'}$  und  $y = \text{'ojhn'}$
- Die maximale Distanz zwischen zwei gemeinsamen Buchstaben beträgt  $\lfloor \min(3, 4)/2 \rfloor = \lfloor 1.5 \rfloor = 1$
- Die Sequenz gemeinsamer Buchstaben in  $x$  ist 'jon'
- Die Sequenz gemeinsamer Buchstaben in  $y$  ist 'ojn'
- Anzahl an gemeinsamen Buchstaben ist:  $c = 3$

# Jaro Ähnlichkeit (Beispiel)

---

- $x = \text{'jon'}$  und  $y = \text{'ojhn'}$
- Die maximale Distanz zwischen zwei gemeinsamen Buchstaben beträgt  $\lfloor \min(3, 4)/2 \rfloor = \lfloor 1.5 \rfloor = 1$
- Die Sequenz gemeinsamer Buchstaben in  $x$  ist 'jon'
- Die Sequenz gemeinsamer Buchstaben in  $y$  ist 'ojn'
- Anzahl an gemeinsamen Buchstaben ist:  $c = 3$
- Anzahl an Transpositionen ist:  $t = 2$

# Jaro Ähnlichkeit (Beispiel)

- $x = \text{'jon'}$  und  $y = \text{'ojhn'}$
- Die maximale Distanz zwischen zwei gemeinsamen Buchstaben beträgt  $\lfloor \min(3, 4)/2 \rfloor = \lfloor 1.5 \rfloor = 1$
- Die Sequenz gemeinsamer Buchstaben in  $x$  ist 'jon'
- Die Sequenz gemeinsamer Buchstaben in  $y$  ist 'ojn'
- Anzahl an gemeinsamen Buchstaben ist:  $c = 3$
- Anzahl an Transpositionen ist:  $t = 2$

$$\Rightarrow \text{Jaro}(x, y) = \frac{1}{3} \left( \frac{3}{3} + \frac{3}{4} + \frac{3-1}{3} \right) = 0.81$$

# Agenda

---

- 1 Problemstellung
- 2 Genauigkeit
  - Mengenbasierte Ähnlichkeitsmaße
  - Sequenzbasierte Ähnlichkeitsmaße
  - **Phonetische Ähnlichkeitsmaße**
  - Semantische Ähnlichkeitsmaße
  - Hybride Ähnlichkeitsmaße
- 3 Skalierbarkeit

# Phonetische Ähnlichkeit

---

# Phonetische Ähnlichkeit

---

- Ähnlichkeit anhand der Wortklänge anstatt der Syntax (sinnvoll wenn Daten telefonisch übermittelt werden)

# Phonetische Ähnlichkeit

---

- Ähnlichkeit anhand der Wortklänge anstatt der Syntax (sinnvoll wenn Daten telefonisch übermittelt werden)
- Verfahren sind zumeist sehr sprachabhängig

# Phonetische Ähnlichkeit

---

- Ähnlichkeit anhand der Wortklänge anstatt der Syntax (sinnvoll wenn Daten telefonisch übermittelt werden)
- Verfahren sind zumeist sehr sprachabhängig
- Verwendung oft auf Personennamen beschränkt  
z.B. 'Meier' vs. 'Mayer', 'Schmidt' vs. 'Schmitt'

# Phonetische Ähnlichkeit

---

- Ähnlichkeit anhand der Wortklänge anstatt der Syntax (sinnvoll wenn Daten telefonisch übermittelt werden)
- Verfahren sind zumeist sehr sprachabhängig
- Verwendung oft auf Personennamen beschränkt z.B. 'Meier' vs. 'Mayer', 'Schmidt' vs. 'Schmitt'
- Stringwerte werden in Codes transformiert

# Phonetische Ähnlichkeit

- Ähnlichkeit anhand der Wortklänge anstatt der Syntax (sinnvoll wenn Daten telefonisch übermittelt werden)
- Verfahren sind zumeist sehr sprachabhängig
- Verwendung oft auf Personennamen beschränkt z.B. 'Meier' vs. 'Mayer', 'Schmidt' vs. 'Schmitt'
- Stringwerte werden in Codes transformiert
- Phonetische Ähnlichkeit ergibt sich aus der Ähnlichkeit der Codes. Im einfachsten Fall:
  - identische Codes  $\Rightarrow$  Ähnlichkeit = 1
  - unterschiedliche Codes  $\Rightarrow$  Ähnlichkeit = 0

# Phonetische Ähnlichkeit

- Ähnlichkeit anhand der Wortklänge anstatt der Syntax (sinnvoll wenn Daten telefonisch übermittelt werden)
- Verfahren sind zumeist sehr sprachabhängig
- Verwendung oft auf Personennamen beschränkt z.B. 'Meier' vs. 'Mayer', 'Schmidt' vs. 'Schmitt'
- Stringwerte werden in Codes transformiert
- Phonetische Ähnlichkeit ergibt sich aus der Ähnlichkeit der Codes. Im einfachsten Fall:
  - identische Codes  $\Rightarrow$  Ähnlichkeit = 1
  - unterschiedliche Codes  $\Rightarrow$  Ähnlichkeit = 0
- Häufig verwendetes Verfahren ist [Soundex](#) [Rus18, Rus22]

# Soundex

---

# Soundex

---

1. Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt

# Soundex

---

1. Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt
2. Mit Ausnahme des ersten Buchstaben werden alle Nichtvokale durch folgende Ziffern ersetzt

b, f, p, v	→	1	l	→	4
c, g, j, k, q, s, x, z	→	2	m, n	→	5
d, t	→	3	r	→	6

# Soundex

---

1. Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt
2. Mit Ausnahme des ersten Buchstaben werden alle Nichtvokale durch folgende Ziffern ersetzt

b, f, p, v	→	1	l	→	4
c, g, j, k, q, s, x, z	→	2	m, n	→	5
d, t	→	3	r	→	6

3. Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes ersetzt (z.B. '1001' wird zu '101')

# Soundex

---

1. Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt
2. Mit Ausnahme des ersten Buchstaben werden alle Nichtvokale durch folgende Ziffern ersetzt

b, f, p, v	→	1	l	→	4
c, g, j, k, q, s, x, z	→	2	m, n	→	5
d, t	→	3	r	→	6

3. Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes ersetzt (z.B. '1001' wird zu '101')
4. Mit Ausnahme des ersten Buchstaben werden alle Nicht-Ziffern entfernt

# Soundex

1. Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt
2. Mit Ausnahme des ersten Buchstaben werden alle Nichtvokale durch folgende Ziffern ersetzt

b, f, p, v	→	1	l	→	4
c, g, j, k, q, s, x, z	→	2	m, n	→	5
d, t	→	3	r	→	6

3. Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes ersetzt (z.B. '1001' wird zu '101')
4. Mit Ausnahme des ersten Buchstaben werden alle Nicht-Ziffern entfernt
5. Der Code wird auf die Länge vier beschränkt (notfalls Auffüllen mit der Ziffer '0')

# Soundex: Beispiele

---

	<b>'Tukker'</b>	<b>'Tacker'</b>	<b>'William'</b>	<b>'Bill'</b>

# Soundex: Beispiele

	<b>'Tukker'</b>	<b>'Tacker'</b>	<b>'William'</b>	<b>'Bill'</b>
Schritt 1	'Tukker'	'Tacker'	'William'	'Bill'

- Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt

# Soundex: Beispiele

	<b>'Tukker'</b>	<b>'Tacker'</b>	<b>'William'</b>	<b>'Bill'</b>
Schritt 1	'Tukker'	'Tacker'	'William'	'Bill'
Schritt 2	'Tu22e6'	'Ta22e6'	'Wi44ia5'	'Bi44'

- Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt
- Mit Ausnahme des ersten Buchstaben werden alle Nichtvokale durch Ziffern ersetzt (k→2, r→6, l→4, m→5)

# Soundex: Beispiele

	'Tukker'	'Tacker'	'William'	'Bill'
Schritt 1	'Tukker'	'Tacker'	'William'	'Bill'
Schritt 2	'Tu22e6'	'Ta22e6'	'Wi44ia5'	'Bi44'
Schritt 3	'Tu2e6'	'Ta2e6'	'Wi4ia5'	'Bi4'

- Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt
- Mit Ausnahme des ersten Buchstaben werden alle Nichtvokale durch Ziffern ersetzt ( $k \rightarrow 2$ ,  $r \rightarrow 6$ ,  $l \rightarrow 4$ ,  $m \rightarrow 5$ )
- Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes Vorkommnis ersetzt

# Soundex: Beispiele

	'Tukker'	'Tacker'	'William'	'Bill'
Schritt 1	'Tukker'	'Tacker'	'William'	'Bill'
Schritt 2	'Tu22e6'	'Ta22e6'	'Wi44ia5'	'Bi44'
Schritt 3	'Tu2e6'	'Ta2e6'	'Wi4ia5'	'Bi4'
Schritt 4	'T26'	'T26'	'W45'	'B4'

- Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt
- Mit Ausnahme des ersten Buchstaben werden alle Nichtvokale durch Ziffern ersetzt ( $k \rightarrow 2$ ,  $r \rightarrow 6$ ,  $l \rightarrow 4$ ,  $m \rightarrow 5$ )
- Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes Vorkommnis ersetzt
- Mit Ausnahme des ersten Buchstaben werden alle Nicht-Ziffern entfernt

# Soundex: Beispiele

	'Tukker'	'Tacker'	'William'	'Bill'
Schritt 1	'Tukker'	'Tacker'	'William'	'Bill'
Schritt 2	'Tu22e6'	'Ta22e6'	'Wi44ia5'	'Bi44'
Schritt 3	'Tu2e6'	'Ta2e6'	'Wi4ia5'	'Bi4'
Schritt 4	'T26'	'T26'	'W45'	'B4'
Schritt 5	<b>'T260'</b>	<b>'T260'</b>	<b>'W450'</b>	<b>'B400'</b>

- Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse von 'h' und 'w' entfernt
- Mit Ausnahme des ersten Buchstaben werden alle Nichtvokale durch Ziffern ersetzt ( $k \rightarrow 2$ ,  $r \rightarrow 6$ ,  $l \rightarrow 4$ ,  $m \rightarrow 5$ )
- Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes Vorkommnis ersetzt
- Mit Ausnahme des ersten Buchstaben werden alle Nicht-Ziffern entfernt
- Der Code wird auf die Länge vier beschränkt (Auffüllen mit '0')

# Soundex (alternative Definition)

---

# Soundex (alternative Definition)

---

1. Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse der Buchstaben 'a', 'e', 'i', 'o', 'u', 'y', 'h' und 'w' entfernt

# Soundex (alternative Definition)

---

1. Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse der Buchstaben 'a', 'e', 'i', 'o', 'u', 'y', 'h' und 'w' entfernt
2. Mit Ausnahme des ersten Buchstaben werden alle verbliebende Buchstaben durch Ziffern ersetzt

b, f, p, v	→	1	l	→	4
c, g, j, k, q, s, x, z	→	2	m, n	→	5
d, t	→	3	r	→	6

# Soundex (alternative Definition)

1. Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse der Buchstaben 'a', 'e', 'i', 'o', 'u', 'y', 'h' und 'w' entfernt
2. Mit Ausnahme des ersten Buchstaben werden alle verbliebende Buchstaben durch Ziffern ersetzt

b, f, p, v	→	1	l	→	4
c, g, j, k, q, s, x, z	→	2	m, n	→	5
d, t	→	3	r	→	6

3. Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes ersetzt (z.B. '1001' wird zu '101'), wenn die den Ziffern zugehörigen Buchstaben im ursprünglichen String nicht durch einen Vokal (inkl. 'y') getrennt waren

# Soundex (alternative Definition)

1. Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse der Buchstaben 'a', 'e', 'i', 'o', 'u', 'y', 'h' und 'w' entfernt
2. Mit Ausnahme des ersten Buchstaben werden alle verbliebende Buchstaben durch Ziffern ersetzt

b, f, p, v	→	1	l	→	4
c, g, j, k, q, s, x, z	→	2	m, n	→	5
d, t	→	3	r	→	6

3. Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes ersetzt (z.B. '1001' wird zu '101'), wenn die den Ziffern zugehörigen Buchstaben im ursprünglichen String nicht durch einen Vokal (inkl. 'y') getrennt waren
4. Der Code wird auf die Länge vier beschränkt (notfalls Auffüllen mit der Ziffer '0')

# Soundex: Beispiele (alternative Definition)

---

	<b>'Tukker'</b>	<b>'Tacker'</b>	<b>'William'</b>	<b>'Bill'</b>

## Soundex: Beispiele (alternative Definition)

	<b>'Tukker'</b>	<b>'Tacker'</b>	<b>'William'</b>	<b>'Bill'</b>
Schritt 1	'Tkkr'	'Tckr'	'Wllm'	'Bll'

- Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse der Buchstaben 'a', 'e', 'i', 'o', 'u', 'y', 'h', und 'w' entfernt

## Soundex: Beispiele (alternative Definition)

	<b>'Tukker'</b>	<b>'Tacker'</b>	<b>'William'</b>	<b>'Bill'</b>
Schritt 1	'Tkk r'	'Tck r'	'Wllm'	'Bll'
Schritt 2	'T226'	'T226'	'W445'	'B44'

- Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse der Buchstaben 'a', 'e', 'i', 'o', 'u', 'y', 'h', und 'w' entfernt
- Mit Ausnahme des ersten Buchstaben werden alle verbliebende Buchstaben durch Ziffern ersetzt (k→2, r→6, l→4, m→5)

## Soundex: Beispiele (alternative Definition)

	<b>'Tukker'</b>	<b>'Tacker'</b>	<b>'William'</b>	<b>'Bill'</b>
Schritt 1	'Tkkr'	'Tckr'	'Wllm'	'Bll'
Schritt 2	'T226'	'T226'	'W445'	'B44'
Schritt 3	'T26'	'T26'	'W45'	'B4'

- Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse der Buchstaben 'a', 'e', 'i', 'o', 'u', 'y', 'h', und 'w' entfernt
- Mit Ausnahme des ersten Buchstaben werden alle verbliebende Buchstaben durch Ziffern ersetzt (k→2, r→6, l→4, m→5)
- Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes Vorkommnis ersetzt

## Soundex: Beispiele (alternative Definition)

	'Tukker'	'Tacker'	'William'	'Bill'
Schritt 1	'Tkkr'	'Tckr'	'Wllm'	'Bll'
Schritt 2	'T226'	'T226'	'W445'	'B44'
Schritt 3	'T26'	'T26'	'W45'	'B4'
Schritt 4	<b>'T260'</b>	<b>'T260'</b>	<b>'W450'</b>	<b>'B400'</b>

- Mit Ausnahme des ersten Buchstaben werden alle Vorkommnisse der Buchstaben 'a', 'e', 'i', 'o', 'u', 'y', 'h', und 'w' entfernt
- Mit Ausnahme des ersten Buchstaben werden alle verbliebende Buchstaben durch Ziffern ersetzt (k→2, r→6, l→4, m→5)
- Alle aufeinanderfolgenden Vorkommnisse der gleichen Ziffer werden durch ein einzelnes Vorkommnis ersetzt
- Der Code wird auf die Länge vier beschränkt (Auffüllen mit '0')

# Soundex

---

# Soundex

---

- Gleiche Codes resultieren nur wenn der erste Buchstabe gleich ist (daher Probleme mit 'Bill' vs. 'William' oder 'Jawornicki' vs. 'Yavornitzky')

# Soundex

---

- Gleiche Codes resultieren nur wenn der erste Buchstabe gleich ist (daher Probleme mit 'Bill' vs. 'William' oder 'Jawornicki' vs. 'Yavornitzky')
- konzipiert für kaukasische Namen, funktioniert aber auch zufriedenstellend für Datensätze mit Namen verschiedener Herkünfte

# Soundex

---

- Gleiche Codes resultieren nur wenn der erste Buchstabe gleich ist (daher Probleme mit 'Bill' vs. 'William' oder 'Jawornicki' vs. 'Yavornitzky')
- konzipiert für kaukasische Namen, funktioniert aber auch zufriedenstellend für Datensätze mit Namen verschiedener Herkünfte
- funktioniert allerdings weniger gut für asiatische Namen

# Soundex

---

- Gleiche Codes resultieren nur wenn der erste Buchstabe gleich ist (daher Probleme mit 'Bill' vs. 'William' oder 'Jawornicki' vs. 'Yavornitzky')
- konzipiert für kaukasische Namen, funktioniert aber auch zufriedenstellend für Datensätze mit Namen verschiedener Herkünfte
- funktioniert allerdings weniger gut für asiatische Namen
- weitere phonetische Maße vorhanden (z.B. [Kölner Phonetik](#) [Pos69], [Metaphone](#) [Phi90] oder [NYSIIS Algorithmus](#) [Taf70])

# Agenda

---

- 1 Problemstellung
- 2 Genauigkeit
  - Mengenbasierte Ähnlichkeitsmaße
  - Sequenzbasierte Ähnlichkeitsmaße
  - Phonetische Ähnlichkeitsmaße
  - **Semantische Ähnlichkeitsmaße**
  - Hybride Ähnlichkeitsmaße
- 3 Skalierbarkeit

# Semantische Ähnlichkeit

---

# Semantische Ähnlichkeit

---

- Erkennen von semantischen Beziehungen zwischen Begriffen

# Semantische Ähnlichkeit

---

- Erkennen von semantischen Beziehungen zwischen Begriffen
- Verwendung von **Ontologien**

# Semantische Ähnlichkeit

---

- Erkennen von semantischen Beziehungen zwischen Begriffen
- Verwendung von **Ontologien**
- Spektrum reicht von formalen Regeln bis hin zu informalen Beschreibungen

