Inverted Files

1 Inverted file index

Inverted file is an index structure which is used for text queries. The structure of an inverted file index can be considered as two parts: dictionary and inverted lists. Considering a collection of documents C, let us assign to every document its identifier d. Dictionary is used to determine distinct words w within all the documents and to count the number of documents Nd in which the word w appears. Dictionary is connected to inverted lists by pointers, that connects a word w with a corresponding inverted list that consists of records of the form $< d, f >$, where d represents the document in which the word w appears and f represents the frequency of a word w in a document d. Thus, dictionary with pointers to inverted lists construct an index, where the whole collection of documents is stored and can be retrieved by queries. Here is an example of a collection of documents which consist of one sentence:

1. Mother bought a kilogram of green apples for a pie.
2. I like baking an apple pie.
3. I do not like green apples.

and below there is an index for this collection:

Word	Nd	Inverted list
mother	1	$\rightarrow < 1, 1 >$
bought	1	$\rightarrow < 1, 1 >$
a	1	$\rightarrow < 1, 2 >$
kilogram	1	$\rightarrow < 1, 1 >$
of	1	$\rightarrow < 1, 1 >$
for	1	$\rightarrow < 1, 1 >$
apple	1	$\rightarrow < 2, 1 >$
apples	2	$\rightarrow < 1, 1 >, < 3, 1 >$
I	2	$\rightarrow < 2, 1 >, < 3, 1 >$
baking	1	$\rightarrow < 2, 1 >$
an	1	$\rightarrow < 2, 1 >$
pie	2	$\rightarrow < 1, 1 >, < 2, 1 >$
do	1	$\rightarrow < 3, 1 >$
not	1	$\rightarrow < 3, 1 >$
like	2	$\rightarrow < 2, 1 >, < 3, 1 >$
green	2	$\rightarrow < 1, 1 >, < 3, 1 >$

2 Optimization of an index efficiency

Inverted file indexes are widely used in text search engines that deal with both a small and a very huge amount of data such as messages, web pages, articles, books etc. The main properties of an index structure is to provide

- optimal time complexity for operations within index and data retrieval
- optimal space complexities
- retrieval effectiveness

In the next subsections we will show the strategies that one can use to optimize the efficiency of an inverted file index.

2.1 Retrieval effectiveness

Retrieval effectiveness represents how relevant and effective a query can be. Since the collection may includes completely different documents a high retrieval effectiveness can not be guaranteed, because there is a big probability that a query will return documents which satisfy the query but are irrelevant for a searcher. In order to optimize effectiveness of queries the ranking strategy can be applied.

Ranking algorithms for a query can determine the documents that are more relevant by analyzing the frequencies of query parameters among the documents.

2.2 Items for indexing

A significant advantage from using indexes is that one has fast data retrieving and manipulation. The other side of such an advantage is that index needs additional space on disk to be stored. Consider the collection we used to demonstrate the inverted index (table above). It can be observed that the index includes several forms of a word and stop words that have big frequencies as on document level as on word level. In such a case the parsing of a documents into tokens can be applied. The parsing strategy provides removing of stop words from document before indexing and also eliminating of different forms of a word into one token for indexing, that has a common prefix of all forms. In such a way all documents in a collection appear to be filtered before indexing and hence, this reduces the size of an index as well as speeds up queries themselves. Usually for a typical collection of documents an index with parsing and ranking has size around 40% of original data.

Of course the parsing strategy can not be applied for every index, because it has a strong dependence on the types of queries required for a search engine. From one side if queries must provide with a list of documents that are highly related to query parameters, then the parsing of documents is reasonable for a search engine, because along with ranking strategy that improves retrieval effectiveness, it will give the best outcome. From the other side, if it is expected that queries must provide an outcome with a perfect match, then parsing of documents will decrease the retrieval effectiveness, because query outcome will include a lot of irrelevant results.

2.3 Index compression

Compression of data in index reduces it to size of around 10% of original data.