Building Fast Search Engines

Hugh E. Williams (hugh@cs.rmit.edu.au)

School of Computer Science and Information Technology, RMIT

Overview

- User's Information Needs
  - Why users use search engines
  - How users query with search engines
- Answers
  - What is a good answer?
- How search engines provide a search service
  - Indexing data
  - Index design
- Architecture of a commercial search engine
- Research
  - Fast searching and emerging technologies

Queries

- Search engines are one tool used to answer information needs
- Users express their information needs as *queries*
  - Usually informally expressed as two or three words (we call this a *ranked query*)
  - A recent study showed the mean query length was 2.4 words per query with a median of 2
  - Around 48.4% of users submit just one query in a session, 20.8% submit two, and about 31% submit three or more
  - Less than 5% of queries use *Boolean operators* (AND, OR, and NOT), and around 5% contain quoted *phrases*

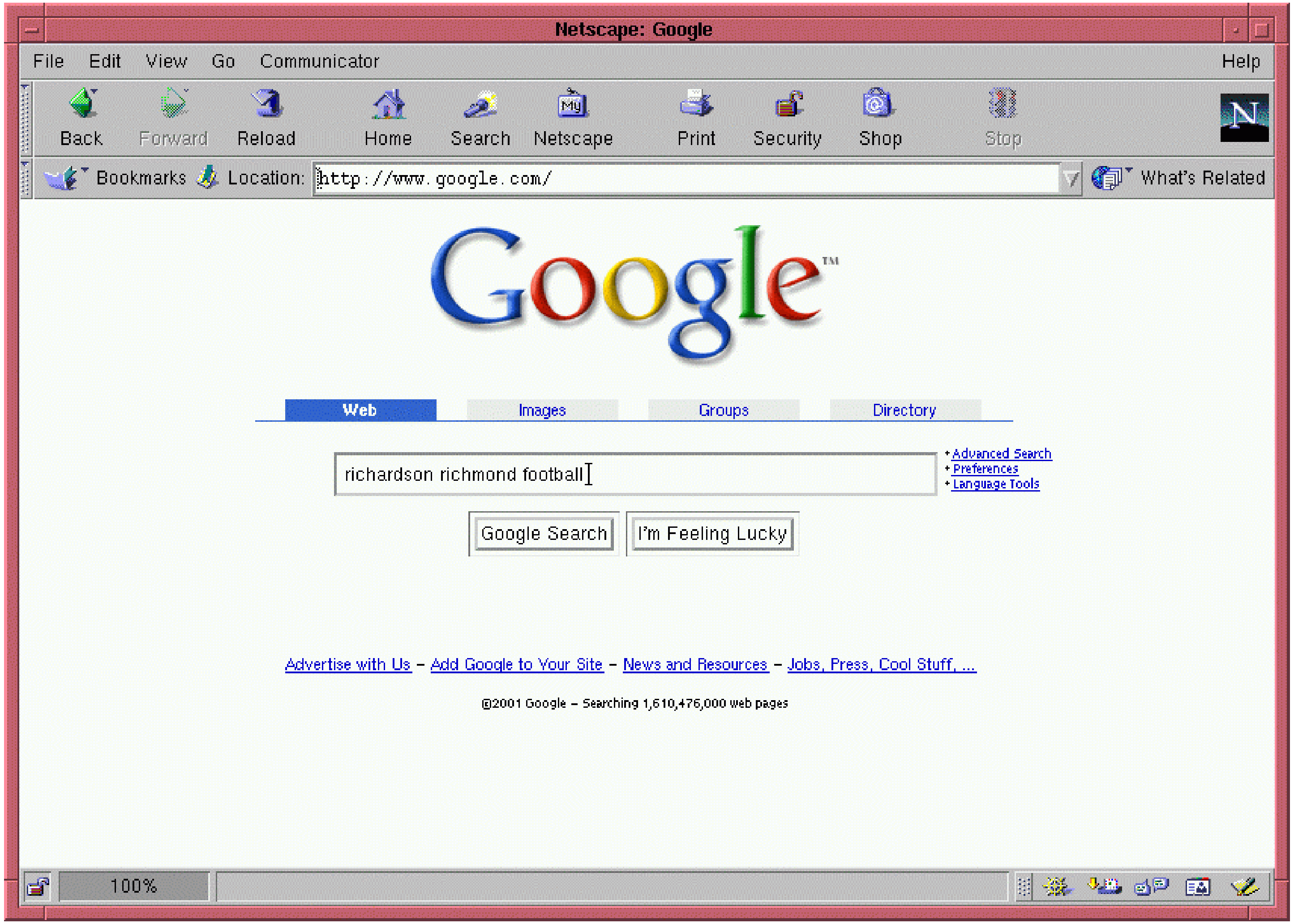
Queries...

