2. Übung zur Vorlesung "Internet-Suchmaschinen" im Wintersemester 2007/2008

Dr. Andreas Hotho, Prof. Dr. Gerd Stumme, MSc. Wi-Inf Beate Krause

15. November 2007

1 Tokenizing

1. Skizzieren Sie in Pseudocode oder einer Programmiersprache Ihrer Wahl, wie man aus einem HTML-Dokument den reinen Text extrahieren kann.

   Welche Probleme hat Ihre einfache Lösung noch?

2. Berücksichtigt Ihr Verfahren auch

   - `ALT`-Tags von Bildern?
   - `TITLE`-Tags von Hyperlinks?
   - Keywords in `META`-Tags?
   - Kommentare und Skripte (sollen ausgeblendet werden!)

   Wie aufwendig ist es, dies alles nachzurüsten?

3. Man will die oben genannten Features haben und zusätzlich auch noch Strukturinformation verarbeiten, also etwa die erste `<h1>`-Überschrift besonders gewichten, Hyperlinks extrahieren, usw.

   Finden Sie eine elegantere Möglichkeit, dies alles anzubieten, ohne von Hand einen entsprechenden Tokenizer zu bauen?

2 Indexstrukturen

Wir nehmen an, daß wir einen invertierten Index für folgenden Korpus gebaut haben:

- 1.000.000.000 Dokumente
- 2.000.000 Terme
- jeder Term komme im Mittel in 100.000 Dokumenten vor

- jeder Term und jeder Listeneintrag sei 16 Byte groß.

Der Index stehe als Aneinanderreihung der Terme und Dokument-Term-Gewichte auf dem Sekundärspeicher.
Weiterhin haben wir den Index auf einem RAID-Array mit

- 4 kB Blockgröße, 64 Bit breite Blocknummern
- 8 ms mittlerer Zugriffszeit
- 50 MB/s Übertragungsrate bei linearem Zugriff

1. Schätzen Sie ab, wie lange das Auffinden eines Term-Vorkommens bei linearer Suche im Mittel dauern würde.
2. Kennen Sie eine Indexstruktur, um eine solche Suche auf dem Hintergrundspeicher zu beschleunigen? Schätzen Sie die Kosten mit einer solchen Datenstruktur ab.

3 Evaluation

Ein Student soll ein Referat über den höchsten Vulkan der Welt, Ojos del Salado, schreiben. Dafür nutzt er zwei Suchmaschinen, und erhält jeweils eine Liste aus 30 URLs. Für diese erstellt er folgende Relevanzlisten (+ repräsentiert eine relevante URL, - repräsentiert eine nicht relevante URL):

$\triangle_1 = (+|+|-|+|-|-|-|+|-|-|-|+|-|-|-|-|-|-|+|-|-|-|-|-|-|-|-|+)$

$\triangle_2 = (-|+|+|-|-|-|+|-|-|+|+|-|+|-|-|-|+|-|-|+|-|-|+|-|+|-|-|-|-|-)$

- Zeichnen Sie für beide Systeme den Precision-Recall Graphen. Ermitteln sie die Precision und Recall Werte dabei in 5er-Abständen (also den 1., 5., 10. usw. Precision Wert). Gehen Sie davon aus, dass für jede Liste insgesamt 12 Dokumente relevant sind.
- Welche Ranking-Liste ist in welchem Anwendungsszenario besser?
- Das F-Measure wird als harmonisches Mittel von Precision und Recall definiert. Welchen Vorteil hat die Benutzung des harmonischen Mittels gegenüber des arithmetischen Mittels?

4 Praxisübung

Abgabe: 28.11.2007

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`!