Google: data structures and algorithms

Petteri Huuhka

Department of Computer Science
University of Helsinki
Petteri.Huuhka@helsinki.fi

Abstract. The architecture of Google's search engine is designed to handle large amounts of web pages and queries. The search engine uses data structures optimized for large data sets and adding more servers to the system easily increases its performance of answering queries. Google's PageRank algorithm uses the link structure of the WWW to calculate the qualities of web pages. Using the link structure in the ordering of the search results makes it harder for anyone to manipulate Google's search results but not impossible. Google also has some means to prevent manipulation and maintain the quality of the search results.

1 Introduction

Nowadays the WWW is very large consisting of billions of web pages and other items such as images, documents and music. At the time of writing Google is a very popular search engine on the Internet. Nielsen//Netratings has measured that almost half of searches done in the US are done with Google, Yahoo being next with 23,8% [Nie06]. From the beginning Google was designed to handle large datasets and many search queries per second. Data structures, data storage and algorithms are designed so that the system is scalable both in the number of pages index per second and the number of queries handled per second [BrP98].

The task of search engines to provide quality search results is difficult because of the large number of documents on the WWW. The quality of Google is partly based on the PageRank algorithm. It is used to calculate a measure of quality of web pages. The position of a page on the search results page is partially dependent on the PageRank value so that quality pages have a higher position. PageRank is also meant to prevent people manipulating the search results. In this it succeeds only partially as there are still ways to manipulate Google both by words on a page and through features of PageRank. [PBM98]

Section 2 introduces some of the data structures used by Google. Google's PageRank algorithm and is presented in the section 3. Section 4 discusses some weaknesses of Google's algorithms.

2 Data structures

The Google File System

In 2003 Google replaced its original BigFiles file system with the Google File System (GFS). Like BigFiles it is used to store files in a distributed matter. GFS consists of a single master and many

chunk servers. Each file is broken into large chunks (e.g. 64 MB) identified with chunk handles. And each chunk is distributed into multiple chunk servers (e.g. three servers) for reliability. When a client wants to read or write a segment of a file it calculates the chunk index using the offset of the segment and the size of a chunk (chunk index=segment offset/chunk size). The master keeps a mapping of a filename and a chunk index into a chunk handle. The master also keeps in memory which chunks each chunk server has. The client can ask the master for the chunk handle and the chunk servers that have the chunk by providing the filename and the chunk index. [Goo7]

The idea of GFS is to gain high performance and fault tolerant system using inexpensive hardware that often fails. GFS is optimized for storing large files, appending large amounts of data to the end of the file and making large streaming reads and small random reads. The system also features atomic concurrent appending of records into a single file with minimal synchronization overhead. This feature is used, for example, for a queue with many producers and one consumer. [Goo7]

Repository

Google stores the full HTML of all the web pages in a repository. Every web page is stored as a packet that includes a docID identifying the document, length of the URL, length of the page, URL and HTML of the page (see figure 1). These packets are stored in a compressed form to save space. The repository is simply composed of a stack of triplets of sync, length of the compressed packet and the packet itself. There are no other data structures in the repository. To find a web page in the repository requires either a pointer to the file or going through the whole file. [Goo7]

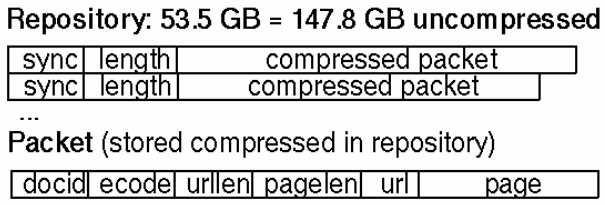


Figure 1 Data contents of the repository and a packet.

Hit Lists

A hit corresponds to an occurrence of a word in a web page. It does not include the word itself or the wordID corresponding to the word, only the properties using 16 bits: capitalization (1 bit), font size (3 bits) and position (12 bits) (see figure 2). A fancy hit is a word that occurs in an URL, title, anchor text or meta tag. The font size of 7 means that a hit is a fancy hit and it therefore uses 4 bits from the position bits to code the type of a fancy hit. The type can be an URL, title, anchor text or meta tag. If a hit is a fancy hit and the type indicates that it occurs in an anchor text, another 4 bits from the position bits are used for a hash of the docID the anchor occurs in.