- About 1.28 million different words were used in queries in the Excite log studied (which contained 1.03 million queries)
- Around 75 words account for 9% of all words used in queries. The top-ten non-trivial words occurring in 531,000 queries are “sex” (10,757), “free” (9,710), “nude” (7,047), “pictures” (5,939), “university” (4,383), “pics” (3,815), “chat” (3,515), “adult” (3,385), “women” (3,211), and “new” (3,109)
- 16.9% of the queries were about *entertainment*, 16.8% about *sex, pornography, or preferences*, and 13.3% concerned *commerce, travel, employment, and the economy*

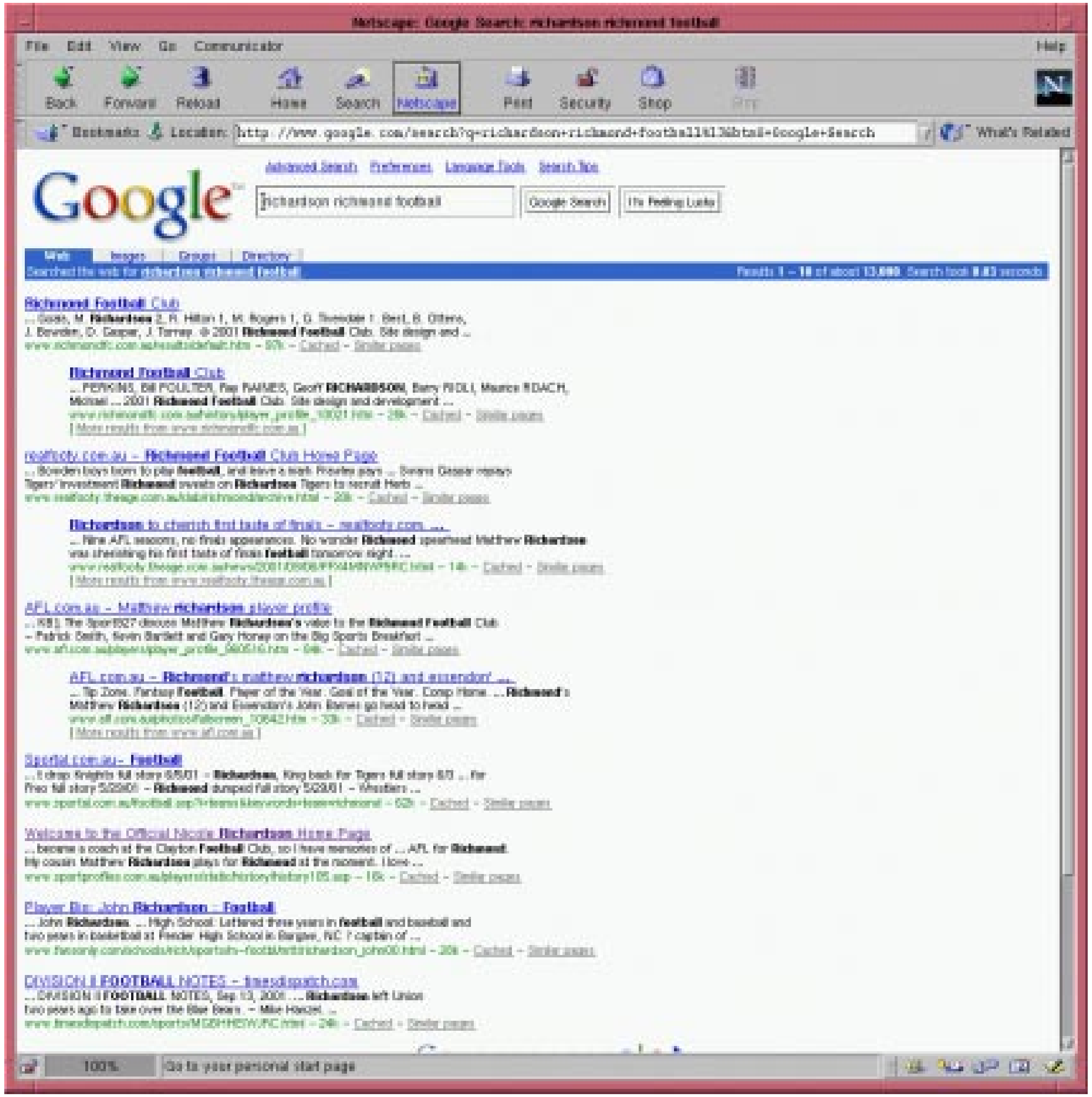
Answers

- What is a good answer to a query?
  - One that is relevant to the user's information need!
  - Search engines typically return ten answers-per-page, where each answer is a short summary of a web document
  - Likely relevance to an information need is *approximated* by *statistical similarity* between web documents and the query
  - Users favour search engines that have *high precision*, that is, those that return relevant answers in the first page of results

An Example Query



Top-ten Answers



Approximating Relevance

- Statistical similarity is used to estimate the relevance of a query to an answer
- Consider the query "Richardson Richmond Football"
  - A good answer contains all three words, and the more frequently the better; we call this *term frequency* (TF)
  - Some query terms are more important—have better discriminating power—than others. For example, an answer containing only "Richardson" is likely to be better than an answer containing only "Football"; we call this *inverse document frequency* (IDF)
