2. Übung zur Vorlesung "Internet-Suchmaschinen" im Sommersemester 2009

– mit Lösungsvorschlägen –

Prof. Dr. Gerd Stumme, M.Sc. Wi-Inf. Beate Krause

06. Mai 2009

1 Boolesches Retrieval (2)

Eine Erweiterung des booleschen Retrieval Models kann ein Ranking durch die Einbeziehung von Termhäufigkeiten in Dokumenten sein. Je höher die Vorkommen eines Terms im Dokument, desto weiter oben in der Rangliste kann das Dokument eingeordnet werden.

- x AND y: $tf(x, D) * tf(y, D)$
- x OR y: $tf(x, D) + tf(y, D)$
- NOT x: 0 if $tf(x, D) > 0$, 1 if $tf(x, D) = 0$

Betrachten Sie die beiden folgenden Dokumente:

D_1 max sagt fischers fritz fischt frische fische
D_2 moritz sagt fischers fritz fischt frische fische frische fische fischt fischers fritz

1. Bestimmen Sie ein Ranking für die folgenden Anfragen: (fritz OR max) AND fische, max OR (moritz AND fische), fritz AND NOT fische.
   - (fritz OR max) AND fische: $Score(D_1) = 2 * 1$, $Score(D_2) = 2 * 2$, damit wird D_2 höher gerankt.
   - max OR (moritz AND fische): $Score(D_1) = 1$, $Score(D_2) = 2$, damit wird D_2 höher gerankt.
   - fritz AND NOT fische: Die Ergebnismenge für diese Anfrage ist die leere Menge.
2. Können Sie sich weitere Abwandlungen des booleschen Modells überlegen, die dessen Ausdrucksmächtigkeit erhöhen?
   - Normalisierung der Termhäufigkeiten mit Beachtung der Länge des Dokuments

	max	moritz	fische	fischers	fischt	frische	fritz	sagt
D_1	1	0	1	1	1	1	1	1
D_2	0	1	2	2	2	2	2	1

Table 1: Term-Dokument-Matrix mit einfachen Häufigkeiten als Gewichtung

	max	moritz	fische	fischers	fischt	frische	fritz	sagt
D_1	0.5	0	0.5	0.5	0.5	0.5	0.5	0.5
D_2	0	0.5	1	1	1	1	1	0.5

Table 2: Term-Dokument-Matrix mit normierten Häufigkeiten als Gewichtung

	max	moritz	fische	fischers	fischt	frische	fritz	sagt
D_1	0.7	0	0	0	0	0	0	0
D_2	0	0.7	0	0	0	0	0	0

Table 3: Term-Dokument-Matrix mit TF-IDF als Gewichtung

- Fuzzy-Mengenoperatoren: *foo* kommt sehr oft vor, *bar* selten, also *foo AND bar* mit mittlerer Häufigkeit, usw.
- Unterscheidung, ob Terme zusammen oder weit voneinander entfernt vorkommen
- Berücksichtigung der Formatierung des Dokuments, z. B.
  - Gesonderte Behandlung von Text in Überschriften
  - Position des Textes im Dokument

2 TF-IDF Gewichtung

1. Betrachten Sie wieder die folgenden Dokumente:

   D_1 max sagt fischers fritz fischt frische fische
   D_2 moritz sagt fischers fritz fischt frische fische frische fische fischt fischers fritz

   Stellen Sie drei Term-Dokument-Matrizen auf. Die erste Matrix enthält als Gewicht die Termfrequenz ohne Normierung, die zweite Matrix als Gewicht die Termfrequenz mit Normierung und die dritte Matrix enthält eine TF/IDF-Gewichtung.

2. Wie sieht das IDF aus, wenn ein Term in allen Dokumenten erscheint? Was bedeuted dies für das Ranking?

   Das IDF wird null, denn $IDF = log(N/N) = log(1) = 0$. Damit wird auch der

TF-IDF Wert null. Terme, die also in allen Dokumenten vorkommen, werden im Ranking nicht berücksichtigt.

3. Wie sieht das IDF aus, wenn ein Term in genau einem Dokument erscheint? Was bedeutet das für das Ranking?

   Kommt ein Dokument genau einmal im Korpus vor, dann ist der IDF Wert $idf = log(N/df) = log(N/1) = log(N)$. Damit wird der höchstmögliche IDF Wert erreicht. Die IDF Werte im Ranking bewerten also diejenigen Terme höher, die nicht im gesamten Korpus verteilt sind. In Kombination mit den Häufigkeitswerten (TF) werden also diejenigen Terme höher gewichtet, die häufig vorkommen, aber nur in wenigen Dokumenten.