The font size is relative to the rest of the document and can have values from 0 to 6 as font size 7 is reserved for fancy hits. The position is the position of the word in the document counting from the beginning. If the position is higher than 4095 it is labeled as 4096 because there are only 12 bits available. A hit occurring in an anchor text is also stored with the hits of the document that is linked and not only with the hits of the document it occurs in. This is because Google believes that an anchor text describes the linked document well. Query terms are not only matched against the words in a document. They also match a document if the words occur in the links pointing to that document. Also the position of an anchor hit is its position in the anchor text instead of the position in the document.

A hit list consists of the number of hits in the list and the list of hits. The number of bits used for the number of hits depends on the context. [Goo7]

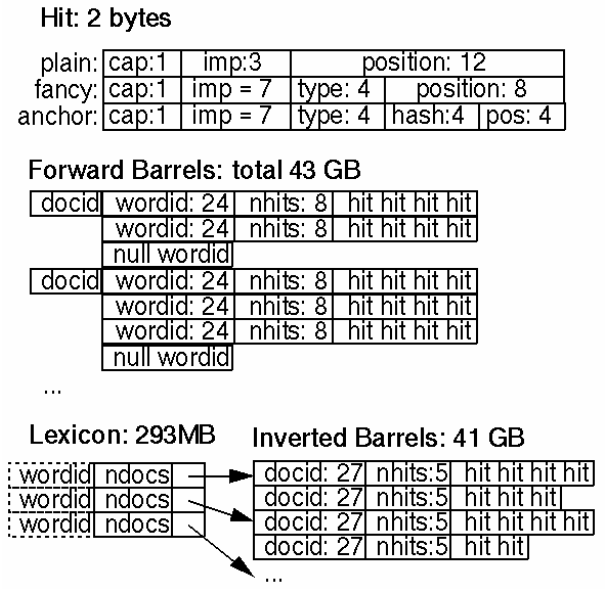


Figure 2 Data contents of a hit, a barrel of the forward index and a barrel of the inverted index.

Forward Index

The forward index (see figure 2) consists of barrels and each barrel has its own range of wordIDs. A record in a barrel consists of a docID followed by wordIDs with their hit lists. The wordIDs in records of a barrel all belong to the wordID range of the barrel. It is probable that all words in a document do not fall into a wordID range of a single barrel. For every document, a record is created in each barrel that includes wordIDs corresponding to the words in the document. It is not necessary to store wordIDs as such in the barrels because all wordIDs are in the wordID range of the barrel. The lower bound of the wordID range can be subtracted from the wordID and the result is used instead of the original wordID. WordIDs use 24 and 8 bits is used for the number of hits in the hit list for a document. [Goo7]

Inverted Index

The inverted index (see figure 2) is used to find all documents that contain a given word. The index is made from the forward index by sorting the contents of the barrels in the order of wordIDs. As a result the inverted index consists of docID lists, one list for each word and grouped into barrels by wordIDs. A docID list contains docIDs each followed by a hit list of the document. The lexicon contains a data structure that maps words into pointers pointing to the beginning of the docID list of the word in the inverted index. The docID lists are sorted by docID to make merging of multiple docID lists simpler when answering to queries. Merging is used to find out which documents include all query terms. Hits that occur in a title or an anchor text are stored in the normal barrels but also in separate barrels. When answering a query these separate barrels are processed first. If using only barrels with words from titles and anchor texts gives enough results it is not necessary to go through the rest of the barrels. This is one of the ways Google gives weight to words in titles and anchor texts. [Goo7]

Sorting a barrel is faster if all its elements fit into the main memory. The size of a barrel can be tens or hundreds of gigabytes so it may not fit into a main memory at once. The barrels can be further

divided into buckets according to the wordIDs and the docIDs. The buckets are then sorted one at the time and the sorted buckets are joined. [Goo7,Goo8]

The forward index and the inverted index can be split into multiple forward indexes and inverted indexes according to the docIDs. Each forward index and inverted index contains information only of documents in some range of docIDs. This speeds up both sorting of the inverted index and answering queries because it allows these actions to be more efficiently parallelized. A query from the user is sent to multiple servers. Each server uses a single inverted index to search for docID lists. DocID lists from all servers are merged to get the final results. [Goo8]

3 PageRank

3.1 Motivation

To provide the user with relevant results is a huge problem for search engines. What makes this difficult is that the user may not be familiar with search engines and does not know how to choose good query terms. The user also may not be sure what he or she is looking for. As a consequence the user enters only a couple of query terms. With this little information the search engine has to sort the web pages containing the query terms in the order of their relevance [BhH98]. To provide quality results Google has invented the PageRank algorithm. It also helps to give good results when only little information of the user's needs is available.