- A popular, state-of-the-art *statistical ranking function* that incorporates these ideas is Okapi

Okapi BM25 Function

- The Okapi ranking function is as follows:

$$\sum_{T \in Q} w \frac{(k_1+1)tf}{K+tf} \times \frac{(k_3+1)qtf}{k_3+qtf}$$

  - *Q* is a query that contains the words *T*
  - *k1*, *b*, and *k3* are constant parameters (*k1*=1.2 and *b*=0.75 work well, *k3* is 7 or 1000)
  - *K* is: $k_1((1-b)+b.dl / avdl)$
  - *tf* is the term frequency of the term with a document
  - *qtf* is the term frequency in the query
  - *w* is: $\log \frac{(N-n+0.5)}{(n+0.5)}$
  - *N* is the number of documents, *n* is the number containing the term
  - *dl* and *avdl* are the document length and average document length
- Overall: ranking uses the number of times a word occurs in a document, the number of documents containing the term, and the document length

More on Ranking...

- Other techniques are used to improve the accuracy of search engines:
  - Google Inc. use their patented PageRank(tm) technology. Google ranks a page higher if it links to pages that are an authorative source, and a link from an authorative source to a page ranks that page higher
  - Relevance feedback is a technique that adds words to a query based on a user selecting a *more like this* option
  - Query expansion adds words to a query using thesaural or other techniques
  - Searching within categories or groups to narrow a search