# Semantische Ähnlichkeit

---

- Erkennen von semantischen Beziehungen zwischen Begriffen
- Verwendung von **Ontologien**
- Spektrum reicht von formalen Regeln bis hin zu informalen Beschreibungen
- Weitverbreitete informale Ontologien sind **Thesauri**

# Semantische Ähnlichkeit

---

- Erkennen von semantischen Beziehungen zwischen Begriffen
- Verwendung von **Ontologien**
- Spektrum reicht von formalen Regeln bis hin zu informalen Beschreibungen
- Weitverbreitete informale Ontologien sind **Thesauri**
- Thesauri eignen sich um Synonymen und Antonymen, sowie um Hyponymen und Hyperonymen zu erkennen

# Thesauri: Beispiel von Synonymen

---

- Vornamen die mit den Vornamen 'Alexander' verwandt sind\*

<b>name</b>	<b>LevSim</b>
Alejandro	0.67
Aleksandr	0.67
Alessandro	0.6
Alexandros	0.8
Alexis	0.44
Leszek	0.33
Sascha	0.11
Xander	0.67

\*Quelle: <http://www.onomastik.com/>

# Thesauri: Beispiel von Synonymen

- Vornamen die mit den Vornamen 'Alexander' verwandt sind\*

<b>name</b>	<b>LevSim</b>
Alejandro	0.67
Aleksandr	0.67
Alessandro	0.6
Alexandros	0.8
Alexis	0.44
Leszek	0.33
Sascha	0.11
Xander	0.67

- Andere Beispiele: 'Bill' vs. 'William', 'James' vs. 'Jim', 'Australien' vs. 'Down Under'

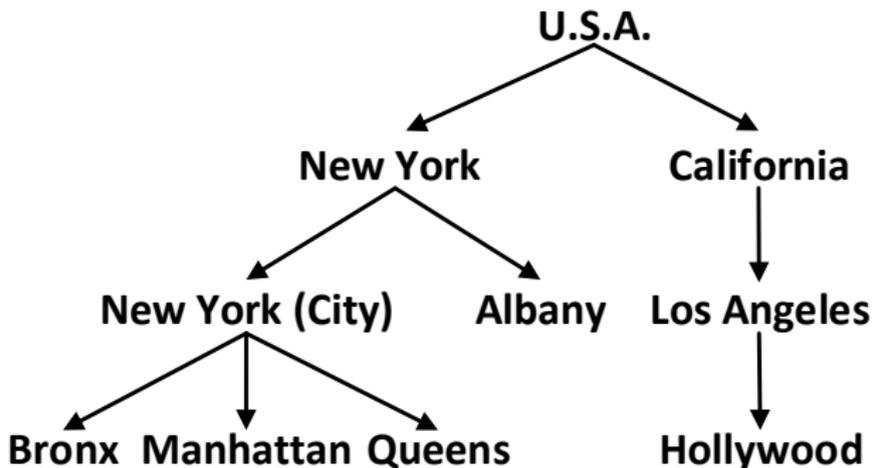
\*Quelle: <http://www.onomastik.com/>

# Thesauri: Beispiel von Hyponymen und Hyperonymen

---

# Thesauri: Beispiel von Hyponymen und Hyperonymen

- Hierarchie von Stadtteilen, Städten und Staaten in den U.S.A.



# Agenda

---

- 1 Problemstellung
- 2 Genauigkeit
  - Mengenbasierte Ähnlichkeitsmaße
  - Sequenzbasierte Ähnlichkeitsmaße
  - Phonetische Ähnlichkeitsmaße
  - Semantische Ähnlichkeitsmaße
  - **Hybride Ähnlichkeitsmaße**
- 3 Skalierbarkeit

# Hybride Ähnlichkeitsmaße

---

# Hybride Ähnlichkeitsmaße

---

- Nachteil Mengenbasiert: Typos werden nicht kompensiert

# Hybride Ähnlichkeitsmaße

---

- Nachteil Mengenbasiert: Typos werden nicht kompensiert
- Nachteil Sequenzbasiert: Wortvertauschungen werden nicht kompensiert

# Hybride Ähnlichkeitsmaße

---

- Nachteil Mengenbasiert: Typos werden nicht kompensiert
- Nachteil Sequenzbasiert: Wortvertauschungen werden nicht kompensiert
- Idee: Beide Konzepte in einem Maß zu vereinen

# Hybride Ähnlichkeitsmaße

---

- Nachteil Mengenbasiert: Typos werden nicht kompensiert
- Nachteil Sequenzbasiert: Wortvertauschungen werden nicht kompensiert
- Idee: Beide Konzepte in einem Maß zu vereinen
- Zerlegung eines Stringwertes  $x$  in eine Multimenge von Token  
 $X = tok(x)$

# Hybride Ähnlichkeitsmaße

- Nachteil Mengenbasiert: Typos werden nicht kompensiert
- Nachteil Sequenzbasiert: Wortvertauschungen werden nicht kompensiert
- Idee: Beide Konzepte in einem Maß zu vereinen
- Zerlegung eines Stringwertes  $x$  in eine Multimenge von Token  
 $X = tok(x)$
- Verwendung eines Sequenzbasiertes Maßes *TokenSim* um die Ähnlichkeit zwischen Token zu bestimmen

# Extended Jaccard Coefficient [NH10]

---

# Extended Jaccard Coefficient [NH10]

---

- Ähnlichkeit zwischen zwei Stringwerten  $x$  und  $y$  als Anteil ähnlicher Token in  $X = tok(x)$  und  $Y = tok(y)$

## Extended Jaccard Coefficient [NH10]

---

- Ähnlichkeit zwischen zwei Stringwerten  $x$  und  $y$  als Anteil ähnlicher Token in  $X = tok(x)$  und  $Y = tok(y)$
- Menge an ähnlichen Tokenpaaren:

$$shared(X, Y) = \{(t_1, t_2) \mid t_1 \in X, t_2 \in Y, TokenSim(t_1, t_2) \geq \theta\}$$

## Extended Jaccard Coefficient [NH10]

- Ähnlichkeit zwischen zwei Stringwerten  $x$  und  $y$  als Anteil ähnlicher Token in  $X = tok(x)$  und  $Y = tok(y)$
- Menge an ähnlichen Tokenpaaren:

$$shared(X, Y) = \{(t_1, t_2) \mid t_1 \in X, t_2 \in Y, TokenSim(t_1, t_2) \geq \theta\}$$

- Menge an Token von  $X$  ohne ähnliches Token in  $Y$ :

$$unique(X) = \{t_1 \mid t_1 \in X, \nexists t_2 \in Y, TokenSim(t_1, t_2) \geq \theta\}$$

## Extended Jaccard Coefficient [NH10]

- Ähnlichkeit zwischen zwei Stringwerten  $x$  und  $y$  als Anteil ähnlicher Token in  $X = tok(x)$  und  $Y = tok(y)$
- Menge an ähnlichen Tokenpaaren:

$$shared(X, Y) = \{(t_1, t_2) \mid t_1 \in X, t_2 \in Y, TokenSim(t_1, t_2) \geq \theta\}$$

- Menge an Token von  $X$  ohne ähnliches Token in  $Y$ :

$$unique(X) = \{t_1 \mid t_1 \in X, \nexists t_2 \in Y, TokenSim(t_1, t_2) \geq \theta\}$$

- Extended Jaccard Coefficient:

$$ExtJacc(X, Y) = \frac{|shared(X, Y)|}{|shared(X, Y)| + |unique(X)| + |unique(Y)|}$$

# Extended Jaccard Coefficient (Beispiel)

---

## Extended Jaccard Coefficient (Beispiel)

---

- $x = \text{'Henri Waternoose'}$   
 $y = \text{'Henry Peter Waternose'}$

## Extended Jaccard Coefficient (Beispiel)

---

- $x = \text{'Henri Waternose'}$   
 $y = \text{'Henry Peter Waternose'}$
- $X = \{\text{'Henri'}, \text{'Waternose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$

## Extended Jaccard Coefficient (Beispiel)

---

- $x = \text{'Henri Waternose'}$   
 $y = \text{'Henry Peter Waternose'}$
- $X = \{\text{'Henri'}, \text{'Waternose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$
- $shared(X, Y) = \{(\text{'Henri'}, \text{'Henry'}), (\text{'Waternose'}, \text{'Waternose'})\}$

## Extended Jaccard Coefficient (Beispiel)

---

- $x = \text{'Henri Waternose'}$   
 $y = \text{'Henry Peter Waternose'}$
- $X = \{\text{'Henri'}, \text{'Waternose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$
- $shared(X, Y) = \{(\text{'Henri'}, \text{'Henry'}), (\text{'Waternose'}, \text{'Waternose'})\}$
- $unique(X) = \emptyset$   
 $unique(Y) = \{\text{'Peter'}\}$

