

Google search engine

The story behind



Introduction

- Information on web growing rapidly
 - Finding right information becomes complicated
 - Existing search engines have bad search algorithms (easy to become no. 1 in results)
-
-

History of Web Search Engines

- 1994: World Wide Web Worm, had an index of 110.000 web pages
 - Nov 1997: top search engines had from 2 million to 100 million web pages
 - Number of queries has also grown from 1500 queries per day to hundreds of million queries per day
-
-

Scaling with the Web

- Fast crawling technology is needed to gather the web documents
- Documents need to be up to date
- Storage space must be used efficiently
- The indexing system must process data efficiently
- Queries must be handled quickly



Two basic ideas behind

- PageRank it makes use of the link structure of the Web to calculate a quality ranking for each web page
 - utilizing links to improve search results
-
-

PageRank

- The link graph of the web is an important resource
 - We assume page A has pages $T_1 \dots T_n$ which point to it .
 - The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85.
 - $C(A)$ is defined as the number of links going out of page A .
-
-

PageRank

- The PageRank of a page A is given as follows:

$$PR(A) = (1-d) + d (PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$$

- PageRank or $PR(A)$ can be calculated using a simple iterative algorithm



Anchor Text

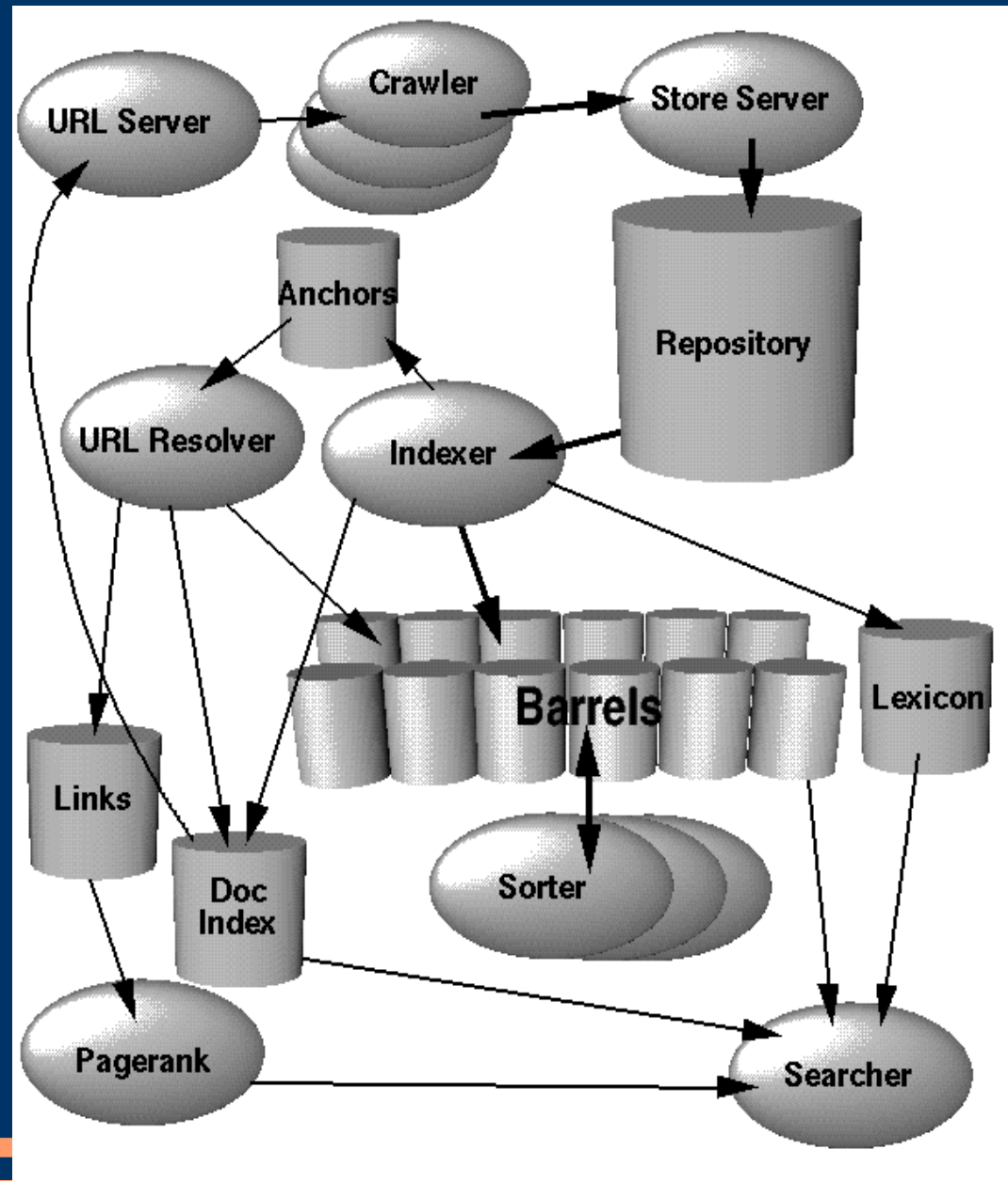
- Most search engines associate the text of a link with the page that the link is on
- Google also associate it with the page the link points to.
- It enables to crawl for non textual web pages also (like images or movies)



