

# Erkennen von Duplikaten und nahen Duplikaten in Textkorpora

Dirk Flamming

IMSEM: Data Mining mit Apache Mahout (WS09/10)

Dozentin: Isabel Drost



Datenbanksysteme und Informationsmanagement  
Technische Universität Berlin

<http://www.dima.tu-berlin.de/>

- Duplikate und nahe Duplikate
  
- Techniken und Werkzeuge
  - Levenshtein-Distanz
  - Fingerprints
  - Stemming
  - Shingles
  
- Exemplarisch: „Syntactic clustering of the Web“
  - Ähnlichkeit von Dokumenten
  - Clustering-Algorithmus
  - Performance

- Identische Duplikate
  - sind exakte Kopien voneinander
  - Bsp.: „Hallo Welt“ – „Hallo Welt“
  
- Nahe Duplikate
  - sind einander ähnlich (wenige andere Wörter) oder das eine Dokument ist in dem anderen enthalten
  - Bsp.: „Hallo Welt“ – „<13:53>Hallo Welt“

- Anwendungsbereiche:
  - Web- Dokumente
    - z.B. in Suchmaschinen oder zum auffinden des Original-Beitrags
  - Dateien in Dateisystemen
    - z.B. um Redundanz von Dateien vorzubeugen
  - E-Mails
    - z.B. für einen Spamfilter
  - Domain-spezifische Korpora
    - z.B. um unterschiedliche Revisionen einer Datei zu finden

## ■ Levenshtein-Distanz

- Beschreibt die minimale Anzahl an Einfüge-, Lösch- und Änderungs-Operationen zwischen zwei Zeichenketten
- Wird vor allem zur Erkennung von Tippfehlern verwendet

- Bsp.:     Juli  
           Kulio

Distanz: 2

- Fingerprints
  - Wird durch Hash-Funktionen ermittelt
    - Problem: kleiner Unterschied zwischen Dokumenten entspricht evtl. einen starken Unterschied der Fingerprints
    - Lösung: simhash
  - ursprünglich nur für (identische) Duplikate verwendet
  - Kombination „simhash“-„Hamming-Abstand“:
    - ähnliche Dokumente haben ähnliche Fingerprints
    - je kleiner der Hamming Abstand, desto ähnlicher die Dokumente

- Stemming
  - Wird vor allem für kurze Text-Segmente verwendet
  - Wörter werden auf ihren Wortstamm zurückgeführt (*nicht linguistisch*)
    - z.B. Zerlegung in Vokal-Konsonanten-Sequenzen
  - Stemmer geben viele falsche Ergebnisse aus, jedoch bedeutend mehr Richtige

- Shingles (Dachziegel)
  - Ein *shingle* ist eine zusammenhängende Teilfolge von Tokens eines Dokuments  $D$ .
  - Das *w-shingling*  $S(D,w)$  ist die Menge aller (einzigartigen) Shingles der Länge  $w$  in  $D$ .
    - Diese Shingles überlagern sich gegenseitig wie Dachziegel

Bsp.:

Das *4-shingling* von  $(a,rose,is,a,rose,is,a,rose)$  entspricht der Menge

$\{(a,rose,is,a),(rose,is,a,rose),(is,a,rose,is)\}$

- Die Menge  $S(D,w)$  enthält  $n-w+1$  *shingles*, wobei  $n$  die Anzahl der Tokens in  $D$  angibt

- *Verwendung von shingles*

- *Mit Fingerprints verbrauchen diese weniger Speicher*

*"In der Praxis sind 64-bit Rabin-Fingerprints ausreichend" (Broder 2000)*

- *resemblance*  $r$  ist das Maß für die Ähnlichkeit zweier Dokumente zueinander

$$r(A, B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|}$$

- *containment*  $c$  ist das Maß, dass beschreibt wie stark ein Dokument A in einem Dokument B enthalten ist

$$c(A, B) = \frac{|S_A \cap S_B|}{|S_A|}$$

- Speicherung aller *shingles* selbst mit Fingerprints ungeeignet
- daher Schätzung von *resemblance* und *containment*
  - man permutiert die Menge der *shingles*  $t$ -mal
  - aus jeder dieser Permutationen wird das numerisch kleinste Element einer neuen Menge (dem sog. *sketch*) hinzugefügt
  - *Broder* zeigt, dass die Verhältnisse der gemeinsamen *shingles* mit denen dieser Schätz-Menge annähernd gleich sind.
  - Alternative: man nehme jedes  $m$ -te Element der Menge  $S(D, w)$

- Clustering-Algorithmus in vier Phasen unterteilt
  - Phase 1: *sketches* für alle Dokumente erstellen
  - Phase 2: *sketches* um  $\langle \text{shingle value}, \text{doc ID} \rangle$  erweitern und nach *shingle value* sortieren
  - Phase 3: alle gemeinsamen *shingles* zweier Dokumente finden
    - Sketches werden um  $\langle \text{doc ID}, \text{doc ID}, l \rangle$  Tripel erweitert
    - $l$  ist die Anzahl gemeinsamer shingles der beiden Dokumente
  - Phase 4: Berechnung der *resemblance* aller Tripel
    - Ist diese über einem gewissen Schwellenwert werden die Dokumente mit einem Union-Find Algorithmus einem Cluster hinzugefügt

- Eine Verbesserung der Performance ließe sich herbeiführen durch:
  - Entfernung von *common shingles*, da diese meist nur automatisch generierte Inhalte aufweisen
  - Dokumente mit identischen shingles zu einem anderen können für einige Anwendungen direkt herausgefiltert werden
  - Verwendung von super shingles:
    - Shingles von shingles
    - Dokumente mit einem gemeinsamen *super shingle*, haben eine hohe Ähnlichkeit
    - Phase 3 wird somit verkürzt, da man nicht alle gemeinsamen *super shingles* finden muss

Danke für die Aufmerksamkeit.

Gibt es noch Fragen?