LEXUM

Sub-Second Search on CanLII

Marc-André Morissette

Law via the Internet Conference, June 2011

Overview

- Why are fast search engines important?
- An introduction to search engine theory
- CanLII's search engine
- Algorithmic improvement: bigrams
- Server improvements
- Conclusion: what's next?

Why are Fast Search Engines Important?

- Search is hard
  - Search algorithms are still nowhere near “intelligent”
  - People are bad at formulating queries
- A search experience should be
  - Iterative: queries are refined based on the results of the previous iteration
  - Fast: queries return results under a second
- CanLII was getting slower
  - Content kept growing
    - > 1M documents; 3.5 billion words; 13 million pages of unique text
  - Limited processing power

Search Engine Theory

- Two phases, hence two programs
  - Phase 1: an ***indexer*** reads every document to be searched and creates an ***inverted index***
  - Phase 2: a ***searcher*** performs keyword searches on the inverted index
- Inverted index
  - A sorted dictionary of all words
  - For every word, a list of all documents containing them
  - For every word-document pair, a list of all the occurrences in the document of that word

Inverted Index: an Example

- Two documents
  - Doc 1: The quick brown fox jumps
  - Doc 2: The fox and the dog sleep
- The Inverted Index

Word	Doc. Freq	Occurences
And	1	(Doc=2;Freq=1;Pos=3)
Brown	1	(Doc=1;Freq=1;Pos=3)
Dog	1	(Doc=2;Freq=1;Pos=5)
Fox	2	(Doc=1;Freq=1;Pos=4) (Doc=2;Freq=1;Pos=2)
Jumps	1	(Doc=1;Freq=1;Pos=5)
Quick	1	(Doc=1;Freq=1;Pos=2)
Sleep	1	(Doc=2;Freq=1;Pos=6)
The	2	(Doc=1;Freq=1;Pos=1) (Doc=2;Freq=2;Pos=1,4)

Phrase Query: an Example

- Query: "*the fox*"
- Example
  - Doc 1: The quick brown fox jumps
  - Doc 2: The fox and the dog sleep

Word	Doc. Freq	Occurences
Fox	2	(Doc=1;Freq=1;Pos=4) (Doc=2;Freq=1;Pos=2)
The	2	(Doc=1;Freq=1;Pos=1) (Doc=2;Freq=2;Pos=1,4)

- Performance
  - # query terms * database size * length of documents

CanLII's Search Engine

- Queries without operators: consider every possibility

Word	Prob	VSM score	Weighted score
Workplace privacy dismissal	53%	0.45	0.239
"Workplace privacy" dismissal	35%	0.85	0.298
Workplace "privacy dismissal"	8%	0	0
"Workplace privacy dismissal"	4%	0	0
Total	**100%**	**Sum:**	0.536

- Probability is calculated by Machine Learning Algorithm
  - Based on a training set of properly quoted queries from our log
- Performance
  - Similar to phrase query

CanLII's Old Search Engine: Theoretical Performance

- Performance is linear with amount of content
  - Disk performance: dependant on size of Inverted Index data to be read from disk
  - CPU performance: dependant on the quantity of comparisons to be made

Example: Section 16 of the Criminal Code

Terms	Calculations	Size on disk
Section	431,147	27 MB
16	597,170	12 MB
Criminal	86,560	3 MB
Code	229,905	6 MB
Section-16	4,556,400	
16-Criminal	644,938	
Criminal-Code	610,891	
Total	**7,151,011**	**48 MB**

CanLII's Old Search Engine: Real World Performance

- Example: Section 16 of the Criminal Code

Component	Time required
Basic overhead (constant)	120 ms
Result display (constant)	350 ms
Fetch occurrences from disk (variable)	3000 ms
Compute score (variable)	1830 ms
Total	5300 ms

Algorithmic Improvement: Bigrams

- Most difficult information to compute
  - The position information for words that occur often: large sums of data
    - To fetch
    - To compute
- Solution: pre-compute a lot of the information
  - Bigrams: insert in the inverted index a new dictionary entry for every word pair

Bigrams: an Example of an Inverted Index

- Example: The quick brown fox jumps

Word	Occurrences
Brown	(Doc=1;Freq=1;Pos=3)
Brown-Fox	(Doc=1;Freq=1;Pos=3)
Fox	(Doc=1;Freq=1;Pos=4)
Fox-Jumps	(Doc=1;Freq=1;Pos=4)
Jumps	(Doc=1;Freq=1;Pos=5)
Quick	(Doc=1;Freq=1;Pos=2)
Quick-Brown	(Doc=1;Freq=1;Pos=2)
The	(Doc=1;Freq=1;Pos=1)
The-Quick	(Doc=1;Freq=1;Pos=1)

Searching an Inverted Index with Bigrams: an Example

- Example query: “quick brown fox”

Word	Occurrences
Brown-Fox	(Doc=1;Freq=1;Pos=3)
Quick-Brown	(Doc=1;Freq=1;Pos=2)

- Significantly faster
  - Because bigrams occur significantly less often than their component words

Bigrams: Theoretical Performance Improvement

- Example: Section 16 of the Criminal Code

Word	Without bigrams		With bigrams	
	Calculations	Size on disk	Calculations	Size on disk
Section	431,147	27 MB	431,147	1,7 MB
16	597,170	12 MB	597,170	2,4 MB
Criminal	86,560	3 MB	86,560	0,3 MB
Code	229,905	6 MB	229,905	0,9 MB
Section-16	4,556,400		16,863	0,3 MB
16-Criminal	644,938		375	0 MB
Criminal-Code	610,891		49,737	1,2 MB
Total	**7,151,011**	**48 MB**	**1,413,867**	**6,9 MB**
Improvement			**5 times less**	**7 times less**

Bigrams: Real World Performance Improvement

- Example: Section 16 of the Criminal Code

Component	Without bigrams	With bigrams
Basic overhead	120 ms	140 ms
Result display	350 ms	350 ms
Fetch occurrences from disk	3000 ms	2000 ms
Compute score	1830 ms	400 ms
Total	**5300 ms**	**2900 ms**

Server Improvements

- Disk Performance
  - Degrades linearly wrt number of terms and their frequency
  - 4 terms requires 3 seconds of continuous disk access
  - Previous solution used a disk array of 40 traditional hard disks
- Solution: use specialized solid-state hardware
  - Fusion-IO IoDRIVE Duo SLC flash drive
  - 260,000 disk operations per second (2600 times what a standard disk can provide and 50 times as much as our previous solutions)
  - New server with 2x6 cores of the most powerful Intel Xeon processors

Server: Real World Performance Improvements

- Example: Section 16 of the Criminal Code

Component	Old server	New Server
Basic overhead	140 ms	30 ms
Result display	350 ms	270 ms
Fetch occurrences from disk	2000 ms	Negligible
Compute score	400 ms	250 ms
Total	**2900 ms**	**550 ms**

Conclusion

- Bigrams improve performance of phrase based queries by an order of magnitude
- Search engines are very sensitive to disk performance, especially for large collections
- The next step
  - The Google approach
    - Divide the Internet into small fragments
    - Have a different computer compute relevance for the pages in each fragment
  - Multicore use possible as long as disk systems can follow