3.2 Algorithm

Introduction

PageRank is used to calculate a quality rank among nodes in a graph, or in the case of WWW among web pages in a collection. The quality of a web page does not depend at all on the content of the page. Only the link structure of the collection is used. Every link on a page is a vote that the linked page contains quality content. In this sense, PageRank resembles a citation counting technique where the rank of a page would be the number of links made to it. PageRank, however, does not treat all links equally. A link from a high-ranking page is more valuable than a link from a page with a lower rank.

The total quality, i.e. PageRank, of a page is a sum of all the values of the links that point to it. The value of a link depends on the properties of the page the link is on: PageRank is divided by the total number of links on the page. As a result any two web pages can be compared to determine which one has a higher rank.

A simplified equation of PageRank is

$$PR(u) = c \sum_{v \in B_u} \frac{PR(v)}{N_v} \quad (1)$$

where PR(u) is the PageRank of a page u, B_u is a set of pages pointing to u, N_v is the number of links on page v and c is a normalization factor. Factor c is used so that the total sum of rank on the graph is constant. The sum of PageRank values can be normalized to any constant but in this article the total sum of rank on the graph is 1. It can be seen from the equation 1 that a PageRank of a page depends on the PageRanks of other pages.

Calculation

There are at least two ways to calculate PageRank values. One possibility is to form an adjacency matrix A of size N times N, where N is the number of web pages on the graph. If there is a link from page u to v, $A_{u,v} = \frac{1}{N_v}$ as in equation 1 and $A_{u,v}=0$ otherwise. In the elements $A_{u,v}$ of the adjacency matrix A, u is the row and v is the column. The PageRank equation can be presented as an equation cR=A*R, where * means matrix multiplication, R is a column vector containing PageRank values for pages and c is a constant.

Let PR(x) be the components of the vector R. The equation for the first element of R, PR(0), would be
$cPR(0)=A_{0,0}PR(0)+A_{0,1}PR(1)+A_{0,2}PR(2)+\ldots+A_{0,N}PR(N)$.

Solving the eigenvector R in equation cR=AR gives PageRank values in the vector R. There may be more than one eigenvector-eigenvalue-pair solution (even complex ones) but we want to find the dominant eigenvector with eigenvalue c. Finally PageRank values are normalized so that the total sum of rank is 1. Calculation of eigenvector and eigenvalue is out of the scope of this paper.

Another way to calculate PageRank values is to iterate the equation 1. At the beginning PR(v) values can be assigned with almost any random values, some constant or with good guesses of the actual value. During iteration the PR'(u) values will eventually converge with the correct PageRank values.

On each iteration cycle a page basically distributes its current PageRank equally to all the pages that it links to. In the following iteration algorithm the sum of PageRank other pages distribute to a single page is calculated. An example of iterative calculation of PageRank is illustrated in the figure 3.

Iteration is performed as follows

1. New PageRank values PR'(u) for each page u are calculated using the PageRank values PR(v) for each page v calculated on the previous iteration. On the first iteration PR(v) values can be, for example, a constant 1/N, where N is the number of pages on the graph.

   $$PR'(u) = \sum_{v \in B_u} \frac{PR(v)}{N_v}$$

   This step can be done also using matrix multiplication

   $$PR' = A * PR$$

   where * means matrix multiplication, A is an adjacency matrix, PR is a column vector of PageRank values of the previous iteration and PR' is a column vector holding the new PageRank values.
2. Normalize PR'(u) values (or matrix PR') so that the total sum of rank on the graph is 1.

   $$\sum PR'(u) = 1$$
3. Calculate L1-norm of vector PR'-PR. L1-norm is the sum of absolute values of vector elements. In this case we calculate the absolute values how much the PageRank value of each page changed from the previous iteration and sum these together. $q = ||PR' - PR||_1 = \sum | PR'(u) - PR(u) |$

4. Assign to PR(v) the value of PR'(v), for each page v.
5. If q>e goto step 1

Iteration ends when the result is good enough i.e. q=<e.

Random surfer model

The justification for using PageRank for ranking web pages comes from the random surfer model. PageRank models the behavior of a web surfer who browses the web. The random surfer is placed on the graph and he always moves to the next page by choosing a link at random from the page he is currently on. The number of times the surfer has visited each page is counted. PageRank of a given page is this number divided by the total number of pages the surfer has browsed.

PageRank is the probability that the random surfer moves to a given page. Comparing to the equation 1, PR(v) is the probability of the surfer being on the page v that has a link to the page u and $\frac{1}{N_v}$ is the probability of the surfer choosing the link to page u.

