Google Scale Data Management

The slides are based on the slides made by Prof. K. Selcuk Candan, which is partially based on slides by Qing Li.

Google (..a course on that??)
Google (..a course on that??)

Google (what?..couldn’t they come up with a better name?)

- ..doesn’t mean anything, right?
- ...or does it mean “search” in French, in German?...
- Surprising, but “Google” actually means something.
  - It comes from googol, which is the name of a “number”…
    - A big one : 1 followed by 100 zeros
- ..this reflects the company’s mission....and the reality of the world:
  - Manage huge amount of data in the world
  - Google’s mission is to organize the world’s data and information and make it universally accessible and useful.
..in fact there is more to Google than search...


Question #1: How can it know so much?...because it crawls...
Question #2: How can it be so fast?

A whole lot of users

One Search Engine

A whole lot of indexed data

Question #1: How can it be so fast?... It indexing and more...

Web

Data

Crawling

Search Engine

Indexing

Index

User

Query

Result

Indexes: quick lookup tables (like the index pages in a book)
Question #2: How can it be so fast?

- use a lot of computers…but, intelligently! (Parallel architecture)
- organize data for fast access (Indexing)
Question #2: How can it be so fast?

- use a lot of computers...but, intelligently! (Parallel architecture)
- organize data for fast access (Indexing)
- most people want to know the same thing anyhow (Caching)

Google Protocol Buffers and BigTable

- What is the data description language used by Google?
  - XML is universal and standard for representing data on Web;
  - For specific situations, XML may be slow;
  - Think about the amount of information stored in Google directories – Any efficiency improvement will be significant;
  - A good design needs compromise / tradeoff between flexibility and efficiency

http://code.google.com/apis/protocolbuffers/docs/overview.html
Google Protocol Buffers

Compared to XML, Protocol buffers:

- are simpler
- are 3 to 10 times smaller in presenting the same information
- are 20 to 100 times faster in processing
- are less ambiguous / flexible
- generate data structures that are similar to object-oriented classes and thus are easier to use programmatically

message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
  enum PhoneType {
    WORK = 0;
    HOME = 1;
    MOBILE = 2;
  }
  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = WORK];
  }
  repeated PhoneNumber phone = 4;
}
Google BigTable

- BigTable is a fast and extremely large-scale database management system;
- It is a compressed, high performance, and proprietary database system built on Google File System (GFS);
- It departs from the convention of a fixed number of columns, instead described by the authors as “a sparse, distributed multi-dimensional sorted map”
Three-level hierarchy structure, analogous to that of a B+-tree

Vector Space Model

<DOC 1>
... cat ....... dog .......
...................dog...........
....mouse ....dog......
mouse .................

Q = < cat, mouse, 0 >

Di = (d_{i1}, d_{i2}, ..., d_{in})
Q = (q_1, q_2, ..., q_n)
Similarity = D_i . Q / |D_i|*|Q|

Di = (1, 2, 3)
Q = (1, 1,0)
Similarity = (1*1+2*1+3*0)/(length of line D1 + length of line Q)

Term weight is only decided by the term frequency
Vectors model of text

- A page with “ASU” with weight <0.5>

Vectors...what are they???

- Page with “ASU” with weight <0.5> and “CSE” <0.7>
Vectors...what are they???

- Page with “ASU” with weight <0.5> and “CSE” <0.7>, and “SCI” <0.9>

Measuring relevance: “Find 2 closest vectors”

Requires “multidimensional” index structures
Question 3: How can it serve the most relevant page?

- Text indexing/keyword search
  - Analyze the content of a page to determine if it is relevant to the query or not

- Link analysis
  - Analyze the (incoming and outgoing) links a pages has to determine if the page is “worthy” or not

Text Indexing
**Text Analysis**

- Query: a set of keywords
- Page/document: is also a set of keywords (where some keywords are more important than the others)

- ...so, to create an index, we need to
  - extract of index terms
  - compute their weights

**Term Extraction**

- **Extraction of index terms**
  - Morphological Analysis (stemming in English)
    - “information”, “informed”, “informs”, “informative”
    - → inform
  - Removal of stop words
    - “a”, “an”, “the”, “is”, “are”, “am”, …
    - they occur too often!!!!
  - Compound word identification
Stop words

- Those terms that occur too frequently in a language are not good discriminators for search.

![Diagram showing frequency distribution of stop words](image)

**An Example**

Identify all unique words in collection of 1,033 Abstracts in biomedicine

- 13,471 terms

Delete 170 common function words included in stop list

- 13,301 terms left

Delete all terms with collection frequency equal to 1 (terms occurring in one doc with frequency 1)

- 7,236 terms left

Remove terminal "s" endings & combine identical word forms

- 6,056 terms left

Delete 30 very high-frequency terms occurring in over 24% of the documents

- 6,026 terms left

Final indexing vocabulary

- 13,471 terms

- 13,301 terms left

- 7,236 terms left

- 6,056 terms left

- 6,026 terms left
How about “important” pages...how to identify them??

- We can learn how important a page is studying its connectivity (“linkages”)

<table>
<thead>
<tr>
<th>(a) Connectivity</th>
<th>(b) Co-citation</th>
<th>(c) Social filtering</th>
<th>(d) Transitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1 → Doc2</td>
<td></td>
<td>Doc1</td>
<td></td>
</tr>
<tr>
<td>Doc1 ← Doc2</td>
<td>X</td>
<td>Doc1</td>
<td></td>
</tr>
<tr>
<td>Doc1 ↔ Doc2</td>
<td></td>
<td>Doc1 ← Doc2</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Doc1 ← Doc2</td>
<td></td>
</tr>
</tbody>
</table>

Hubs and authorities

- Good hubs should point to good authorities
- Good authorities must be pointed by good hubs.
Page Rank PR()

- Basic idea: more links to a page implies a better page
  - But, all links are not created equal
  - Links from a more important page should count more than links from a weaker page

- Basic PageRank PR(A) for page A:
  - \( \text{outDegree}(B) = \text{number of edges leaving page B} \)
  - \( \text{hyperlinks on page B} \)
  - Page B distributes its rank boost over all the pages it points to

\[
PR(A) = PR(C)/1 \\
PR(B) = PR(A)/2 \\
PR(C) = PR(A)/2 + PR(B)/1
\]

How to compute ranks??

- PageRank (PR) definition is recursive
  - Rank of a page depends on and influences other pages

- Solved iteratively
  - To compute PageRank:
    - Choose arbitrary initial \( PR_{\text{old}} \) and use it to compute \( PR_{\text{new}} \)
    - Repeat, setting \( PR_{\text{old}} \) to \( PR_{\text{new}} \) until PR converges (the difference between old and new PR is sufficiently small)
  - Rank values typically converge in 50-100 iterations
  - Eventually, ranks will converge
Open-Source Search Engine Code

- Lucene Search Engine
  - http://lucene.apache.org/
- SWISH
  - http://swish-e.org/
- Glimpse
  - http://webglimpse.net/
- and more

Reference

- Inside the Google Machine
  http://www.youtube.com/watch?v=2FSE3TNFkJQ&NR=1