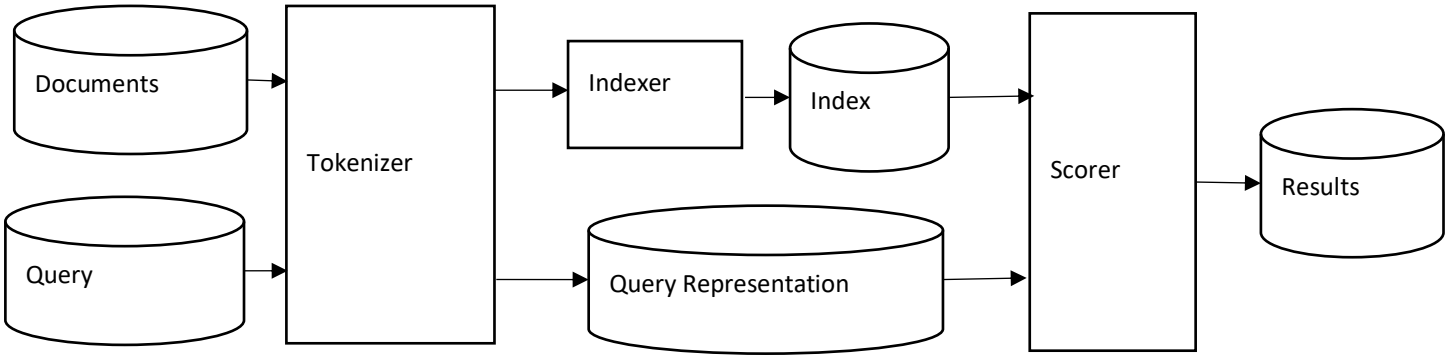
Text Retrieval and Search Engines

Text Retrieval System Architecture



Tokenization

Normalize lexical units such that words with similar meanings are mapped to the same indexing term and map all inflectional forms of words to the same root form (**Stemming**).

Indexing

Create a data structure to enable fast search, this can mean manipulation of the documents themselves or creating new structures that map to the documents. The most popular of these is the **inverted index**.

Inverted Index

An inverted index, also referred to as a postings file or inverted file, is a structure that stores information about content together with its location in a document collection. Creation of an Inverted Index is a two-step process requiring the creation of a dictionary or lexicon which is used to generate the index. Here is a simple example.

Example Documents	Dictionary	Inverted Index
Doc1 = “To be” Doc2 = “Or not to be”	“to” “be” “or” “not”	“to”={Doc1, Doc2} “be”={Doc1, Doc2} “or”={Doc2} “not”=a{Doc2}

Generally the dictionary will be of modest size and can be stored in memory but the index file will be huge. Typically a **sort-based method** is used to generate the index file and then some form of compression of is applied.

Zipf’s Law

An empirical law which states that the frequency *F(w)* of a word in a language is inversely proportional to its rank *r(w)* in the frequency table.

$$F(w) = \frac{C}{r(w)^{\alpha}} \; where \; \alpha \approx 1, C \approx 0.1$$

Zipf’s Law is sometimes used to compress the index of a structure by filtering out **stop words.**

Sort-Based Index File Generation

1. Collect local (termID, docID, freq) tuples.
2. Sort local tuples (to make “runs”).
3. Pair-wise merge runs.
4. Output inverted file.

Inverted Index Compression

Term Frequency Compression – Small numbers tend to occur more frequently, use fewer bits for small integers at the cost of more bits for large integers.

Document ID Compression – Use the document gap or "d-gap" instead of individual document id. This is feasible due to sequential access.

Frequently used methods include Binary Code, Unary Code, γ-code and δ-code.

Unary Code

Unary coding represents a number n as n ones followed by a zero **or in the case of strictly positive integers** as (n-1) ones followed by a zero.

n (non-negative)	n (strictly positive)	Unary
0	1	0
1	2	10
2	3	110
3	4	1110

γ-code

To encode a number n in γ-code you must calculate two values x and y such that $2^x + y = n$

1. Calculate $x = floor(log_2(n))$.
2. Encode x using the unary method **for non-negative numbers**.
3. Calculate $y = n - 2^x$.
4. Express y as a binary number using x bits.
5. Append y to x.

n	x	y	unary x	binary y (in x bits)	γ-code
1	0	1	0	1	01
2	1	0	10	0	100
3	1	1	10	1	101

To decode a γ-coded number:

1. Read and count the number of ones c until the first zero is reached.
2. Read the remaining digits and interpret as a binary number b.
3. The encoded number is then $2^c + b$.

