

System architecture

Ola Natvig

March 24, 2009

1 System architecture

The system is designed around three major data structures, or indices. These are the occurrence index which store inverted lists with positions. The content index stores the content of the documents, and is used for snippets and clustering. The last index is the statistics index which is used to store document statistics which may be used to calculate relevance.

All these indices are wrapped by one class **SearchEngineMaster**, the interface for the system uses this master class to feed documents and to execute queries.

1.1 Statistics index structure

The statistics index is a mapping from document id to a statistics object containing information such as most frequent term, document vector $tf * idf$ length and number of unique terms.

1.2 Content index structure

The content index stores the content of the documents indexed, these documents are stored as lists of tokens. These are the raw tokens of the documents nothing is added or removed. By concatenating any consecutive subsequence of these tokens a section of the original document will be produced. This property are used when extracting snippets.

1.3 Occurrence index structure

The occurrence index consists of two parts, the dictionary and the inverted file. The dictionary contains the terms of documents, and a pointer into the inverted file. The inverted file contains the inverted lists for each term in alphabetical order. The inverted list of a term contains the documents occurrences that contain the term sorted on increasing document id. Each document occurrence contains a list the positions within the document where the term occurred. The index structure is shown in Figure ??.

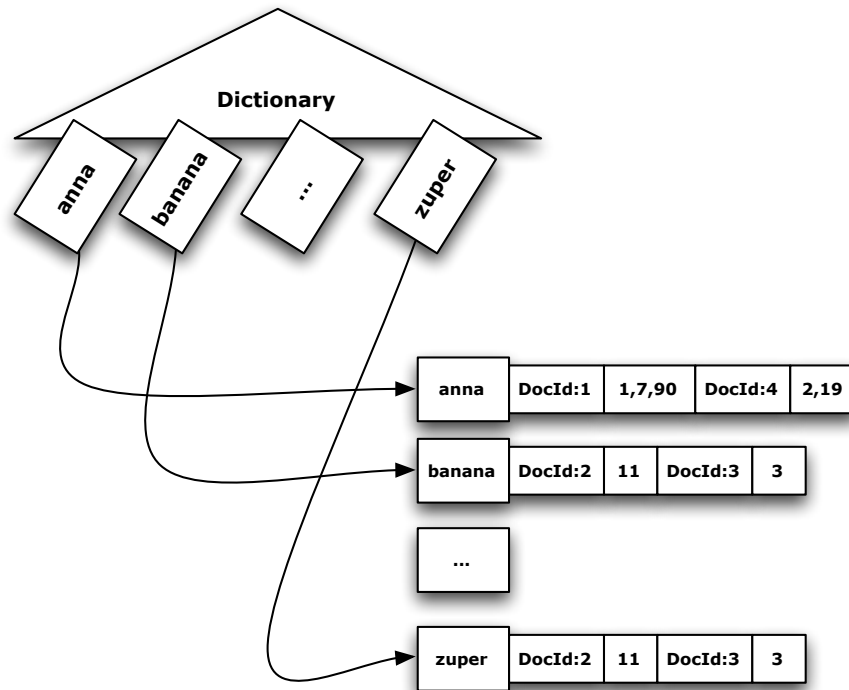


Figure 1: Occurrence index structure.

1.4 Index building

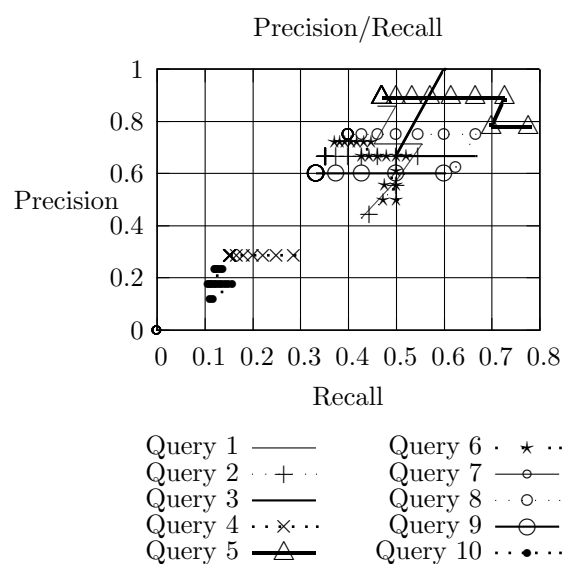
When building indices, all documents fed to the system are given a unique document id from a global counter.

The occurrence index is built in two phases, in the first phase documents are converted to inverted lists and combined into one inverted list representing one index update.

When the index is flushed, the index update that were built in the first phase is merged with the existing inverted index. Since the inverted lists are sorted on term, document and position merging the lists is a trivial task.

During this merge, the statistics for each document is calculated and stored in the statistics index.

When the merge is completed the dictionary is updated so that the pointers in the dictionary points into the new inverted file.



(a) Average QPS

Figure 2: Varied concurrency level using 8 nodes