## Extended Jaccard Coefficient (Beispiel)

- $x = \text{'Henri Waternose'}$   
 $y = \text{'Henry Peter Waternose'}$
- $X = \{\text{'Henri'}, \text{'Waternose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$
- $shared(X, Y) = \{(\text{'Henri'}, \text{'Henry'}), (\text{'Waternose'}, \text{'Waternose'})\}$
- $unique(X) = \emptyset$   
 $unique(Y) = \{\text{'Peter'}\}$
- $ExtJacc(X, Y) = \frac{2}{2+0+1} = \frac{2}{3}$

# Generalized Jaccard Coefficient [DHI12]

---

# Generalized Jaccard Coefficient [DHI12]

---

- Ähnlichkeit zwischen zwei Stringwerten  $x$  und  $y$  als normalisiertes Maximum Weighted Matching ihrer Token

# Generalized Jaccard Coefficient [DHI12]

---

- Ähnlichkeit zwischen zwei Stringwerten  $x$  und  $y$  als normalisiertes Maximum Weighted Matching ihrer Token
- Betrachtung der Ähnlichkeiten der Token von  $X = tok(x)$  und  $Y = tok(y)$  als bipartiter Graph

# Generalized Jaccard Coefficient [DHI12]

---

- Ähnlichkeit zwischen zwei Stringwerten  $x$  und  $y$  als normalisiertes Maximum Weighted Matching ihrer Token
- Betrachtung der Ähnlichkeiten der Token von  $X = tok(x)$  und  $Y = tok(y)$  als bipartiter Graph
- Entferne alle Kanten deren Gewicht (Ähnlichkeit zweier Token) kleiner als  $\theta$  ist

# Generalized Jaccard Coefficient [DHI12]

---

- Ähnlichkeit zwischen zwei Stringwerten  $x$  und  $y$  als normalisiertes Maximum Weighted Matching ihrer Token
- Betrachtung der Ähnlichkeiten der Token von  $X = tok(x)$  und  $Y = tok(y)$  als bipartiter Graph
- Entferne alle Kanten deren Gewicht (Ähnlichkeit zweier Token) kleiner als  $\theta$  ist
- Berechne das Maximum Weighted Matching  $M$  des reduzierten bipartiten Graphen

# Generalized Jaccard Coefficient [DHI12]

- Ähnlichkeit zwischen zwei Stringwerten  $x$  und  $y$  als normalisiertes Maximum Weighted Matching ihrer Token
- Betrachtung der Ähnlichkeiten der Token von  $X = tok(x)$  und  $Y = tok(y)$  als bipartiter Graph
- Entferne alle Kanten deren Gewicht (Ähnlichkeit zweier Token) kleiner als  $\theta$  ist
- Berechne das Maximum Weighted Matching  $M$  des reduzierten bipartiten Graphen
- Generalized Jaccard Coefficient:

$$GenJacc(X, Y) = \frac{\sum_{(t_1, t_2) \in M} TokenSim(t_1, t_2)}{|X| + |Y| - |M|}$$

# Maximum Weighted Matching

---

**Gegeben:** Ein (gewichteter) Graph  $G = (V, E, w)$

- Ein Matching  $M$  von  $G$  ist eine Menge an Kanten die sich keine Knoten teilen (die Kanten sind paarweise nicht adjazent)

# Maximum Weighted Matching

---

**Gegeben:** Ein (gewichteter) Graph  $G = (V, E, w)$

- Ein Matching  $M$  von  $G$  ist eine Menge an Kanten die sich keine Knoten teilen (die Kanten sind paarweise nicht adjazent)
- Ein Maximal Matching  $M$  von  $G$  ist ein Matching das nicht erweitert werden kann (d.h. das Hinzufügen einer weiteren Kante von  $G$  bedeutet, dass  $M$  kein Matching mehr ist)

# Maximum Weighted Matching

---

**Gegeben:** Ein (gewichteter) Graph  $G = (V, E, w)$

- Ein Matching  $M$  von  $G$  ist eine Menge an Kanten die sich keine Knoten teilen (die Kanten sind paarweise nicht adjazent)
- Ein Maximal Matching  $M$  von  $G$  ist ein Matching das nicht erweitert werden kann (d.h. das Hinzufügen einer weiteren Kante von  $G$  bedeutet, dass  $M$  kein Matching mehr ist)
- Ein Maximum Matching  $M$  von  $G$  ist ein Matching das (unter allen mgl. Matchings) die maximale Anzahl an Kanten besitzt

# Maximum Weighted Matching

---

**Gegeben:** Ein (gewichteter) Graph  $G = (V, E, w)$

- Ein Matching  $M$  von  $G$  ist eine Menge an Kanten die sich keine Knoten teilen (die Kanten sind paarweise nicht adjazent)
- Ein Maximal Matching  $M$  von  $G$  ist ein Matching das nicht erweitert werden kann (d.h. das Hinzufügen einer weiteren Kante von  $G$  bedeutet, dass  $M$  kein Matching mehr ist)
- Ein Maximum Matching  $M$  von  $G$  ist ein Matching das (unter allen mgl. Matchings) die maximale Anzahl an Kanten besitzt
- Ein Maximum Weighted Matching  $M$  von  $G$  ist ein Matching bei dem die Gesamtsumme aller Kantengewichte maximal ist

# Generalized Jaccard Coefficient (Beispiel)

---

# Generalized Jaccard Coefficient (Beispiel)

---

- $X = \{\text{'Henri'}, \text{'Waternoose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$

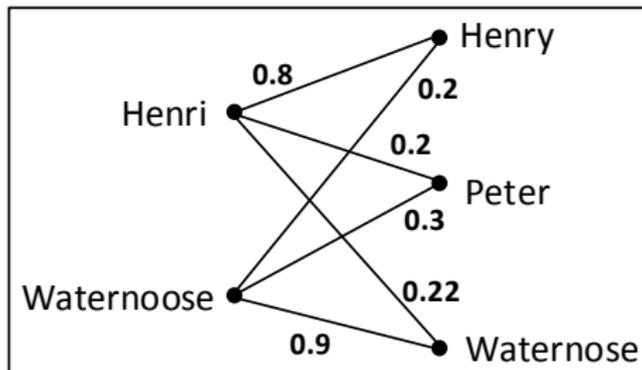
# Generalized Jaccard Coefficient (Beispiel)

---

- $X = \{\text{'Henri'}, \text{'Waternoose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$
- threshold  $\theta = 0.21$

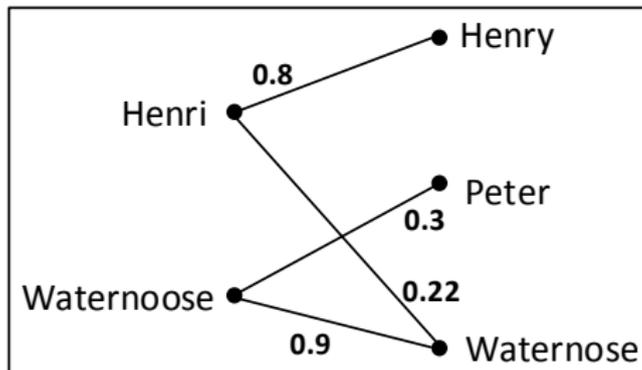
# Generalized Jaccard Coefficient (Beispiel)

- $X = \{\text{'Henri'}, \text{'Waternose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$
- threshold  $\theta = 0.21$



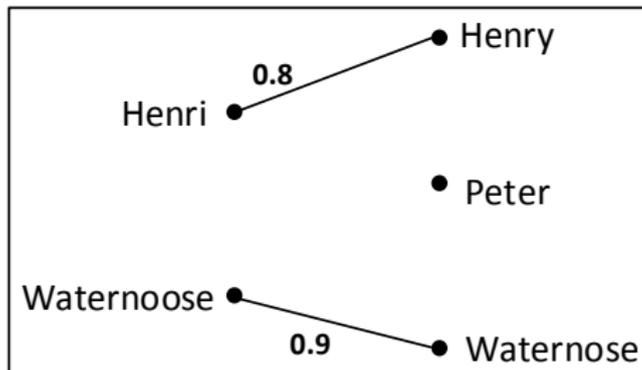
# Generalized Jaccard Coefficient (Beispiel)

- $X = \{\text{'Henri'}, \text{'Waternoose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$
- threshold  $\theta = 0.21$



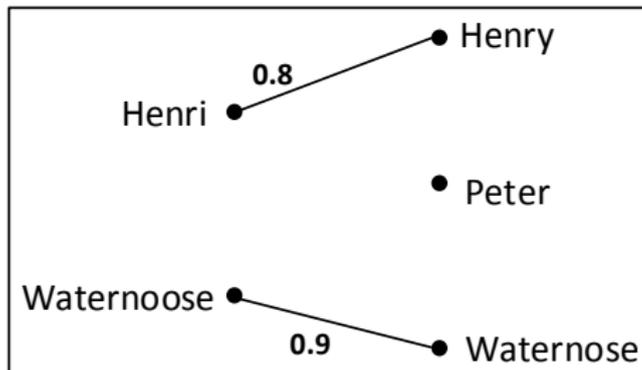
# Generalized Jaccard Coefficient (Beispiel)