Dangling links

Some of the pages on the web have no links to other pages. Some of the links in the database have been picked up from fetched web pages but the web pages those links are pointing to are not yet fetched. If a page has not been fetched it is considered to be a page that has no links to other pages. These pages, called dangling links, without any links are a problem because their PageRank cannot be redistributed to other pages. Their PageRank "leaks out" of the graph and therefore gets lost.

To prevent this, all pages without links and links to these pages are removed before calculation of PageRank values (see the example in figure 4). Removing pages from the graph can cause other pages to lose all their links and become dangling links themselves. This is why removing dangling links is an iterative process. On every iteration all dangling links are identified before they are removed, and the number of iterations needed is counted. It is not necessary to remove all dangling links. A couple of iterations will remove most of them.

After removing dangling links PageRank values can be calculated for the remaining web pages. When this is done, the removed dangling links are returned to the graph with PageRank value of 0. The PageRank algorithm is then iterated as many times as it took iterations to remove dangling links so that every page gets a rank.

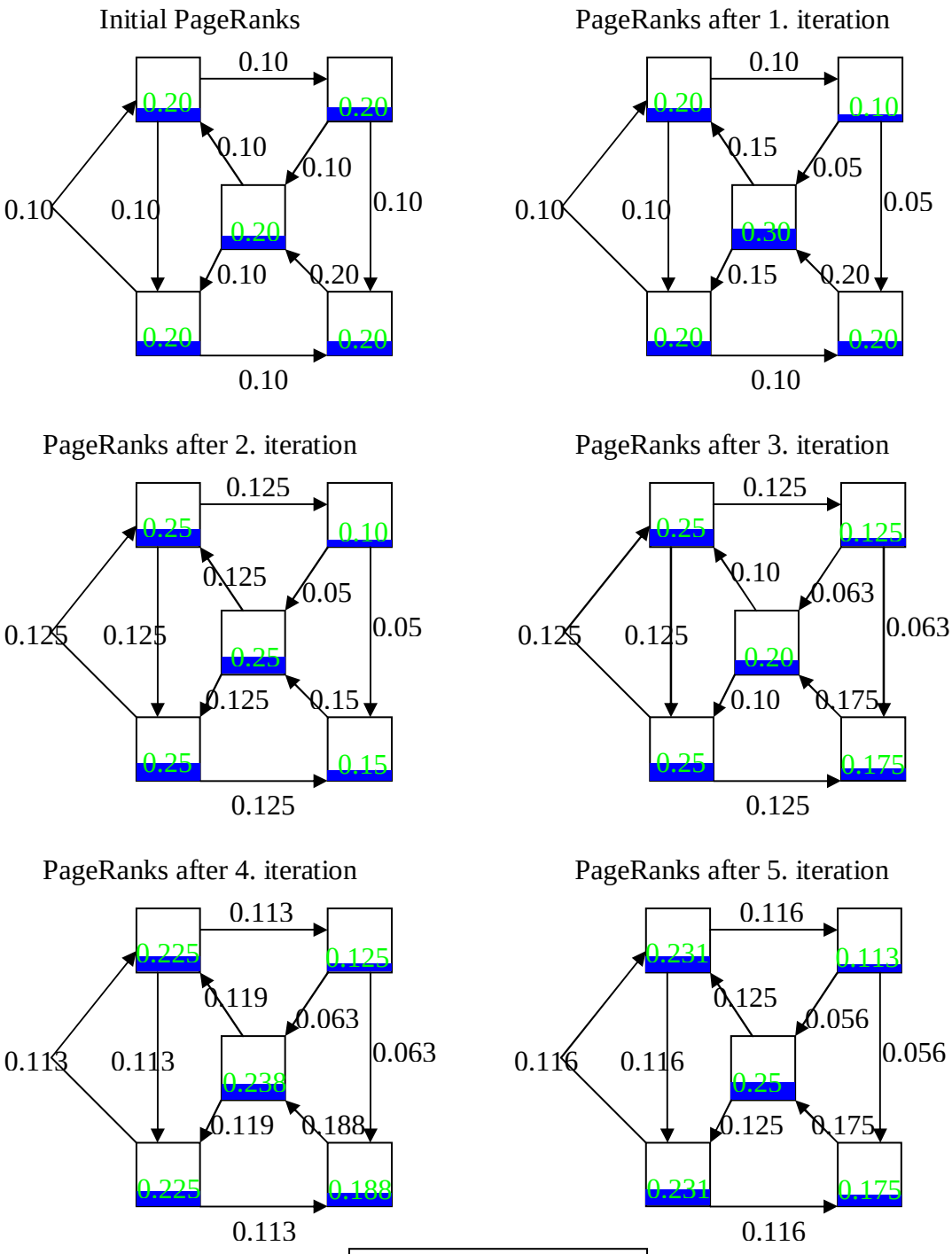


Figure 3 The graphs illustrate how the PageRank values converge to the final PageRank values when using iteration. The values in the nodes are the PageRank values of the nodes and the values in the edges are the values of the links in the next iteration. The value is also the amount of PageRank a page distributes to the linked pages.

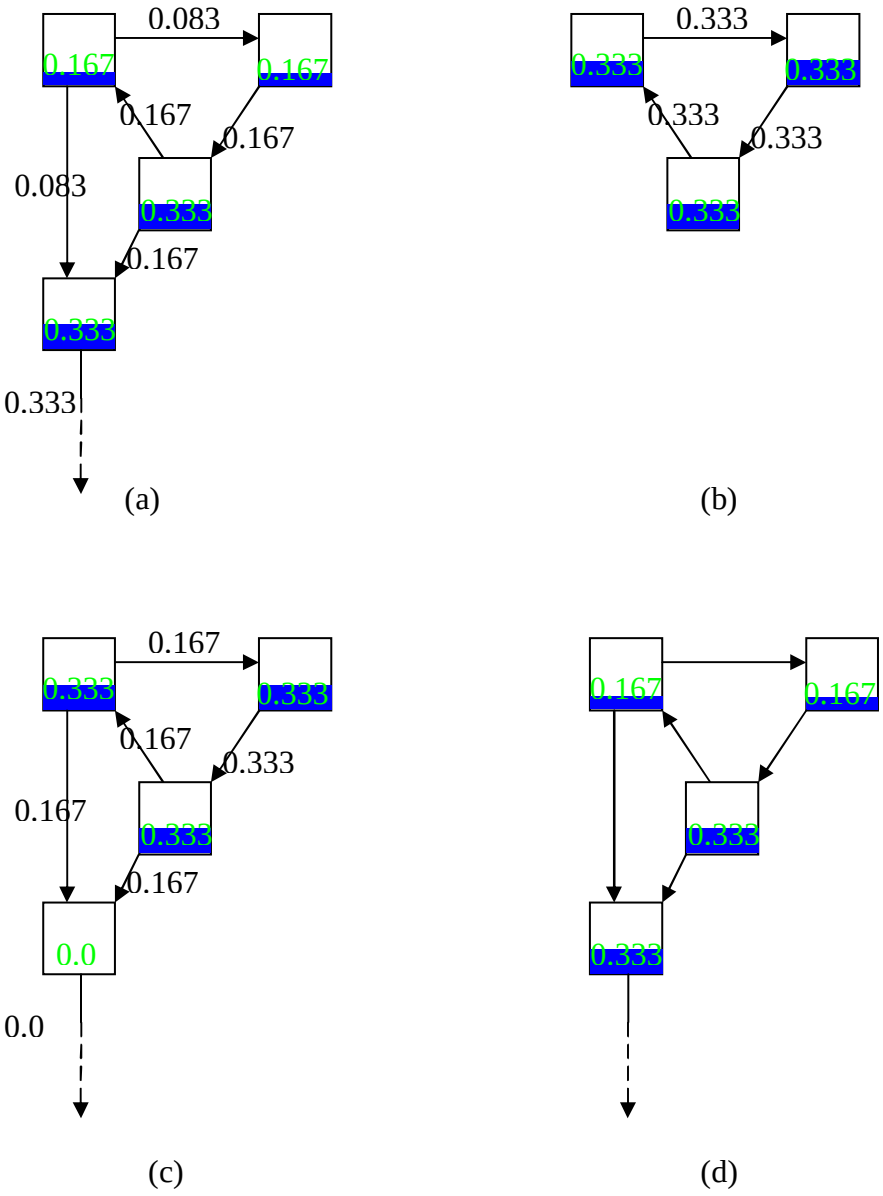


Figure 4 (a) a graph with a dangling link leaking PageRank, (b) the dangling link is removed in one iteration of the dangling link removal algorithm and final PageRank values have been iterated, (c) the dangling link is returned with a PageRank of 0, (d) PageRank algorithm has been iterated once to get the final PageRank values for the original graph.

