Inverted indexes: Types and techniques

Inverted indexes

Documents are normally stored as lists of words, but inverted indexes invert this by storing for each word the list of documents that the word appears in, hence the name “Inverted index”. If you want to support phrase and proximity queries you need to store word positions for each document, i.e. the positions that the word appears in. The granularity of a position can range from byte offset to word to paragraph to section, but usually it is stored at word position granularity.

Compression

There are many ways to store the lists in a more compact form. They can be divided into two categories depending on whether the number of bits they use for coding a single value is always a multiple of 8 or not.

- Variable length integers: Instead of using 32 bits to store every value, we can use variable length integers that only use as many bytes as needed.
- Elias gamma: Elias gamma coding [Eli75] consists of the number written in binary, prefixed by N zeros, where N = number of bits in the binary representation − 1.
- Golomb-Rice: Golomb coding [Gol66] differs from Elias codes in that it is a parameterized one.
- Delta coding: We can increase our compression ratios for the lists of numbers significantly if we store them delta coded. This means that instead of storing absolute values, we store the difference to the previous value in the list.

Construction

In-memory inversion is not feasible for large collections so in those cases we have to store temporary results to disk. Since disk seeks are expensive, the best way to do that is to construct in- memory inversions of limited size, store them to disk, and then merge them to produce the final inverted index.

Inverted index techniques

Document preprocessing

Documents are normally not indexed as-is, but are preprocessed first. They are converted to tokens in the lexing phase, the tokens are possibly transformed into more generic ones in the stemming phase, and finally some tokens may be dropped entirely in the stop word removal phase.

Lexing

Lexing refers to the process of converting a document from a list of characters to a list of tokens, each of which is a single alphanumeric word.

Stemming

Stemming means not indexing each word as it appears after lexing, but transforming it to its morphological root (stem) and indexing that instead. The most common stemming algorithm used for the English language is Porter's.

Stop words

Stop words are words like "a", "the", "of", and "to", which are so common that nearly every document contains them. A stop word list contains the list of
words to ignore when indexing the document collection.

Query types

Normal

A normal query is any query that is not explicitly indicated by the user to be a specialized query of one of the types described later in this section.

Boolean

Boolean queries are queries where the search terms are connected to each other using the various operators available in Boolean logic [Boo54], most common ones being AND, OR and NOT. Usually parentheses can be used to group search terms. A simple example is madonna AND discography, and a more complex one is bruce AND mclaren AND NOT ("formula one" OR "formula 1" OR f1). These are implemented using an inverted index as follows:

- NOT A pure NOT is usually not supported in Full- Text Search (FTS) implementations since it can match almost all of the documents. Instead it must be combined with other search terms using the AND operator, and after those are processed and the preliminary result set is available, that set is then further pruned by eliminating all documents from it that contain the NOT term. This is done by retrieving the document list for the NOT term and removing all document ids in it from the result set.
- OR The query term1 OR term2 OR . . . term n is processed by retrieving the document lists for all of the terms and combining them by a union operation, i.e., a document id is in the final result set if it is found in at least one of the lists.
- AND The query term1 AND term2 AND . . . term n is processed by retrieving the document lists for all of the terms and combining them by an intersection operation, i.e., a document id is in the final result set if it is found in all of the lists.
- Unlike the OR operation which potentially expands the result set for each additional term, the AND operation shrinks the result set for each additional term. This allows AND operations to be implemented more efficiently.

Phrase

Phrase queries are used to find documents that contain the given words in the given order.

Proximity

Proximity queries are of the form term1 NEAR(n) term2, and should match documents where term1 occurs within n words of term2.

Wildcard

Wildcard queries are a f orm of fuzzy, or inexact, matching. There are two main variants:

- Whole-word wildcards, where whole words are left unspecified.
- In-word wildcards, where part of a single word is left unspecified.

These can be handled by first expanding the wildcard word to all the words it matches and then running the modified query normally with the search term replaced by (word1 OR word2 OR . . . wordn).

Result ranking

Traditionally, the information retrieval field has used a similarity measure between the query and a document as the basis for ranking the results. The theory is that the more similar the query and the document are to each other, the better the document is as an answer to the query. Most methods of calculating this measure are fairly similar to each other and use the factors listed below in various ways.

Some of the factors to consider are: the number of documents the query term is found in (ft), the number of times the term is found in the document (fd,t), the total number of documents in the collection (N), the length of the document (Wd) and the length of the query (Wq).

If a document contains a few instances of a rare term, that document is in all probability a better answer to the query than a document with many instances of a common term, so we want to weigh terms by their inverse document frequency (IDF, or 1 ft). Combining this with the term frequency (TF, or fd,t) within a document gives us the famous TF × IDF equation.

Query evaluation optimization

The query evaluation optimization research literature, however, ignores the existence of this hybrid query type almost completely and discusses just plain ranked queries, i.e., queries with no particular syntax which are supposed to return the k most relevant documents as the first k results. This is unfortunate since normal ranked queries are almost useless on huge collections like the Internet, because almost any query besides the most trivial one needs to use extended query methods like disallowing some words (Boolean AND NOT) and matching entire phrases (phrase query) to successfully find the relevant documents from the vast amounts in the collection.