

Information Retrieval - Sommersemester 2008

Dipl.-Inform. Ingo Frommholz, LF 138

ingo.frommholz@is.inf.uni-due.de

Übungsblatt 6

keine Abgabe

Aufgabe 12: Implementierung einer invertierten Liste

- (a) Implementiere eine invertierte Liste, die Boolesches Retrieval ermöglichen soll. Die invertierte Liste soll dabei z.B. das folgende Java-Interface unterstützen:

```
public interface SimpleIndex {

    /**
     * Add a single document to the index
     * @param docid the document ID
     * @param docterm the document terms and their frequencies
     */
    public void addDocument(String docid,
                           Hashtable<String, Integer> docTerms);

    /**
     * Find all documents in index containing the search term
     * @param searchTerm the search term
     * @return the document list
     */
    public List<String> getDocs(String searchTerm);

    /**
     * Find all documents in index not containing the search term
     * @param searchTerm the search term
     * @return the resulting document list
     */
    public List<String> getDocsNOT(String searchTerm);

    /**
     * find all documents in a list not containing the search term
     * @param searchTerm the search term
     * @return the resulting document list
     */
    public List<String> getDocsNOT(String searchTerm,
                                   List<String> docids);
}
```

```

/**
 * Combine two partial results with OR
 * @param first document list
 * @param 2nd document list
 * @return the resulting document list
 */
public List<String> combineResultsOR(List<String> docids1,
                                   List<String> docids2);

/**
 * Combine two partial results with AND
 * @param first document list
 * @param 2nd document list
 * @return the resulting document list
 */
public List<String> combineResultsAND(List<String> docids1,
                                     List<String> docids2);

```

- (b) Schreibe ein kleines Java-Programm, das eine (hartkodierte) Menge von Texten indexiert. Stemming und Stopwort-Eliminierung können an dieser Stelle ignoriert werden.
- (c) Schreibe ein kleines Java-Programm, das für eine einfache Boolesche Anfrage mit maximal zwei Termen (a , $a \wedge b$, $a \vee b$, $a \wedge \neg b$, $a \vee \neg b$, ...) die passenden Dokumente aus den indexierten Artikeln zurück liefert.
- (d) *Coordination Level Match* ist ein vereinfachter Spezialfall des Vektorraum-Modells. Setze es um, so dass zu einer Liste von Anfragetermen eine ge-rankte Liste der indexierten Dokumente zurückgeliefert wird.
- (e) Überlege, wie man Dein Programm für den Index erweitern müsste, um das Vektorraummodell mit $tf \times idf$ zu unterstützen.

Aufgabe 13: Pattern Matching

Gegeben seien der folgende Text und das gesuchte Pattern:

Test: VIERHUNDERTVIERUNDVIERZIG
Pattern: VIERZIG

- (a) Führe in allen Einzelschritten eine Textsuche nach Knuth-Morris-Pratt durch. Wie sieht insbesondere die Präfix-Funktion *next* aus?
- (b) Führe in allen Einzelschritten eine Textsuche nach Boyer-Moore durch. Wie sehen insbesondere die Felder *dd* und *d* aus?
- (c) Verwende schließlich den Shift-Or-Algorithmus für eine Suche nach dem Pattern VIERZIG. Betrachte insbesondere, wie sich der jeweils neue Statusvektor ergibt.