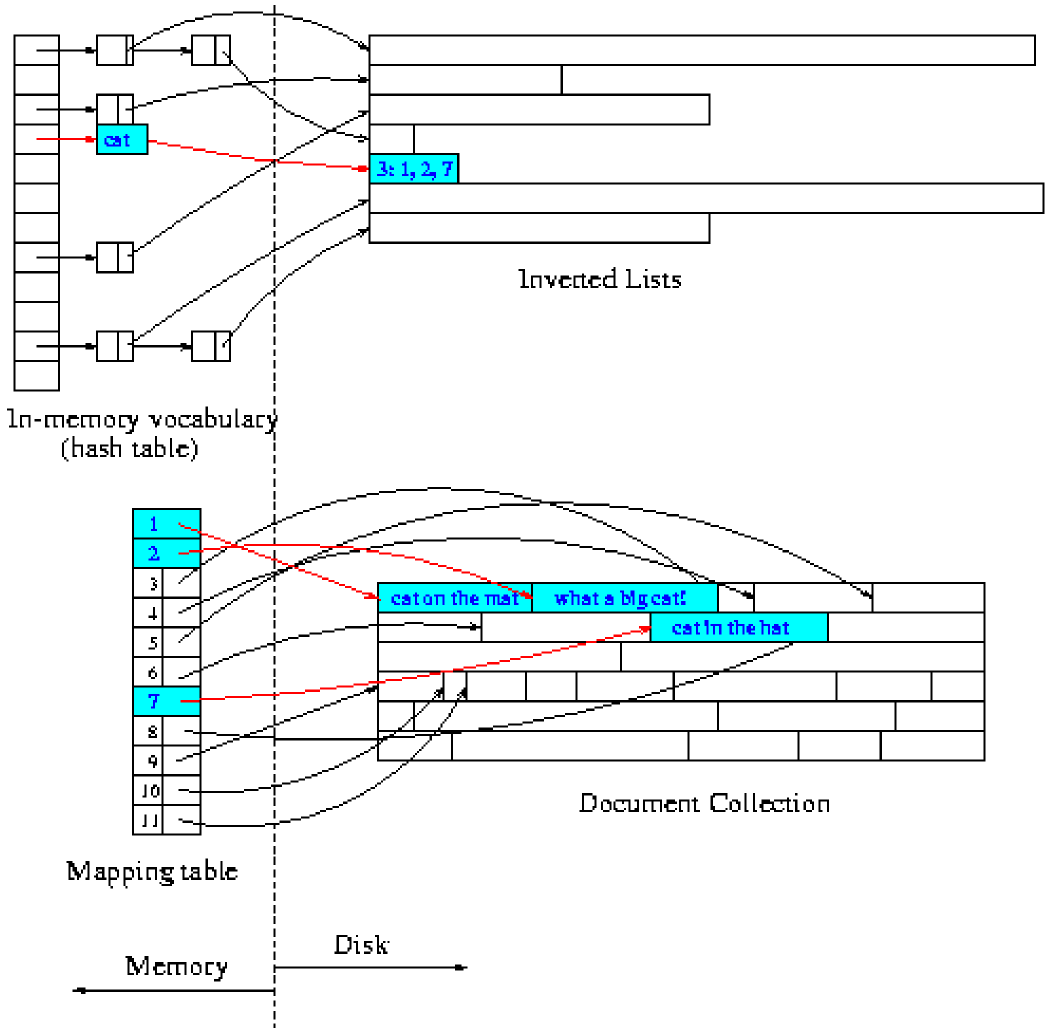
How Search Engines Work

- Search engines work as follows:
  - They retrieve (*spider* or *crawl*) documents from the Web
  - Documents are stored as a *collection* in a centralised repository
  - The collection is *indexed* to allow fast ranking to find answers
  - A web interface is provided for entering queries and presenting answers
  - *Document summarisation* is used to present short answers to the user for judging relevance
  - Documents are updated and re-indexed regularly

Indexing Data

- All search engines use inverted indexes to support fast searching
- An inverted index consists of two components:
  - A searchable *in-memory vocabulary* of all words in the collection; stored with each word is the IDF and a pointer to the inverted list for that word
  - An *on-disk inverted list* for each word in the collection. This list contains:
    - the documents that contain the word
    - the term frequency of the word in each document
    - the *offset* or *offsets* of the word in each document (this is optional, and is used for *proximity* and *phrase queries*)

Indexing Data



Resolving Queries

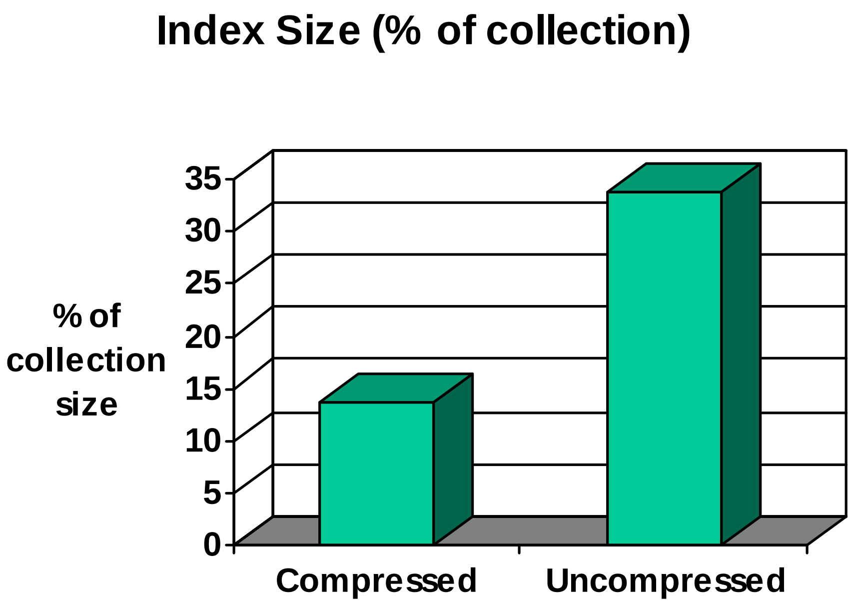
- Queries are resolved using the inverted index
- Consider the example query "Cat Mat Hat". This is evaluated as follows:
  - Select a word from the query (say, "Cat")
  - Retrieve the inverted list from disk for the word
  - Process the list. For each document the word occurs in, add weight to an *accumulator* for that document based on the TF, IDF, and document length
  - Repeat for each word in the query
  - Find the best-ranked documents with the highest weights
  - Lookup the document in the *mapping table*
  - Retrieve and summarise the documents, and present to the user