- $X = \{\text{'Henri'}, \text{'Waternoose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$
- threshold  $\theta = 0.21$



# Generalized Jaccard Coefficient (Beispiel)

- $X = \{\text{'Henri'}, \text{'Waternose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$
- threshold  $\theta = 0.21$



- $GenJacc(X, Y) = \frac{0.8+0.9}{2+3-2} = \frac{1.7}{3} = 0.567$

# Monge-Elkan Similarity [ME97]

---

# Monge-Elkan Similarity [ME97]

---

- Ähnlichkeit von Stringwert  $x$  zu Stringwert  $y$  als durchschnittliche maximale Ähnlichkeit der Token in  $X = tok(x)$  zu Token in  $Y = tok(y)$

# Monge-Elkan Similarity [ME97]

---

- Ähnlichkeit von Stringwert  $x$  zu Stringwert  $y$  als durchschnittliche maximale Ähnlichkeit der Token in  $X = tok(x)$  zu Token in  $Y = tok(y)$
- Monge-Elkan Similarity:

$$MongeElkan(X, Y) = \frac{1}{|X|} \sum_{i=1}^{|X|} \max_{j=1}^{|Y|} TokenSim(t_i, t_j)$$

# Monge-Elkan Similarity [ME97]

- Ähnlichkeit von Stringwert  $x$  zu Stringwert  $y$  als durchschnittliche maximale Ähnlichkeit der Token in  $X = tok(x)$  zu Token in  $Y = tok(y)$
- Monge-Elkan Similarity:

$$MongeElkan(X, Y) = \frac{1}{|X|} \sum_{i=1}^{|X|} \max_{j=1}^{|Y|} TokenSim(t_i, t_j)$$

- gerichtetes Ähnlichkeitsmaß (**nicht** symmetrisch)  
⇒ für manche  $X$  und  $Y$  gilt:

$$MongeElkan(X, Y) \neq MongeElkan(Y, X)$$

# Monge-Elkan Similarity (Beispiel)

---

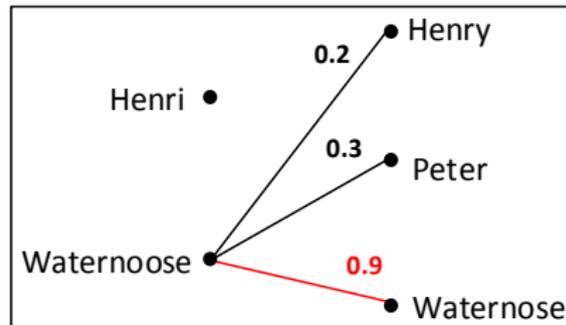
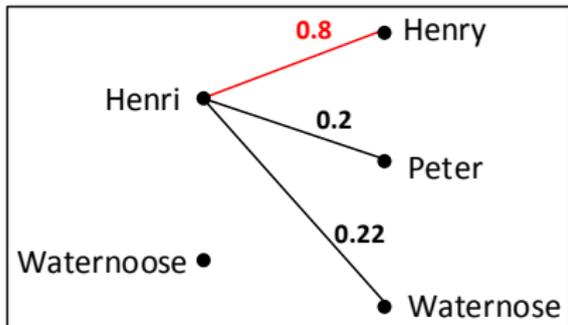
# Monge-Elkan Similarity (Beispiel)

---

- $X = \{\text{'Henri'}, \text{'Waternoose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$

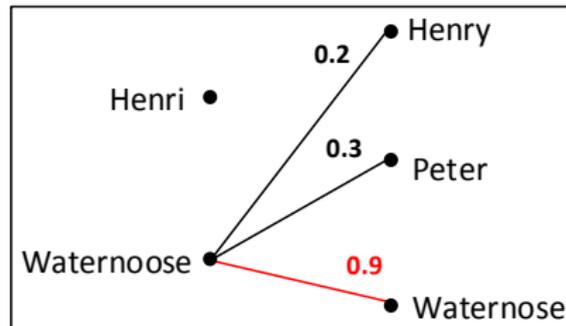
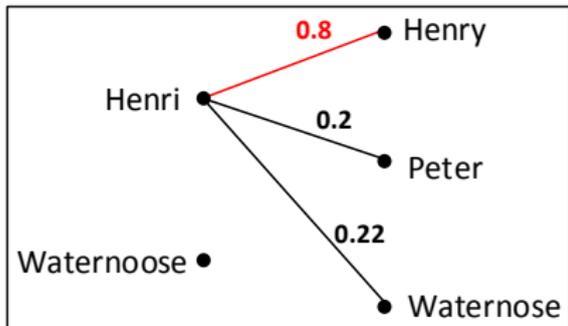
# Monge-Elkan Similarity (Beispiel)

- $X = \{\text{'Henri'}, \text{'Waternoose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$



# Monge-Elkan Similarity (Beispiel)

- $X = \{\text{'Henri'}, \text{'Waternoose'}\}$   
 $Y = \{\text{'Henry'}, \text{'Peter'}, \text{'Waternose'}\}$

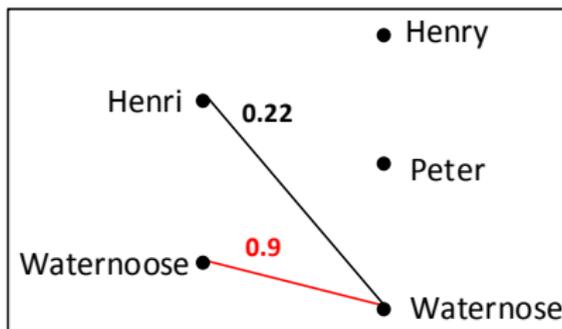
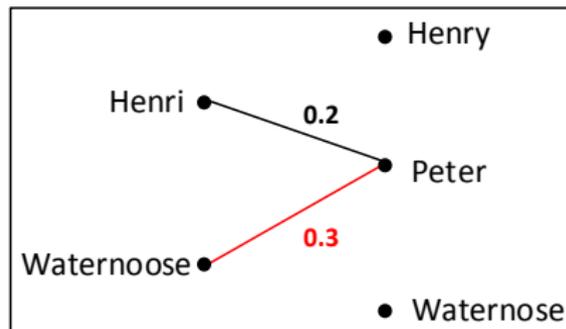
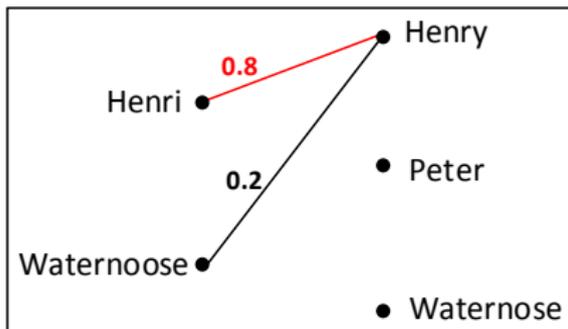


- $MongeElkan(X, Y) = \frac{1}{2} \times (0.8 + 0.9) = 0.85$

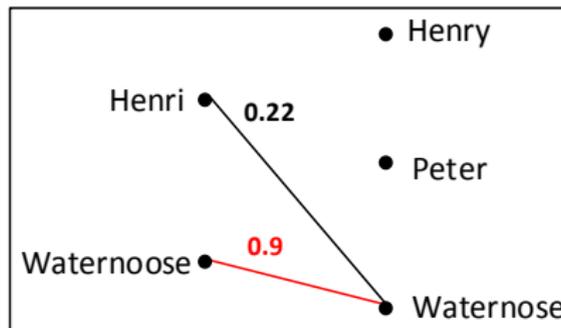
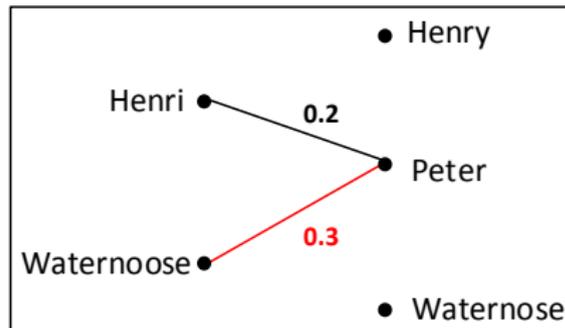
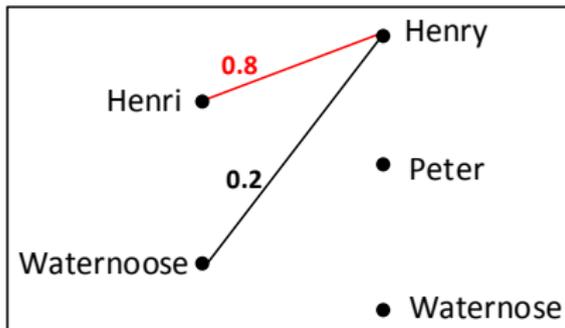
# Monge-Elkan Similarity (Beispiel)