δ-code

This encoding is similar to γ-code but the unary prefix is replaced with its γ encoding:

1. Calculate $x = floor(log_2(n))$.
2. Encode (x+1) using the γ-code method.
3. Calculate $y = n - 2^x$.
4. Express y as a binary number using x bits.
5. Append y to x.

n	x	y	γ-code of (x+1)	binary y (in x bits)	δ-code
1	0	1	01	1	011
2	1	0	100	0	1000
3	1	1	100	1	1001

To decode a δ-coded number:

1. Read and count the number of ones c until the first zero is reached.
2. Read the next 2^c digits as the binary number b1.
3. Read the remainder of the digits as the binary number b2.
4. Calculate p the highest power of 2 in the original number using $p = 2^c + b1 - 1$
5. The encoded number is equal to $2^p + b2$.

Scoring Function

The scoring function is used to rank documents returned by a query and is mostly made up of a number of different weight factors. The weight factors of the document and the query, $f_d(d)$ and $f_q(q)$ respectively, are known in advance. In addition the cumulative weights of each term in the query with respect to both the document and the query must be calculated by combining them in the function $h\big(g(t_1, d, q), \dots, g(t_k, d, q)\big)$.

The general form of a scoring function is thus:

$$f(q, d) = f_a(h\big(g(t_1, d, q), \dots, g(t_k, d, q)\big), f_d(d), f_q(q))$$

The Cranfield Evaluation Methodology

The Cranfield evaluation methodology was developed in the 1960's to test the accuracy, efficiency and usability of a text retrieval system in a laboratory setting. The basic principle is to build reusable test collections for study incorporating document collections, query samples and relevance judgements. This test collection can then be used to compare different text retrieval systems.

Precision

$$P = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ retrieved\ documents}$$

Recall

$$R = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ relevant\ documents}$$

F-Measure

In an ideal situation precision = recall = 1.0 in reality high recall tends to be associated with low precision. The overall accuracy of a text retrieval system can be more reliably determined using a measure which combines both parameters. One such measure is the F-Measure:

$$F_\beta = \frac{1}{\frac{\beta^2}{\beta^2+1}\frac{1}{R} + \frac{1}{\beta^2+1}\frac{1}{P}} = \frac{(\beta^2 + 1)P * R}{\beta^2 P + R}$$

The parameter β can be varied but is often set to a value of 1.

$$F_1 = \frac{2PR}{P + R}$$

Average Precision of a Ranking System

The significant points in a plot of precision versus recall are those which represent a new relevant document being retrieved. These points can be used to determine an average precision for a system, assuming there are a total number of relevant documents N_{rel} which are returned at various ranks r then:

$$Average\ Precision = \frac{\sum_{i=1}^{N_{rel}} \frac{i}{r_i}}{N_{rel}}$$

For example if a collection contains 5 relevant documents which are returned at ranks 1, 3, 9, 25 and 100 then the average precision is:

$$AP = \frac{\frac{1}{1} + \frac{2}{3} + \frac{3}{9} + \frac{4}{25} + \frac{5}{100}}{5}$$

When there are a large number of documents the average precision is often determined at a cutoff point assuming that any remaining relevant documents not yet returned will contribute a value of zero to the average precision.

Mean Average Precision

The precision of a system may be described using the mean of the average precision over a set of queries. Often the geometric mean is preferred to the arithmetic mean because it is more sensitive to small numbers and thus is more effective at indicating the accuracy of a system looking for rare words.

Mean Reciprocal Rank

In the special case where only one relevant document is retrieved the average precision is equal to the reciprocal of its rank *r* so:

$$AP = 1/r$$

Multi-Level Relevance Judgements

In the case where documents have different relevance levels relevance testing of the retrieval system needs to take this into account. One way of determining this is using the cumulative gain from every document retrieved. If each document has a relevance score these scores can be added with each document returned but they must be "discounted" based on their rank because very relevant documents retrieved with low rank are less useful. The Discounted Cumulative Gain at rank *x* is defined as:

$$DCG@rank\ x = rel_1 + \sum_{i=2}^{x} \frac{rel_i}{\log(i)}$$

Where rel_1 is the relevance of the first document retrieved and rel_i is the relevance of the i^{th} document retrieved.

The DCG is usually normalized by dividing by the **Ideal DCG** which results from all the most relevant documents being retrieved first, followed by the next relevant and so on.

$$nDCG = \frac{DCG}{IdealDCG}$$

Statistical Significance Testing

A number of statistical tests are available to compare the success of different text retrieval systems. One common method is using the "null hypothesis" which determines the likelihood that the differences between two systems are due to chance alone by calculating the p-value of the results.

Pooling

Pooling is a method for generating subsets of a document collection in order to make a comparison more efficient. One way to do this is to use a number of different ranking systems to return a small number of documents and then combine all the returned documents into the subset.