Rank sink

A problem in the simplified equation is what the authors call a rank sink. Think of a subgraph of interconnected nodes with no links to the rest of the graph. Also every node links at least one other node, no dangling links, so that rank does not “leak from the graph” and thereby disappear. One or more nodes in the rest of the graph have a link or links to some nodes of the subgraph. During iteration these nodes are pouring their rank to the subgraph. As there are no links out of the subgraph the total sum of rank on the subgraph increases. The members of the subgraph thus get higher rank than they are supposed to. The simplest form of a rank sink is two nodes both linking the other and a link from some other node to one of these nodes.

A way to narrow this down and that way reduce the problem is introducing a dumping factor to the PageRank algorithm.

$$PR(u) = \frac{(1-d)}{N} + d \sum_{v \in B_u} \frac{PR(v)}{N_v} \quad (2)$$

$$PR(u) = dE(u) + d \sum_{v \in B_u} \frac{PR(v)}{N_v} \quad (3)$$

, where d is a damping factor (a constant real number between 0 and 1) and E(u) is a constant specific to the page u.

The equations 2 and 3 are two versions of the algorithm. Equation 2 is a special case of equation 3 where $dE(u) = \frac{1-d}{N}$, where N is the number of nodes in the graph.

In terms of a random surfer there is a probability of d that the surfer follows some link on the page he is at. On the other hand there is a probability of (1-d) that the surfer jumps to a random page. Because it is equally probable for the surfer to jump to any page on the graph the probability of the surfer jumping to a certain page is $\frac{1-d}{N}$. [PBM98,BrP98]

4 Weaknesses of Google

Keyword spamming

As Google is based on query terms like most search engines it is possible to cheat Google by adding words in to pages. If a user makes a query that includes some of the added words Google gives this page as a result even though the page has nothing to do with the words. The extra words make the search engine see the page differently than the user. [Wik1,Goo1]

The extra words can be hidden from the user by making the text small or the same color as the background, hiding the text behind an image or placing it outside the screen using absolute positioning in CSS, or putting the text in an HTML element that is not visible to the user (e.g. noscript) [Wik1,Wik2]. Irrelevant words can also be hidden in the keywords meta tag but not many search engines support this meta tag because of its abuse [Wik1,Sul02]. Google claims that it analyses only content that the user sees on the page. This suggests that hiding extra words does not work. [Goo2]

Link spamming

To increase its PageRank a web page needs to be linked from other web pages with a high PageRank [PBM98]. There are many possibilities how to get linked from another page.

Blogs, guest books and wikis have a possibility of users adding comments. These comments can also include links. If the page owner does not check the submitted comments before they are shown on the page it is possible for anyone to add advertisement and other links. [Wik3]

Page hijacking

Sometimes the same content can be found from many URLs. For example the host part of the URL can be different if the same pages are found on both www.host.com and host.com (i.e. the same without www.). A page can also be mirrored to protect it from a hardware failure. A mirrored page can prevent a page being censored or removed completely from the net. Many sites use mirroring as a way to balance load. [Wik4]

Some search engines spare the users from seeing duplicate or similar enough pages as separate results. Therefore the search engine has to decide which URL of the duplicate or similar pages it is going to show to the user [Wik5]. Google uses PageRank as a part of the decision making process; the URL with the higher PageRank is likely to be chosen [SD1].

This feature can also be used to have an URL of a malicious webmaster to be shown instead of the real URL of a page. A duplicate page can be made by copying the contents manually, using a server-side-scripting to automatically copy the contents or issuing "302 Found" as an HTTP response code [Wik5,SCH05. In Google this is likely to happen only when the PageRank of the original page is lower than that of the malicious copy. For a respectable web page this is rare [SD1].

The "302 Found" response code means that the fetched page is temporarily moved to a different location and the temporary URL is provided in the Location HTTP header of the response. The contents are fetched from the temporary URL but the original URL is stored because the new location is only used temporary. [Sch05]

