Session Outline

- Ranking documents in response to a query
- Measuring the quality of such rankings
- Case Study: Tuning 40 parameters at the London School of Economics
- Coffee Break
- Web Search Engineering
- Field Work: how do Web search engines really work?
- Stretch Break
- Discussion: Other IR problems for machine learning
- Historical context
The Web Search Scene (Sep 10)

1. Google
2. Yahoo!
3. BING - Bing Is Not Google

- These have the vast majority of the market in the USA but they’re not alone.
Baidu - leader in China
Naver - leader in Korea
Новости 03:41
1. Райкойон сложил с себя чемпионский плюнокомич
2. Открылась станция метро «Славянский бульвар»
3. Фильм Алексея Гордона-клавдного «Бумажный солдат» получил награды
4. Саркози и Медведев обсудят регулирование ситуации на Кавказе
5. На Кубу обрушился новый ураган «Айс»

Я.Онлайн
А ты?

Например, три закона робототехники

Вход

Каталог сайтов

Австралия. 8 сентября, понедельник

Погода: Канберра +5  днем

Телепрограмма
Google revenue: US$23.65B in 2009!

Let’s get a share of all this money!

Let’s build a Search Engine.
My design parameters for new Web SE

- 500 million queries/day - say peak of 10000 per second.

- Average page revisit frequency - once per month

- 50 billion Web pages
  - Average page size is 25KB (2008)
  - Average URL length is approx 100 characters (5 terabytes!)
- 1.25 petabyte of data (Plus, at least 100% network overhead)
  - 2.5 million gigabytes of network traffic per crawl.
- hundreds of millions of distinct words
- 25 trillion word occurrences
Design issues for the spider

- Data bandwidth = 9.6 gigabits/sec.
  - better not focus the spider on one server (or one country!)
  - need a huge degree of parallelism
  - a major load balancing problem
- Cache needs 5 terabytes to store just URLs!
  - need very efficient lookup methods
Parallel approach to Indexing and Query Processing

- Divide the data across a cluster of PCs
- Node = 2.8 GHz i7 PC, 8 GB RAM, 2 x 1 TB disk,
  - $2k per year to lease and operate
- Replicate clusters and use load-balancing network front-end
Query processing

- Each node is responsible for 50 million documents (1 tB)
  - Therefore need 1000 nodes to handle full collection,
- If each node can process 10 queries/sec
  - Need 10000 queries/sec, therefore 1000 clusters.
  - Need $1000 \times 1000 = 1,000,000$ PCs
  - $2.0B$/year lease cost.

- Say query result page is 25kB long
  - 10000 queries/sec - 2.5 gigabits/sec
(Thanks to Wei-Ying Ma (MSRA) and Mark Sanderson, Sheffield University)
Electricity

- Say each node consumes 250 Watts of electricity.
- Total power draw for 1,001,000 machines $= 250$ megaWatts
- Plus airconditioning at 1:3 $= 83$ megaWatts
- Total power $= 333$ megaWatts
Index structure

Postings (uncompressed).
(2,3)(7,1)(11,2)(17,1)(22,6)

Term Dictionary

Term       count  postings
aaaaa       1
oboe        5
oblong      3
zzzzz       2

Score       DocID    Length  QIE  Snippet
0.145       doc001   2106    0.6
0.212       doc002   5327    0.735 Arist...
0.707       doc003   4108    0.33
0.009       doc004   2999    0.1
0.031       doc005   101     0.2
0.100       doc006   27111   0.7

Acc.s      Document Table
Data structure sizes (per node)

- Term Dictionary – each entry: 12 byte string, 4 byte count, 4 byte offset
  - total size: $10M \times 20 = 200\text{ MB}$
- Doc Table – each entry: 50 byte string (name), 4 byte length, 4 byte score, 100 byte string (snippet)
  - total size: $50M \times 158 = 8.0\text{ gB}$
- Inverted File – each entry: 4-byte docno, 4-byte count
  - total size: $6.5\text{ B} \times 8 = 53\text{ gB}$
- Compressed text for document summaries
  - 100GB?