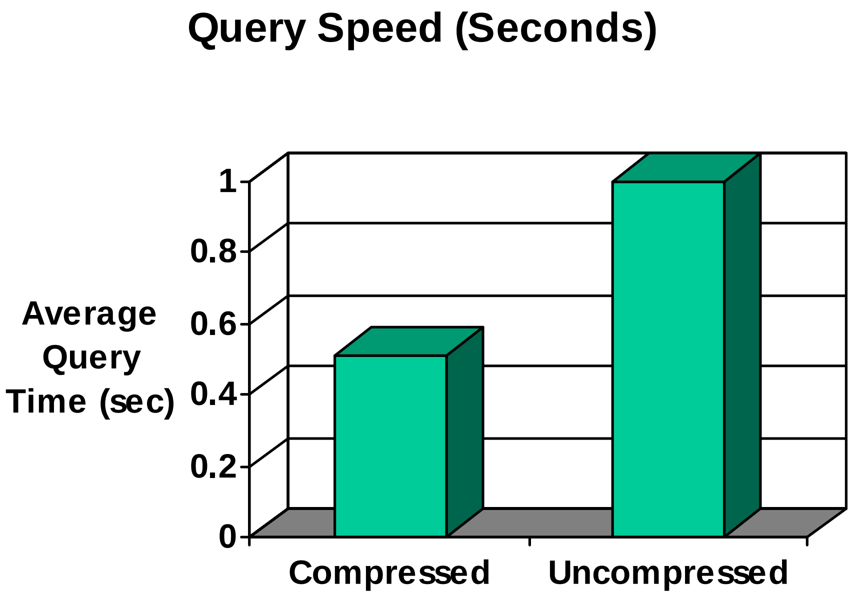
Fast Search Engines

- There are many well-known principles for building a fast search engine
- Perhaps the most important is compression:
  - Inverted lists are stored in a compressed format. This allows more information per second to be retrieved from disk, and it lowers disk head seek times
  - As long as decompression is fast, there is a beneficial trade-off in time
  - Documents are stored in a compressed format for the same reason
  - Different compression schemes are used for lists (which are integers) and documents (which are multimedia, but mostly text)

Fast Search Engines...

- Average query times and index sizes for 25,000 queries on 10 gigabytes of indexed Web data

