Multimedia Information Extraction and Retrieval SoSe 2010
Exercise Sheet 1

Prof. Dr. Ralf Möller, Sebastian Wandelt

21.04.2010

1. Explain why naive matrix-representations for term incidence vectors are not feasible in practice. What is the underlying idea for alternative representations?

   Solution:

   - *number of entries=number of documents* $*$ *number of terms*
   - *matrizes grow fast, but are extremely sparse*
   - *alternative: only record the* 1*-positions; inverted index with linked lists (why linked lists? for updates ... keep order)*

2. Explain the notions of *dicitionary files* and *posting files*.

   Solution:

   - *idea: split document-specific information (posting) from summary information (dictionary)*
   - *dictionary: term, number of documents and frequency sum*
   - *posting: document id and freuqency*

3. Why are postings sorted by document ID? **Solution:**

- *for query processing etc. we often need to perform join operations*
- *for sorted lists, optimized implementations of joins exist (e.g. sort-merge join)*
- *thus, only a matter of efficiency (i.e. a little more overhead during preparation (=sorting) yields better performance online (query answering))*

4. What is the idea of *tokenization*? What are the major issues/problems? **Solution:**

   Idea: break up sentences into units (=tokens). Issues:

   - *Composite nouns,*
   - *Numbers/dates,*
   - *Language issues,*
   - *Upper/lower case,*
   - *etc.*

5. Are *stop words* still used nowadays when creating dictionaries? **Solution:**

   - *Idea of stop words: Exclude common words from the dictionary (because the likelyhood of occurence in queries is expected to be quite low)*
   - *examples: the, I, a, and*
   - *not used so often anymore, because of extended compression techniques*

6. Which approaches for optimizing merge operations do you know? **Solution:**

   - *naive solution: walk through two postings simultaneously (linear in the total number of postings)*
   - *optimization: introduce skip pointers, e.g. hints, if we can skip n elements*

- *example:*
  - *Given* 20 *documents, and a query* $q = q_1 \cap q_2$
  - *let* $sol(q_1) = [1, 3, 17, 19]$
  - *let* $sol(q_2) = [2, 3, 5, 9, 10, 12, 17, 18]$
  - *then skip pointers could help us to* skip *(in the ebst case)* $5, 9, 10, 12$ *above*
  - *question: where to place the skip pointers to have most use?*

7. Which approaches for answering *phrase queries* do you know, and what disadvantages do they have? **Solution:**

   - *biword-dictionaries (store pairs of terms) ... yields false positives*

   - *positional indexes ... (store for each term its position in the document)... problem: posting storage grows substantially*