- It fits easily.
- What about link, anchor text and user behaviour indexes?
- UK-2007 (115M pages) has 4.1 billion links!
Indexing - Use one cluster

- Index 50M pages in about 2.5 days.
- Dirt cheap
Two ways of query processing: TAAT and DAAT

Query: **lpg vehicle subsidy**

1. **[lpg; 27,000]** - (2,7) (2,9) (2,100) (173,5) (2005, 19) (2005,178) (9999, 1) ...  
2. **[vehicle; 112,000]** - (13,31) (18,25) (173,6) (5006,88) (9999, 18) ....
3. **[subsidy; 11,000]** - (99,108) (173,7) (173,99) (7798,13) (9999,205) ...

- **TAAT (Term at a time)**  
  - Process all postings for lpg, then all for vehicle, then subsidy.  
  - Non-zero scores for every doc containing one or more of the terms

- **DAAT (Document at a time)**  
  - Scan three postings lists in ‘parallel’  
  - Only count matches which meet a criterion. e.g. AND or WAND (Broder et al, CIKM, 2003)

(Use skipping)
DAAT Ranking

- Fewer results to sort (full matches only)
  - Can stop when we have required number of matches (or so)
- Assign document numbers in order of descending static score.
  - Compression techniques relying on doc-number diffs still work
  - Ensure that unscanned matches are not likely to be good.
  - Use limited document accumulators (better memory residency)
- Can apply local ranking heuristics to candidate set.
  - Terms in proximity, terms in headings etc.
Cutting costs

▶ Answer Caching
  ▶ Pre-compute answers to popular queries (manually if nec.)
  ▶ Top X queries account for 30% of query load
  ▶ One PC can serve answers for all X queries.
  ▶ Save 300 clusters, $600M per year!

▶ Process Queries on Partial Data, eg. document titles, anchor text.

▶ Shorten results page

▶ Make spidering adaptive, continuous

▶ Avoid crawling spam and low value content

▶ Make query processing more efficient
  ▶ more data per node
  ▶ less time per query
  ▶ (but 10 queries/sec is already good going)
General Efficiency Observations

- Two types of efficiency technique:
  - A. No loss of effectiveness
    - fast algorithms
  - B. Trade-off efficiency/effectiveness
    - use heuristics to reduce work
- Memory management is critical – Can reduce memory demand by
  - careful design/layout of data structures
  - compression
- Disk accesses very slow = 5M instructions
  - Avoid as much as possible
- Disk transfers negligible except for very common terms
  - compression pays off in reduced transfer costs
Hardware Issues

1. Choose cheap, low-heat-output, reliable servers
2. Build in fault tolerance. Can’t allow a whole cluster to be down because one node has failed!
3. Efficient fault diagnosis and repair procedures.
Speeding up query processing

1. Use a limited number of score accumulators
2. Process term lists in parallel, not sequentially
3. Assign document numbers according to descending "value"
4. Stop early
5. Seek perfect load balance across clusters. The cluster is as slow as the slowest node.
Result Presentation

- For each top-ranked doc, print DOCID, snippet etc.
- How to generate summaries
  - Canned summaries - query independent
  - Query biased summaries - quite a computational challenge
Other Issues

- Privacy
  - David Hawking
  - 687 897 255
  - 02 6161 7777
  - 5163 9999 9999 9999
  - 85 Smith St, Nowhere 2586

- Masses of query and click data – what can we use it for?
  - Search quality tuning – machine learning
  - Ranking evidence
  - Spelling corrections.
  - Query suggestions

- Legal issues with advertisers, copyright owners, ...
Doing More with Queries

Maintaining Quality - Are we really Better?
Side-by-side comparisons
The Web search engine business model is very risky

Starting a new Web search engine requires:

- Lots of hardware (intelligently chosen and operated.)
- Huge capital
- Very large bandwidth network connections
- Clever engineering (CE)
  - save huge amounts of network traffic during spidering
  - save vast amounts of query processing hardware

CE during QP relies on

- caching / pre-computation
- clever parallelism
- memory residency / avoidance of disk accesses
- smart algorithms
BTW - What’s the role of a Chief Scientist?