---

# Monge-Elkan Similarity (Beispiel)



# Monge-Elkan Similarity (Beispiel)



- $$\text{MongeElkan}(Y, X) = \frac{1}{3} \times (0.8 + 0.3 + 0.9) = 0.67$$

# Agenda

---

- 1 Problemstellung
- 2 Genauigkeit
  - Mengenbasierte Ähnlichkeitsmaße
  - Sequenzbasierte Ähnlichkeitsmaße
  - Phonetische Ähnlichkeitsmaße
  - Semantische Ähnlichkeitsmaße
  - Hybride Ähnlichkeitsmaße
- 3 Skalierbarkeit

# Skalierbarkeit

---

- Vergleichen jedes Stringwertes  $x \in A$  mit jedem Stringwert  $y \in B$  ist bei großen Mengen zu ineffizient (quadratische Komplexität)

# Skalierbarkeit

---

- Vergleichen jedes Stringwertes  $x \in A$  mit jedem Stringwert  $y \in B$  ist bei großen Mengen zu ineffizient (quadratische Komplexität)
- Verwendung einer Methode *FindCands* welche eine Menge an möglichen Matches  $C \subseteq B$  für jedes  $x \in A$  effizient bestimmt

Für jeden Stringwert  $x \in A$

- verwende *FindCands* um eine Menge  $C \subseteq B$  zu bestimmen
- für jeden Stringwert  $y \in C$ 
  - wenn  $s(x, y) \geq \theta$ , dann klassifiziere  $(x, y)$  als Match

# Skalierbarkeit

- Vergleichen jedes Stringwertes  $x \in A$  mit jedem Stringwert  $y \in B$  ist bei großen Mengen zu ineffizient (quadratische Komplexität)
- Verwendung einer Methode *FindCands* welche eine Menge an möglichen Matches  $C \subseteq B$  für jedes  $x \in A$  effizient bestimmt

Für jeden Stringwert  $x \in A$

- verwende *FindCands* um eine Menge  $C \subseteq B$  zu bestimmen
  - für jeden Stringwert  $y \in C$ 
    - wenn  $s(x, y) \geq \theta$ , dann klassifiziere  $(x, y)$  als Match
- Hilfreich bei Schema Matching

# Skalierbarkeit

---

- Vergleichen jedes Stringwertes  $x \in A$  mit jedem Stringwert  $y \in B$  ist bei großen Mengen zu ineffizient (quadratische Komplexität)
- Verwendung einer Methode *FindCands* welche eine Menge an möglichen Matches  $C \subseteq B$  für jedes  $x \in A$  effizient bestimmt

Für jeden Stringwert  $x \in A$

- verwende *FindCands* um eine Menge  $C \subseteq B$  zu bestimmen
- für jeden Stringwert  $y \in C$ 
  - wenn  $s(x, y) \geq \theta$ , dann klassifiziere  $(x, y)$  als Match
- Hilfreich bei Schema Matching
- Bei Duplikatenerkennung wenig hilfreich, da  $\theta$  stark kontextabhängig

# Invertierter Index [DHI12]

---

# Invertierter Index [DHI12]

---

- Aufbauen eines invertierten Indexes über die Token der einzelnen Stringwerte von  $B$

# Invertierter Index [DHI12]

- Aufbauen eines invertierten Indexes über die Token der einzelnen Stringwerte von  $B$

## Set A

---

1: {lake, mendota}

2: {lake, monona, area}

3: {lake, mendota, monona, dane}

## Set B

---

4: {lake, monona, university}

5: {monona, research, area}

6: {lake, mendota, monona, area}

Token in B	ID Listen
area	5,6
lake	4,6
mendota	6
monona	4,5,6
research	5
university	4

# Invertierter Index [DHI12]

- Aufbauen eines invertierten Indexes über die Token der einzelnen Stringwerte von  $B$

## Set A

1: {lake, mendota}

2: {lake, monona, area}

3: {lake, mendota, monona, dane}

## Set B

4: {lake, monona, university}

5: {monona, research, area}

6: {lake, mendota, monona, area}

Token in B	ID Listen
area	5,6
lake	4,6
mendota	6
monona	4,5,6
research	5
university	4

- Ermöglicht ein schnelles Auffinden aller Stringwerte  $y \in B$  die ein Token mit einem Stringwert  $x \in A$  teilen

# Invertierter Index [DHI12]

- Aufbauen eines invertierten Indexes über die Token der einzelnen Stringwerte von  $B$

## Set A

1: {lake, mendota}

2: {lake, monona, area}

3: {lake, mendota, monona, dane}

## Set B

4: {lake, monona, university}

5: {monona, research, area}

6: {lake, mendota, monona, area}

Token in B	ID Listen
area	5,6
lake	4,6
mendota	6
monona	4,5,6
research	5
university	4

- Ermöglicht ein schnelles Auffinden aller Stringwerte  $y \in B$  die ein Token mit einem Stringwert  $x \in A$  teilen
- Die Reduktion an Vergleichen ist aber oft noch nicht groß genug

# Size Filtering [DHI12]

---

# Size Filtering [DHI12]

---

- Auswahl an Kandidaten anhand ihrer Länge

# Size Filtering [DHI12]

---

- Auswahl an Kandidaten anhand ihrer Länge
- B-Baum mit allen Stringwerten in  $B$  als Datensätze und deren Länge als Schlüsselwert.

# Size Filtering [DHI12]

---

- Auswahl an Kandidaten anhand ihrer Länge
- B-Baum mit allen Stringwerten in  $B$  als Datensätze und deren Länge als Schlüsselwert.
- Besonders nützlich für Jaccard Coefficient

$$\text{Jacc}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

# Size Filtering [DHI12]

- Auswahl an Kandidaten anhand ihrer Länge
- B-Baum mit allen Stringwerten in  $B$  als Datensätze und deren Länge als Schlüsselwert.
- Besonders nützlich für Jaccard Coefficient

$$Jacc(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

- Es gilt:  $Jacc(X, Y) \leq |Y|/|X| \leq 1/Jacc(X, Y)$   
(Anm.:  $Jacc(X, Y)$  maximal wenn  $Y \subseteq X$  oder  $X \subseteq Y$ )

# Size Filtering [DHI12]

- Auswahl an Kandidaten anhand ihrer Länge
- B-Baum mit allen Stringwerten in  $B$  als Datensätze und deren Länge als Schlüsselwert.
- Besonders nützlich für Jaccard Coefficient

$$Jacc(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

- Es gilt:  $Jacc(X, Y) \leq |Y|/|X| \leq 1/Jacc(X, Y)$   
(Anm.:  $Jacc(X, Y)$  maximal wenn  $Y \subseteq X$  oder  $X \subseteq Y$ )
- Vorgabe:  $x$  und  $y$  sind ein Match wenn  $Jacc(X, Y) \geq \theta$

# Size Filtering [DHI12]

- Auswahl an Kandidaten anhand ihrer Länge
- B-Baum mit allen Stringwerten in  $B$  als Datensätze und deren Länge als Schlüsselwert.
- Besonders nützlich für Jaccard Coefficient

$$Jacc(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

- Es gilt:  $Jacc(X, Y) \leq |Y|/|X| \leq 1/Jacc(X, Y)$

(Anm.:  $Jacc(X, Y)$  maximal wenn  $Y \subseteq X$  oder  $X \subseteq Y$ )

- Vorgabe:  $x$  und  $y$  sind ein Match wenn  $Jacc(X, Y) \geq \theta$

$\Rightarrow y$  kann nur ein Match von  $x$  sein wenn  $|X| \times \theta \leq |Y| \leq |X|/\theta$

# Size Filtering (Beispiel) [DHI12]

## Set A

---

1: {lake, mendota}

2: {lake, monona, area}

3: {lake, mendota, monona, dane}

## Set B

---

4: {lake, monona, university}

5: {monona, research, area}

6: {lake, mendota, monona, area}

Token in B	ID Listen
area	5,6
lake	4,6
mendota	6
monona	4,5,6
research	5
university	4

# Size Filtering (Beispiel) [DHI12]

## Set A

---

- 1: {lake, mendota}
- 2: {lake, monona, area}
- 3: {lake, mendota, monona, dane}

## Set B

---

- 4: {lake, monona, university}
- 5: {monona, research, area}
- 6: {lake, mendota, monona, area}

Token in B	ID Listen
area	5,6
lake	4,6
mendota	6
monona	4,5,6
research	5
university	4