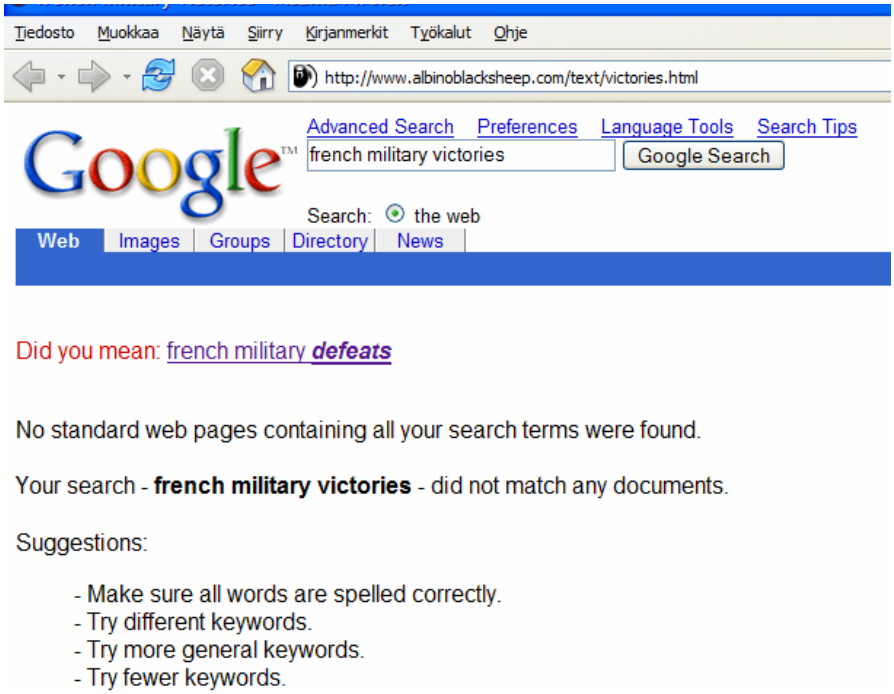
To get profit of this situation a malicious webmaster can show a different page to the user than search engines [Sch05]. When a search engine fetches a page from a server it identifies itself in the User-Agent HTTP header. Using this information a server can decide which page it returns [Wik6]. Another way is to give the same page both to the user and search engines but a redirection sends the user to another page. Javascript and refresh META tag are some of the redirection techniques. Both of them change the contents of the page but the page remains similar enough to be considered a duplicate. [Wik5,Wik7]

Google bomb

Just like Google trusts that links are good as votes of quality it also trusts that anchor text describes the linked pages. The search results may include a page even though it does not contain the query terms. The reason for this is that a page has linked the page and used the query terms in the anchor text [Wik8].

Using this technique Google can find pages that it has not yet crawled. It makes it possible to index files without textual information, e.g. videos, images, music. Also pages that are forbidden to be crawled (using /robots.txt) are indexed this way. [BrP98]

In Google bombing, a number of people make links from their web pages to a chosen page. All these links contain the same text as the anchor text. When there are enough people doing this the chosen page will be at the top of the search results when using the anchor text as a query. A famous Google bomb is the keywords "miserable failure" returning the home page of President George W. Bush. In another Google bomb the words "French military victories" return a fake Google search results page claiming that there is no page containing all the query terms and suggests searching "French military defeats" [Wik8].



All the details of how to make a Google bomb are not clear because Google does not release information about its algorithms or algorithm changes. According to Steve Lerner, creator of the French military victories parody page, it was not an intentional Google bomb. Lerner linked the page in his blog (http://www.albinoblacksheep.com/archive/february2003.html) without any advice to create links. People started to link the parody page by themselves thus raising its ranking. [Mcn04]

Solutions from Google

Google explicitly forbids most of the methods described above and forbids also other methods influencing the quality of search results [Goo1,Goo4]. As a counter measure Google can remove sites that do not follow its webmaster guidelines [Goo3]. The removal can also be for a period of time. When the site has been changed to comply Google's guidelines a reinclusion request of the site can be made to Google. The request should include who created the original site and how something like this is prevented to happen again [Cut06].

Google states that it is reluctant to manually remove sites so that they do not appear in search results [Goo4]. However a webspam team inside Google works to keep up the quality of the search engine [Cut06].

A rel attribute with nofollow value was introduced to be used with an anchor tag <a> (<a rel="nofollow" …) to reduce link spam in blogs, guest books and wikis. When this is added to an anchor Google ignores the link as a vote and therefore the linked page does not get any PageRank from this page. [Goo5]

5 Conclusion

Search engines have to manage the ever growing number of web pages as the WWW gets even larger. It also means that data structures holding the information have to cope with this or get replaced by new ones. Also the growing number of queries demands the search engines to scale for this information need.

The quality of a search engine comes both from its ability to provide the most relevant web pages to the user and from being immune to manipulation attempts. Google's PageRank tries to do the both but the competition in the search engine market challenges to improve search algorithms even further.

Acknowledgements

The author wishes to thank the reviewers Anna and Fredrik for the valuable comments concerning the contents, the conventions and the parts that needed improving.

