Bases de Datos 2, Resumen 9
Miguel Ku Liang - 2019061913

Inverted indexes: Types and techniques

1. Introduction

They store for each word, the documents that it appears in.

2. Inverted indexes

You can store word positions and word frequency. Some implementations store two inverted lists, one for document lists and the other for full word position lists. Other store metainformation about each hit or storing separately the lexicon.

2.1 Compression

Storing inverted lists uncompressed wastes space. Simple byte-aligned methods are the preferred choice for compression:

- Variable length integers
- Elias gamma
- Golomb-Rice
- Delta coding

2.1.2 Construction

Constructing an inverted index uses a lot of memory, disk space and CPU. The best way is to construct in-memory inversions of limited size, store them to disk and merge them to produce the final inverted index.

3. Inverted index techniques

3.1 Document preprocessing

3.1.1 Lexing

Process of converting a document into a list of tokens.

3.1.2 Stemming

Transforming each word to its morphological root and indexing that.

3.1.3 Stop words

Common words to ignore when indexing the document.

3.2 Query types

3.2.1 Normal

Any query that is not explicitly indicated by the user to be a specialized query of one type.

3.2.2 Boolean

Queries where the search terms are connected with boolean operators.

3.2.3 Phrase

Queries used to find documents that contain the given words in the given order.

3.2.4 Proximity

Queries that match documents where term1 is within n words of term2.

3.2.5 Wildcard

Queries with inexact matching. There are whole-word wildcards and in-word wildcards.

3.3 Result ranking

Few documents from a query result are used, so ranking the search results is important. Some factors to consider are:

- Number of documents the query term is found in
- Number of times the term is found in the document
- Total number of documents in the collection
- Length of the document
- Length of the query

3.4 Query evaluation optimization

The main things to optimize are the quality of the results returned and the time taken to process the query. There are two basic methods of evaluating queries: term-at-a-time and document-at-a-time.