- Annahme:  $\theta = 0.8$  und  $X = \{\text{lake, mendota}\}$

# Size Filtering (Beispiel) [DHI12]

## Set A

- 
- 1: {lake, mendota}  
 2: {lake, monona, area}  
 3: {lake, mendota, monona, dane}

## Set B

- 
- 4: {lake, monona, university}  
 5: {monona, research, area}  
 6: {lake, mendota, monona, area}

Token in B	ID Listen
area	5,6
lake	4,6
mendota	6
monona	4,5,6
research	5
university	4

- Annahme:  $\theta = 0.8$  und  $X = \{\text{lake, mendota}\}$
- Damit  $y$  ein Match von  $x$  sein kann, muss gelten:

$$1.6 = 2 \times 0.8 \leq |Y| \leq 2/0.8 = 2.5$$

# Size Filtering (Beispiel) [DHI12]

## Set A

- 
- 1: {lake, mendota}  
 2: {lake, monona, area}  
 3: {lake, mendota, monona, dane}

## Set B

- 
- 4: {lake, monona, university}  
 5: {monona, research, area}  
 6: {lake, mendota, monona, area}

Token in B	ID Listen
area	5,6
lake	4,6
mendota	6
monona	4,5,6
research	5
university	4

- Annahme:  $\theta = 0.8$  und  $X = \{\text{lake, mendota}\}$
- Damit  $y$  ein Match von  $x$  sein kann, muss gelten:

$$1.6 = 2 \times 0.8 \leq |Y| \leq 2/0.8 = 2.5$$

⇒ Da für alle  $y \in B$  gilt:  $|Y| \geq 3 \Rightarrow$  Kein Kandidat für  $x$

# Prefix Filtering (Overlap Measure) [DHI12]

---

# Prefix Filtering (Overlap Measure) [DHI12]

---

- Idee: Wenn zwei Mengen viele gemeinsame Elemente haben, müssen sich alle großen Untermengen überschneiden

# Prefix Filtering (Overlap Measure) [DHI12]

---

- Idee: Wenn zwei Mengen viele gemeinsame Elemente haben, müssen sich alle großen Untermengen überschneiden
- Wenn  $|X \cap Y| \geq k \Rightarrow$  jede Untermenge von  $X' \subseteq X$  mit  $|X'| \geq |X| - (k - 1)$  überschneidet sich mit  $Y$

# Prefix Filtering (Overlap Measure) [DHI12]

---

- Idee: Wenn zwei Mengen viele gemeinsame Elemente haben, müssen sich alle großen Untermengen überschneiden
- Wenn  $|X \cap Y| \geq k \Rightarrow$  jede Untermenge von  $X' \subseteq X$  mit  $|X'| \geq |X| - (k - 1)$  überschneidet sich mit  $Y$
- Beispiel:  $X = \{\text{lake, mendota, monona, dane}\}$   
und  $Y = \{\text{lake, mendota, monona, area}\}$

# Prefix Filtering (Overlap Measure) [DHI12]

---

- Idee: Wenn zwei Mengen viele gemeinsame Elemente haben, müssen sich alle großen Untermengen überschneiden
- Wenn  $|X \cap Y| \geq k \Rightarrow$  jede Untermenge von  $X' \subseteq X$  mit  $|X'| \geq |X| - (k - 1)$  überschneidet sich mit  $Y$
- Beispiel:  $X = \{\text{lake, mendota, monona, dane}\}$   
und  $Y = \{\text{lake, mendota, monona, area}\}$   
Überschneidung:  $|X \cap Y| = 3$

# Prefix Filtering (Overlap Measure) [DHI12]

- Idee: Wenn zwei Mengen viele gemeinsame Elemente haben, müssen sich alle großen Untermengen überschneiden
- Wenn  $|X \cap Y| \geq k \Rightarrow$  jede Untermenge von  $X' \subseteq X$  mit  $|X'| \geq |X| - (k - 1)$  überschneidet sich mit  $Y$
- Beispiel:  $X = \{\text{lake, mendota, monona, dane}\}$   
und  $Y = \{\text{lake, mendota, monona, area}\}$   
Überschneidung:  $|X \cap Y| = 3$   
 $\Rightarrow$  alle Mengen in  $\{X' \mid X' \subseteq X, |X'| \geq 2\}$  schneiden  $Y$

# Prefix Filtering (Overlap Measure) [DHI12]

- Idee: Wenn zwei Mengen viele gemeinsame Elemente haben, müssen sich alle großen Untermengen überschneiden
- Wenn  $|X \cap Y| \geq k \Rightarrow$  jede Untermenge von  $X' \subseteq X$  mit  $|X'| \geq |X| - (k - 1)$  überschneidet sich mit  $Y$
- Beispiel:  $X = \{\text{lake, mendota, monona, dane}\}$   
und  $Y = \{\text{lake, mendota, monona, area}\}$

Überschneidung:  $|X \cap Y| = 3$

$\Rightarrow$  alle Mengen in  $\{X' \mid X' \subseteq X, |X'| \geq 2\}$  schneiden  $Y$

z.B.  $X' = \{\text{lake, dane}\}$  schneidet  $Y$

# Prefix Filtering (Overlap Measure) [DHI12]

---

## Prefix Filtering (Overlap Measure) [DHI12]

---

- Vorgabe:  $x \in A$  und  $y \in B$  sind ein Match, wenn  $|X \cap Y| \geq k$

## Prefix Filtering (Overlap Measure) [DHI12]

---

- Vorgabe:  $x \in A$  und  $y \in B$  sind ein Match, wenn  $|X \cap Y| \geq k$
- Aufbau eines invertierten Indexes der Token in  $B$

## Prefix Filtering (Overlap Measure) [DHI12]

---

- Vorgabe:  $x \in A$  und  $y \in B$  sind ein Match, wenn  $|X \cap Y| \geq k$
- Aufbau eines invertierten Indexes der Token in  $B$
- Auswahl eines beliebigen  $X' \subseteq X$  mit  $|X'| \geq |X| - (k - 1)$   
(z.B. der Präfix von  $X$  der Länge  $|X| - (k - 1)$ )

# Prefix Filtering (Overlap Measure) [DHI12]

---

- Vorgabe:  $x \in A$  und  $y \in B$  sind ein Match, wenn  $|X \cap Y| \geq k$
- Aufbau eines invertierten Indexes der Token in  $B$
- Auswahl eines beliebigen  $X' \subseteq X$  mit  $|X'| \geq |X| - (k - 1)$   
(z.B. der Präfix von  $X$  der Länge  $|X| - (k - 1)$ )
- Bestimmung der Kandidaten von  $x$  mit Hilfe des Indexes

# Prefix Filtering (Overlap Measure) [DHI12]

---

- Vorgabe:  $x \in A$  und  $y \in B$  sind ein Match, wenn  $|X \cap Y| \geq k$
- Aufbau eines invertierten Indexes der Token in  $B$
- Auswahl eines beliebigen  $X' \subseteq X$  mit  $|X'| \geq |X| - (k - 1)$   
(z.B. der Präfix von  $X$  der Länge  $|X| - (k - 1)$ )
- Bestimmung der Kandidaten von  $x$  mit Hilfe des Indexes
- Reduziert Anzahl an Kandidaten pro  $x \in A$

# Prefix Filtering (Beispiel 1) [DHI12]

## Set A

---

- 1: {lake, mendota}
- 2: {lake, monona, area}
- 3: {lake, mendota, monona, dane}

## Set B

---

- 4: {lake, monona, university}
- 5: {monona, research, area}
- 6: {lake, mendota, monona, area}
- 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

# Prefix Filtering (Beispiel 1) [DHI12]

## Set A

---

- 1: {lake, mendota}
- 2: {lake, monona, area}
- 3: {lake, mendota, monona, dane}

## Set B

---

- 4: {lake, monona, university}
- 5: {monona, research, area}
- 6: {lake, mendota, monona, area}
- 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

- Annahme:  $k = 2$  und  $X = \{\text{lake, mendota}\}$

# Prefix Filtering (Beispiel 1) [DHI12]

## Set A

---

- 1: {lake, mendota}
- 2: {lake, monona, area}
- 3: {lake, mendota, monona, dane}

## Set B

---

- 4: {lake, monona, university}
- 5: {monona, research, area}
- 6: {lake, mendota, monona, area}
- 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

- Annahme:  $k = 2$  und  $X = \{\text{lake, mendota}\}$
- $X' = \{\text{lake}\}$

# Prefix Filtering (Beispiel 1) [DHI12]

## Set A

---

- 1: {lake, mendota}
- 2: {lake, monona, area}
- 3: {lake, mendota, monona, dane}

## Set B

---

- 4: {lake, monona, university}
- 5: {monona, research, area}
- 6: {lake, mendota, monona, area}
- 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

- Annahme:  $k = 2$  und  $X = \{\text{lake, mendota}\}$
- $X' = \{\text{lake}\}$
- Kandidaten für  $x$  sind:  $\{4, 6\}$