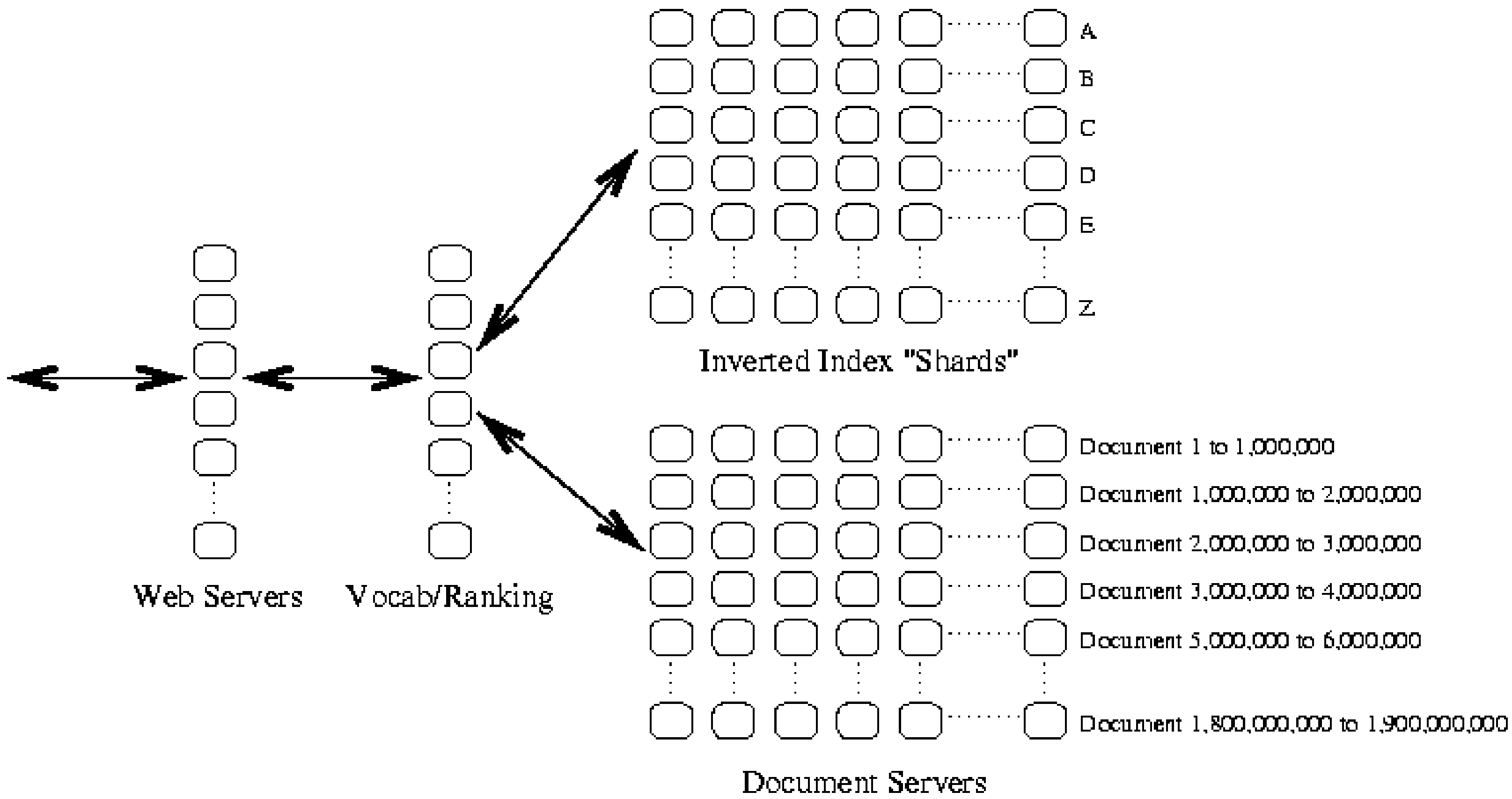




Fast Search Engines...

- Other principles of fast searching:
  - Sort disk accesses to minimise disk head movement when retrieving lists or documents
  - Use hash tables in memory to store the vocabulary; avoid slow hash functions that use modulo
  - Pre-calculate and store constants in ranking formulae
  - Carefully choose integer compression schemes
  - Organise inverted lists so that the information frequently needed is at the start of the list
  - Use heap structures when partial sorting is required
  - Develop a query plan for each query

Search Engine Architecture



Search Engine Architecture...

- The inverted lists are divided amongst a number of servers, where each is known as a *shard*
- If an inverted list is required for a particular range of words, then that *shard server* is contacted
- Each shard server can be replicated as many times as required; each server in a shard is identical
- Documents are also divided amongst a number of servers
- Again, if a document is required within a particular range, then the appropriate *document server* is contacted
- Each document server can also be replicated as many times as required

What we're working on...

- The Search Engine Group here at RMIT specialises in research into fast search engines and applications of search technology to other domains
- We are currently investigating:
    - Fast phrase querying using new index structures
    - Answer summarisation
    - Index design
    - Fast vocabulary searching and accumulation
    - Index construction
    - DNA and protein search engines
    - Image and video management and retrieval
    - General-purpose compression of collections
- Our new research testbed search engine will be released under the GPL later this year

Pointers (& advertising!)

- The Search Engine Group, *http://goanna.cs.rmit.edu.au/~jz/seg/*
- My home page, *http://www.cs.rmit.edu.au/~hugh/*
- Witten, Moffat, and Bell, “Managing Gigabytes”, 2nd edition, Morgan Kaufmann, 1999
- Spink, Wolfram, Jansen and Saracevic, “Searching the web: The public and their queries”, Journal of the American Society for Information Science, 52(3), 226--234, 2001. Queries are available from: *http://www.mds.rmit.edu.au/~hugh/queries/*
- Williams and Zobel, “Compressing Integers for Fast File Access”, The Computer Journal, 42(3), 193-201, 1999.
- Moffat, Zobel, and Sharman, “Text compression for dynamic document databases”, IEEE Transactions on Knowledge and Data Engineering, 9(2):302-313, March-April 1997.
- Zobel and Moffat, “Adding compression to a full text retrieval system”, Software-Practice and Experience, 25(8):891-903, 1995.
- Zobel, Heinz, and Williams, “In-memory Hash Tables for Accumulating Text Vocabularies”, Information Processing Letters. To appear.