

## 6 Anfrageausführung

Transformation von Anfragen auf Query-Vektoren

- + ggf. geeignetes Postprocessing für negierte Terme u.ä. (siehe Kap. 2)
- + ggf. Preprocessing zur Thesaurus-basierten **Query-Expansion**
- + ggf. **Query-Verfeinerung** nach erstem Resultat mit **Relevanz-Feedback** durch den Benutzer
  - durch Änderung der Termgewichte (Term Reweighting) (z.B. gemäß Probabilistischem Retrieval-Modell, siehe Kap. 3)
  - durch Hinzunahme von Termen mit geeigneten Gewichten

## Vereinfachter Query Processor

- Abb. auf mehrere Vektorraum-Queries
- 1) Entferne Stoppwörter aus der Anfrage und reduziere alle anderen Wörter auf Terme (in Stammform)
  - 2) Transformiere Anfrage  $q$  in die Form  

$$[(t_{11} \text{ AND } t_{12} \text{ AND } \dots \text{ AND } t_{1k_1}) \text{ OR } \dots \text{ OR } (t_{r1} \text{ AND } t_{r2} \text{ AND } \dots \text{ AND } t_{rk_r})] \text{ AND NOT } t_{(r+1)} \text{ AND NOT } \dots \text{ AND NOT } t_{(r+s)}$$
  - 3) for  $i=1$  to  $r$  do  
 Finde DocId-Listen für  $t_{i1}$  bis  $t_{ik_i}$  und berechne Vereinigung  $V_i$  dieser DocIds  
 Setze den RSV von  $d \in V_i$  auf  $\text{sim}(d, q_i)$   
 Berechne Vereinigung  $V$  der DocId-Mengen  $V_1, \dots, V_r$
  - 4) Sortiere alle  $d \in V$  absteigend nach ihren RSVs
  - 5) Hole Dokument  $d \in V$  in Sortierfolge  
 - ggf. mit einem heuristischen Abbruchkriterium - und eliminiere Dokumente  $d$ , in denen einer oder mehrere der Terme  $t_{(r+1)}$  bis  $t_{(r+s)}$  vorkommen
- Post-processing

## Thesaurus-basierte Query-Expansion

Für Query-Vektor  $q: t_1 t_2 \dots t_k$

generiere mittels Synonymen, Hypernymen, Hyponymen usw.

Query-Vektoren  $q_{\text{syn}}$  mit allen Synonymen der Terme von  $q$

$q_{\text{hyper}}$  mit allen Hypernymen der Terme von  $q$

$q_{\text{hypo}}$  mit allen Hyponymen der Terme von  $q$  usw.

sowie den ursprünglichen Query-Vektor  $q_{\text{orig}}$

Berechne Suchresult-Ranking gemäß

$$\text{sim}(d, q) = \sum_{i \in \{\text{orig}, \text{syn}, \text{hyper}, \text{hypo}, \dots\}} \alpha_i q \times d_i \quad (\text{Skalarprodukt pro Klasse } i)$$

mit geeigneten Gewichten  $\alpha_i$  ( $0 < \alpha_i < 1$ ,  $\alpha_{\text{orig}} = 1$ )

Bestimmung der Gewichte experimentell

## Query-Verfeinerung mit Relevanz-Feedback nach Rocchio

Für Resultat  $D$  der Query  $q$  bestimme das Relevanz-Feedback des Benutzers aus einer Partitionierung von  $D$  in

- $D_r$ : die Menge der relevanten Dokumente in  $D$  und
- $D_n$ : die Menge der nicht relevanten Dokumente in  $D$ .

Generiere verfeinerte Query  $q'$ :

$$\vec{q}' := \alpha \vec{q} + \frac{\beta}{|D_r|} \sum_{d_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|} \sum_{d_j \in D_n} \vec{d}_j$$

mit geeigneten Gewichten  $\alpha, \beta, \gamma \in [0, 1]$  (typischerweise  $\alpha > \beta > \gamma$ )

Variation:

$$\vec{q}' := \alpha \vec{q} + \beta \sum_{d_j \in D_r} \vec{d}_j - 1 \vec{d}_{\max} \quad \text{mit dem höchstplazierten Dokument } d_{\max} \in D_n$$

+ weitere Variationen

## Query-Verfeinerung durch Termauswahl

Aus  $D_r$  werden Terme  $T$  bestimmt, die für die Query-Verfeinerung potentiell interessant sind.

Die Termmenge  $T$  wird nach einem heuristischen Kriterium sortiert.

- Die diesbezüglich besten  $z$  Terme werden zur Query hinzugenommen, oder
- der Benutzer wählt Terme aus der sortierten Liste  $T$  aus.

Ein praxistaugliches Sortierkriterium ist z.B.

$$\text{rdf}_i * \text{rcf}_i * 1/\text{nois}_i \quad \text{mit} \quad \text{nois}_i := \sum_{j=1}^N \frac{tf_{ij}}{cf_i} \log_2 \frac{cf_i}{tf_{ij}}$$

mit  $\text{rdf}_i$ : Anzahl der Dokumente in  $D_r$ , die  $t_i$  enthalten

$\text{rcf}_i$ : akkumulierte Häufigkeit von  $t_i$  in  $D_r$

Alternative Sortierkriterien sind u.a.

$$\log \frac{p_i(1-q_i)}{(1-p_i)q_i} \quad \text{mit } p_i = P[X_i=1|R] \text{ und } q_i = P[X_i=1|\neg R] \text{ o.ä.}$$