4. Warum ist der IDF Wert eines Terms immer endlich?

   Wir definieren eine untere und eine obere Grenze für die IDF Werte. Obere Grenze: Wir nehmen an, das $DF(0) \neq 0$, ein Term kommt also mindestens einmal im Korpus vor. Dann gilt: $idf = log(N/df) = log(N)$. Untere Grenze: Ein Term kommt in jedem Dokument vor. Dann gilt (s.o.) $idf = log(N/N) = log(1) = 0$.

   Die Funktion zwischen den beiden Grenzen ist monoton fallend: $x < y \rightarrow \frac{1}{x} > \frac{1}{y} \rightarrow log(\frac{1}{x}) > log(\frac{1}{y})$ Der Bildbereich der Funktion liegt also zwischen den oben definierten Grenzen.

3 Vektorraummodel

1. In der Vorlesung wurde ein Maß $cosSim(a, b)$ für die Ähnlichkeit zweier Dokumente a und b eingeführt. Dementsprechend sei $d_c(a, b) := 1 - cosSim(a, b)$ als Abstandsmaß definiert.

   Weiterhin kann man die euklidische Distanz $d_e(a, b) := \|a - b\|_2 := \sqrt{\sum_i (a_i - b_i)^2}$ definieren.

   Berechnen Sie den euklidischen und den Kosinusabstand von D_1 und D_2 für die Matrix mit den einfachen Häufigkeiten. (Sie brauchen nicht die numerischen Werte auszurechnen, Ausdrücke der Art $3 + \sqrt{19}$ reichen.) Was beobachten Sie?

$$\begin{aligned} d_c(D_1, D_2) &= 1 - \frac{1+2+2+2+2+2}{\sqrt{7}\sqrt{22}} \\ &= 1 - \frac{11}{\sqrt{154}} \\ &\approx 0.11 \end{aligned}$$

$$\begin{aligned} d_e(D_1, D_2) &= \sqrt{1^2 + 1^2 + (5 \cdot 1^2) + 0^2} \\ &= \sqrt{7} \\ &\approx 2.6 \end{aligned}$$

Wie man sieht, liefert die Kosinusdistanz für Dokumente ähnlichen Inhalts, aber unterschiedlicher Länge einen kleinen Abstand, während die Eukliddistanz hier einen großen Abstand ergibt.

2. Rechnen Sie nach, in welchem Zusammenhang die beiden Maße stehen, wenn man mit normierten Dokumenten ($\|a\| = \|b\| = 1$) arbeitet!

   Es gilt: $d_c(a, b) = 1 - \frac{\langle a,b \rangle}{\|a\| \, \|b\|} = 1 - \langle a, b \rangle = 1 - \sum_i a_i b_i$

   Für das Euklid-Maß gilt:

$$\begin{aligned} d_e(a, b) &= \|a - b\|_2 \\ &= \sqrt{\sum_i (a_i - b_i)^2} \\ &= \sqrt{\sum_i (a_i^2 + b_i^2 - 2 a_i b_i)} \\ &= \sqrt{\sum_i a_i^2 + \sum_i b_i^2 - 2 \sum_i a_i b_i} \\ &= \sqrt{1 + 1 - 2 \, \langle a, b \rangle} \qquad \text{(wegen } \|a\| = \|b\| = 1\text{)} \\ &= \sqrt{2 \, d_c(a, b)} \end{aligned}$$

   Für normierte Dokumente gilt also der einfache Zusammenhang $d_e = \sqrt{2 \, d_c}$. Das Ranking der Dokumente ist also für beide Maße dasselbe, nur die Werte sind unterschiedlich.

4 Praxisübung

Abgabe: 19.05.2009

Implementieren Sie einen invertierten Index mit TF-IDF-Gewichtung entsprechend dem Interface `InvertedIndex`! Die Termgewichte sollen wie in der Vorlesung skizziert normiert sein. Es gibt wieder eine Testklasse `IndexText`, mit der Sie Ihren Index ausprobieren können. Folgende Dokumente sind die am höchsten gerankten für die jeweils genannten Terme:

Term	Datei → Gewicht, ...
november	8683 → 0.5246, 3639 → 0.1749, ...
shipbuilding	1902 → 0.2623, 6541 → 0.2623, 5818 → 0.1749, ...
sugarcane	10306 → 0.2736, 11173 → 0.2736, 4630 → 0.1824, 259 → 0.1824, ...

Tip: Implementieren Sie die Postinglisten als absteigend sortierte `Collection` von `TokenOccurence`-Objekten. Es ist etwas trickreich, dazu das Interface `Comparable<TokenOccurrence>` in `TokenOccurrence` korrekt (!) zu implementieren. Lesen Sie zuerst die Java-Dokumentation zu `Comparable`!