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 Vektor-raum-Queries

1) Entferne Stoppwörter aus der Anfrage und
reduziere alle anderen Wörter auf Terme (in Stammform)

2) Transformiere Anfrage q in die Form
[(t11 AND t12 AND ... AND $t1k_1$) OR ... OR
(tr1 AND tl2 AND ... AND trk_r)]
AND NOT t(r+1) AND NOT ... AND NOT t(r+s)

} q1 ... qr

3) for i=1 to r do
Finde DocId-Listen für ti1 bis tik_i und
berechne Vereinigung Vi dieser DocIds
Setze den RSV von d ∈ Vi auf sim(d,qi)
Berechne Vereinigung V der DocId-Mengen V1, ..., Vr

4) Sortiere alle d ∈ V absteigend nach ihren RSVs

Post-processing

5) Hole Dokument d ∈ 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

Thesaurus-basierte Query-Expansion

Für Query-Vektor q: t1 t2 ... tk

generiere mittels Synonymen, Hypernymen, Hyponymen usw.

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

q_{hyper} mit allen Hypernymen der Terme von q

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

sowie den ursprünglichen Query-Vektor q_{orig}

Berechne Suchresult-Ranking gemäß

$$sim(d,q) = \sum_{i \in \{orig, syn, hyper, hypo, ...\}} a_i \, q \times d_i$$ (Skalarprodukt pro Klasse i)

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

Bestimmung der Gewichte experimentell

Query-Verfeinerung mit Relevanz-Feedback nach Rocchio

Für Resultat D der Query q bestehe 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‘:

$$\vec{q}' := a\vec{q} + \frac{b}{|D_r|} \sum_{d_j \in D_r} \vec{d}_j - \frac{g}{|D_n|} \sum_{d_j \in D_n} \vec{d}_j$$

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

Variation:

$$\vec{q}' := a\vec{q} + b \sum_{d_j \in D_r} \vec{d}_j - l\, \vec{d}_{\max}$$

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.

$$rdf_i * rcf_i * 1/noise_i \quad \text{mit} \quad noise_i := \sum_{i=1}^{N} \frac{tf_{ij}}{cf_i} \log_2 \frac{cf_i}{tf_{ij}}$$

mit rdf_i: Anzahl der Dokumente in D_r, die t_i enthalten

rcf_i: akkumulierte Häufigkeit von t_i in D_r

Alternative Sortierkriterien sind u.a.

$$\log \frac{p_i(1-q_i)}{(1-p_i)q_i}$$ mit pi=P[Xi=1|R] und qi=P[Xi=1|¬R] o.ä.