# Prefix Filtering (Beispiel 1) [DHI12]

## Set A

- 
- 1: {lake, mendota}  
 2: {lake, monona, area}  
 3: {lake, mendota, monona, dane}

## Set B

- 
- 4: {lake, monona, university}  
 5: {monona, research, area}  
 6: {lake, mendota, monona, area}  
 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

- Annahme:  $k = 2$  und  $X = \{\text{lake, mendota}\}$
- $X' = \{\text{lake}\}$
- Kandidaten für  $x$  sind:  $\{4, 6\}$
- Ohne Prefix Filtering: Kandidaten für  $x$  sind:  $\{4, 6, 7\}$

# Prefix Filtering (Beispiel 2) [DHI12]

## Set A

---

- 1: {lake, mendota}
- 2: {lake, monona, area}
- 3: {lake, mendota, monona, dane}

## Set B

---

- 4: {lake, monona, university}
- 5: {monona, research, area}
- 6: {lake, mendota, monona, area}
- 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

# Prefix Filtering (Beispiel 2) [DHI12]

## Set A

---

- 1: {lake, mendota}
- 2: {lake, monona, area}
- 3: {lake, mendota, monona, dane}

## Set B

---

- 4: {lake, monona, university}
- 5: {monona, research, area}
- 6: {lake, mendota, monona, area}
- 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

- Annahme:  $k = 2$  und  $X = \{\text{lake, monona, area}\}$

# Prefix Filtering (Beispiel 2) [DHI12]

## Set A

---

- 1: {lake, mendota}
- 2: {lake, monona, area}
- 3: {lake, mendota, monona, dane}

## Set B

---

- 4: {lake, monona, university}
- 5: {monona, research, area}
- 6: {lake, mendota, monona, area}
- 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

- Annahme:  $k = 2$  und  $X = \{\text{lake, monona, area}\}$
- $X' = \{\text{lake, monona}\}$

# Prefix Filtering (Beispiel 2) [DHI12]

## Set A

---

- 1: {lake, mendota}  
 2: {lake, monona, area}  
 3: {lake, mendota, monona, dane}

## Set B

---

- 4: {lake, monona, university}  
 5: {monona, research, area}  
 6: {lake, mendota, monona, area}  
 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

- Annahme:  $k = 2$  und  $X = \{\text{lake, monona, area}\}$
- $X' = \{\text{lake, monona}\}$
- Kandidaten für  $x$  sind:  $\{4, 6\}$  (lake) und  $\{4, 5, 6\}$  (monona)

# Prefix Filtering (Beispiel 2) [DHI12]

## Set A

- 
- 1: {lake, mendota}  
 2: {lake, monona, area}  
 3: {lake, mendota, monona, dane}

## Set B

- 
- 4: {lake, monona, university}  
 5: {monona, research, area}  
 6: {lake, mendota, monona, area}  
 7: {dane, area, mendota}

Token in B	ID Listen
area	5,6,7
lake	4,6
mendota	6,7
monona	4,5,6
research	5
university	4
dane	7

- Annahme:  $k = 2$  und  $X = \{\text{lake, monona, area}\}$
- $X' = \{\text{lake, monona}\}$
- Kandidaten für  $x$  sind:  $\{4, 6\}$  (lake) und  $\{4, 5, 6\}$  (monona)
- Ohne Prefix Filtering: Kandidaten für  $x$  sind:  $\{4, 5, 6, 7\}$

# Prefix Filtering (Jaccard Koeffizient) [DHI12]

---

# Prefix Filtering (Jaccard Koeffizient) [DHI12]

---

- Aus  $Jacc(X, Y) \geq \theta$  folgt [DHI12]:

$$|X \cap Y| \geq k = \lceil \frac{\theta}{1 + \theta} \times (|X| + |Y|) \rceil$$

# Prefix Filtering (Jaccard Koeffizient) [DHI12]

- Aus  $Jacc(X, Y) \geq \theta$  folgt [DHI12]:

$$|X \cap Y| \geq k = \lceil \frac{\theta}{1 + \theta} \times (|X| + |Y|) \rceil$$

- Beispiel:  $X = \{\text{lake, mendota, monona, dane}\}$   
und  $Y = \{\text{lake, mendota, monona, area}\}$

# Prefix Filtering (Jaccard Koeffizient) [DHI12]

- Aus  $Jacc(X, Y) \geq \theta$  folgt [DHI12]:

$$|X \cap Y| \geq k = \lceil \frac{\theta}{1 + \theta} \times (|X| + |Y|) \rceil$$

- Beispiel:  $X = \{\text{lake, mendota, monona, dane}\}$   
und  $Y = \{\text{lake, mendota, monona, area}\}$

Annahme:  $\theta = 0.5$

# Prefix Filtering (Jaccard Koeffizient) [DHI12]

- Aus  $Jacc(X, Y) \geq \theta$  folgt [DHI12]:

$$|X \cap Y| \geq k = \lceil \frac{\theta}{1 + \theta} \times (|X| + |Y|) \rceil$$

- Beispiel:  $X = \{\text{lake, mendota, monona, dane}\}$   
und  $Y = \{\text{lake, mendota, monona, area}\}$

Annahme:  $\theta = 0.5$

$$\Rightarrow k = \lceil 8/3 \rceil = \lceil 2.67 \rceil = 3$$

# Andere Ähnlichkeitsmaße [DHI12]

---

# Andere Ähnlichkeitsmaße [DHI12]

---

- Bisher nur Overlap Measure und Jaccard Koeffizient

## Andere Ähnlichkeitsmaße [DHI12]

---

- Bisher nur Overlap Measure und Jaccard Koeffizient
- Jede dieser Techniken kann für ein anderes Ähnlichkeitsmaß  $s(x, y)$  verwendet werden, wenn wir  $s(x, y)$  mit Hilfe von Constraints durch eines der beiden Maße ausdrücken können

# Andere Ähnlichkeitsmaße [DHI12]

- Bisher nur Overlap Measure und Jaccard Koeffizient
- Jede dieser Techniken kann für ein anderes Ähnlichkeitsmaß  $s(x, y)$  verwendet werden, wenn wir  $s(x, y)$  mit Hilfe von Constraints durch eines der beiden Maße ausdrücken können
- **Levenshtein:** Aus  $LevDst(x, y) \leq \epsilon$  folgt:

$$|X \cap Y| \geq k = \lceil \max(|X|, |Y|) + q - 1 - q \times \epsilon \rceil$$

wenn  $q$ -grams zur Tokenbildung verwendet wurden, d.h.  $X$  entspricht der Menge an  $q$ -grams von  $x$

# Andere Ähnlichkeitsmaße [DHI12]

- Bisher nur Overlap Measure und Jaccard Koeffizient
- Jede dieser Techniken kann für ein anderes Ähnlichkeitsmaß  $s(x, y)$  verwendet werden, wenn wir  $s(x, y)$  mit Hilfe von Constraints durch eines der beiden Maße ausdrücken können
- **Levenshtein:** Aus  $LevDst(x, y) \leq \epsilon$  folgt:

$$|X \cap Y| \geq k = \lceil \max(|X|, |Y|) + q - 1 - q \times \epsilon \rceil$$

wenn  $q$ -grams zur Tokenbildung verwendet wurden, d.h.  $X$  entspricht der Menge an  $q$ -grams von  $x$

- **Kosinus:** Aus  $CosSim(X, Y) \geq \theta$  folgt:

$$|X \cap Y| \geq k = \lceil \theta \times \sqrt{|X| \times |Y|} \rceil$$

# Literatur I

---

- [BG07] Indrajit Bhattacharya and Lise Getoor.  
Collective entity resolution in relational data.  
*TKDD*, 1(1), 2007.
- [DHI12] Anhai Doan, Alon Halevy, and Zachary Ives.  
*Principles of Data Integration*.  
Morgan Kaufmann, 2012.
- [ME97] Alvaro E. Monge and Charles Elkan.  
An efficient domain-independent algorithm for detecting approximately  
duplicate database records.  
In *DMKD*, pages 0-, 1997
- [NH10] Felix Naumann and Melanie Herschel.  
*An Introduction to Duplicate Detection*.  
Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

# Literatur II

---

- [Phi90] Lawrence Philips.  
Hanging on the metaphone.  
*Computer Language Magazine*, 7(12):39-44, December 1990.
- [Pos69] Hans Joachim Postel.  
Die kölnen Phonetik. Ein Verfahren zur Identifizierung von Personennamen  
auf der Grundlage der Gestaltanalyse.  
*IBM-Nachrichten*, pages 925-931, 1969.
- [Russ18] R.C. Russell.  
A method of phonetic indexing, 1918.
- [Russ22] R.C. Russell.  
A method of phonetic indexing, 1922.
- [Taf70] Robert L. Taft.  
*Name Search Techniques*.  
Bureau of Systems Development, New York State Identification  
and Intelligence System, 1970.