Other features

- First, it has location information for all hits and so it makes extensive use of proximity in search.
 - Second, Google keeps track of some visual presentation details such as font size of words. Words in a larger or bolder font are weighted higher than other words.
 - Third, full raw HTML of pages is available in a repository.
-
-

System anatomy



System anatomy

- Most of Google is implemented in C or C++ for efficiency and can run in either Solaris or Linux
 - web crawling done by several distributed crawlers.
 - URLserver that sends lists of URLs
 - Crawlers fetch web pages
 - Storeserver compress and stores web pages into repository.
 - Every web page has an associated ID number called a docID which is assigned whenever a new URL is parsed out of a web page.
-
-

System anatomy

- The indexing function is performed by the indexer and the sorter.
 - The indexer performs a number of functions. It reads the repository, uncompresses the documents, and parses them.
 - The indexer distributes these hits into a set of "barrels", creating a partially sorted forward index.
 - It parses out all the links in every web page and stores important information about them in an anchors file.
-
-

System anatomy

- URLresolver reads the anchors file and converts relative URLs into absolute URLs and in turn into docIDs.
 - The links database is used to compute PageRanks for all the documents.
 - The sorter takes the barrels, which are sorted by docID and resorts them by wordID to generate the inverted index.
-
-

System anatomy

- A program DumpLexicon takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher.
 - The searcher is run by a web server and uses the lexicon built by DumpLexicon together with the inverted index and the PageRanks to answer queries.
-
-

Major data structures

- **BigFiles:** virtual files spanning multiple file systems and are addressable by 64 bit integers
 - **Repository:** contains the full HTML of every web page (compressed with zlib)
 - **Document index:** keeps information about each document. It is a fixed width ISAM index, ordered by docID.
 - **Lexicon:** has several different forms, kept in system memory.
-
-

Major data structures

- Hit list: corresponds to a list of occurrences of a particular word in a particular document including position, font, and capitalization information.
 - Hit lists account for most of the space in the forward and the inverted indices.
 - Because of this, it is important to represent them as efficiently as possible.
 - Forward and inverted index: both are stored in barrels except inverted have been processed by sorter.
-
-

Indexing the Web

- Parsing
- Indexing Documents into Barrels
- Sorting
 - sorts it by wordID to produce an inverted barrel for title and anchor hits and a full text inverted barrel.



Searching

1. Parse the query.
2. Convert words into wordIDs.
3. Seek to the start of the doclist in the short barrel for every word.
4. Scan through the doclists until there is a document that matches all the search terms.
5. Compute the rank of that document for the query.
6. If we are in the short barrels and at the end of any doclist, seek to the start of the doclist in the full barrel for every word and go to step 4.
7. If we are not at the end of any doclist go to step 4.

Sort the documents that have matched by rank and return the top k.

Some interesting data

- For 24 million web pages there is:
 - Total size of fetched pages: 147.8 GB
 - Compressed repository: 53.5 GB
 - Short Inverted Index: 4.1 GB
 - Full Inverted Index: 37.2 GB
 - Lexicon: 293 MB
 - Temporary Anchor Data: 6.6 GB
 - Document Index: 3.9 GB
 - Links Database: 3.9
 - Total Without Repository: 55.2 GB
 - Total With Repository: 108.7 GB
-
-

Some interesting Data

- Number of Web Pages Fetched: 24 million
- Number of Urls Seen: 76.5 million
- Number of Email Addresses: 1.7 million
- Number of 404's: 1.6 million



Thank you for your attention

Presentation files and more resources on:

www.ivan-turkovic.com/projects