References

[BhH98] Bharat, K. and Henzinger, M., Improved Algorithms for Topic Distillation in a Hyperlinked Environment., Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, Melbourne, Australia, 1998. [Also: http://gatekeeper.dec.com/ pub/DEC/SRC/publications/monika/sigir98.pdf].
[BrP98] Brin, S. ja Page, L., The Anatomy of a Large-Scale Hypertextual Web Search Engine, http://dbpubs.stanford.edu:8090/ pub/ showDoc.Fulltext? lang=en &doc=1998-8& format=pdf& compression=& name=1998-8.pdf.
[Cut06] Cutts, M., Ramping Up on International Webspam, 2006, http://www.mattcutts.com/ blog/ ramping-up-on-international-webspam/.
[Goo1] Google, Webmaster Guidelines, http://www.google.com/ support/ webmasters/ bin/ answer.py? answer=35769.
[Goo2] Official Google Webmaster Central Blog, Inside Google Sitemaps: Improving your site's indexing and ranking, http://sitemaps.blogspot.com/ 2006/ 02/ improving-your-sites-indexing-and.html.
[Goo3] Google, Why are sites blocked from the Google index?, http://www.google.com/ support/ webmasters/ bin/ answer.py? answer=40052.
[Goo4] Google Blog, Googlebombing 'failure', http://googleblog.blogspot.com/ 2005/ 09/ googlebombing-failure.html.
[Goo5] Google, How do I tell Googlebot not to crawl a single outgoing link on a page?, http://www.google.com/ support/ webmasters/ bin/ answer.py? answer=33582& query=nofollow& topic=&type=.
[Goo6] Google, Google Search Engine (Main page), http://web.archive.org/ web/ 20050910063558/ http://www.google.com/index.html.
[Goo7] Ghemawat, S., Gobioff, H. and Leung, S., The Google File System. SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 29–43, New York, NY, USA, 2003. ACM Press. [Also: http://labs.google.com/ papers/ gfs-sosp2003.pdf].
[Goo8] Barroso, L., Dean, J. and Hölzle, U., Web Search for a Planet: The Google Cluster Architecture. IEEE Micro, March-April 2003, pages 22-28. [Also: http://labs.google.com/ papers/ googlecluster-ieee.pdf].
[Mcn04] McNichol, T., Engineering Google Results to Make a Point., 2004, http://www.nytimes.com/ 2004/ 01/ 22/ technology/ circuits/ 22goog.html? ex=1390194000& en=90e67992909e0ad4& ei=5007& partner=USERLAND.
[Nie06] Sullivan, D., Nielsen NetRatings Search Engine Ratings, 2006, http:// searchenginewatch.com/ showPage.html? page=2156451.
[PBM98] Page, L., Brin, S., Motwani, R. ja Winograd, T., The Pagerank Citation Ranking: Bringing Order to the Web, http://dbpubs.stanford.edu/ pub/

showDoc.Fulltext?lang=en&doc=1999-66& format=pdf& compression=& name=1999-66.pdf.
[Sch05] Schmidt, C., Page Hijack: The 302 Exploit, Redirects and Google., 2005, http://clsc.net/ research/ google-302-page-hijack.htm.
[SD1] Slashdot, Re:Ugh. This is so not true., http://slashdot.org/ comments.pl? sid=143465& cid=12024599.
[Sul02] Sullivan, D., How To Use HTML Meta Tags, 2002, http://searchenginewatch.com/ showPage.html? page=2167931.
[Wik1] Wikipedia, Spamdexing, http://en.wikipedia.org/ wiki/ Spamdexing.
[Wik2] Wikipedia, Keyword Stuffing, http://en.wikipedia.org/ wiki/ Keyword_stuffing.
[Wik3] Wikipedia, Spam in Blogs, http://en.wikipedia.org/ wiki/ Spam_in_blogs.
[Wik4] Wikipedia, Mirror Computing, http://en.wikipedia.org/ wiki/ Mirror_%28computing%29.
[Wik5] Wikipedia, Page Hijacking, http://en.wikipedia.org/ wiki/ Page_hijacking.
[Wik6] Wikipedia, Cloaking, http://en.wikipedia.org/ wiki/ Cloaking.
[Wik7] Wikipedia, URL Redirection, http://en.wikipedia.org/ wiki/ URL_redirection.
[Wik8] Wikipedia, Google Bomb, http://en.wikipedia.org/ wiki/ Google_bomb.