Efficiency - Compression

(COSC 488)

Nazli Goharian
nazli@cs.georgetown.edu

1

Efficiency Techniques

- Indexing
- ➢ Compression
- Index Pruning (Top Doc)
- Efficient Query Processing
  - Query thresholding
  - Partial result set processing

2

Facts

- Index (specially position index) may be similar size or generally larger than collection.
- Stop words removal eliminates about half the size of an inverted index. “the” occurs in 7 percent of English text.
- Half of terms occur only once *(hapax legomena)*, so they only have one entry in their posting list
- Some terms have very long posting lists.
- Search engine must mange efficiently the memory hierarchy
- Approaches:
  - Stop word removal, stemming, case folding (*lossy*)
  - *Loss-less* Compression

3

Sample Collection/Index Size before & after Compression

(from: Information Retrieval, Buttcher, Clarke, Cormack)

Collection	Collection Uncompressed	Collection Compressed (gzip)	Index Uncompressed	Index Compressed (vByte)
Shakespeare	7.5 MB	2.0 MB	10.5 MB	2.7 MB
TREC4-5	1904.5 MB	582.9 MB	2331.1 MB	533.0 MB
Gov2	425.8 GB	79.9 GB	328.3 GB	62.1 GB

4

Compression: Goal

- Reducing the storage requirements; compression ratio
- Reducing I/O
- Storing more data in memory cache, and expedite query processing
- Efficiency of decompression algorithm is important!

5

Things to Compress

- Lexicon
  - Not compressed, if fits in memory
  - Compressed, if does not fit in the memory to support query throughput
- Posting List
  - Term Frequencies
  - Document Identifiers
  - Positions

6

Lexicon Compression

- Terms are in lexicographical order, sharing a common prefix. To prevent storing duplicates, use *Front coding (generally saves ~40%), as*

  (preffix lenght, suffix lengh, suffix)

  <"book",(4,3,"ing"), (7,1,"s")><"book",(4,3,"let")…>

- Storing terms in lexicon as a string with pointers indicating end of a term and start of the next term.
- Hashing on terms

7

Delta (gaps) Encoding

- Change the numbers to smaller numbers, thus, fewer bits!
- More common terms -> larger PL-> smaller gaps-> smaller numbers
- Applied to posting lists
  - Term: <d_1,tf_1, {positions}, (d_2,tf_2),{positions}, ... (d_n,tf_n, {positions}>
- Documents are ordered, so each d_i is replaced by the interval difference (*d-gaps*), namely, $d_i - d_{i-1}$
- Smaller *d-gaps* for more common terms
- Index is reduced to ~15% of database size.
- Generally is applied first and then the gaps are further compressed.

8

Compression Techniques of Inverted Indices

- Fixed Length
  - Byte Aligned
- Variable Length
  - Elias Encoding (γ), a family of universal codes
  - Huffman
  - …

9

Byte-Aligned Compression

- Done within byte boundaries to improve Run-time at slight cost to compression ratio.
- Each number is represented by fixed number of bytes, from which 2 bits are length indicators.
- ~15-20% of uncompressed inverted index, when stop words are used.

10

Byte-Aligned Compression

- Algorithm:
  - Take doc id differences (*d-gaps*)
  - Identify number of bytes needed for each *d-gap*.
  - Write length indicator for each *d-gap* in preceding 2 bits.
  - Write the binary representation of *d-gaps*.

Byte-Aligned Compression

0 - 63	00xxxxxx
64 - (16K-1)	01xxxxxx xxxxxxxx
16K - (4M-1)	10xxxxxx xxxxxxxx xxxxxxxx
4M - (1G-1)	11xxxxxx xxxxxxxx xxxxxxxx xxxxxxxx

0	00000000
1	00000001
...	...
63	00111111
64	01000000 01000000
65	01000000 01000001

- The hope here is that the document distance between posting list nodes will be small.

Huffman Coding

182 *Information Retrieval: Implementing and Evaluating Search Engines* · ©MIT Press, 2010 · DRAFT

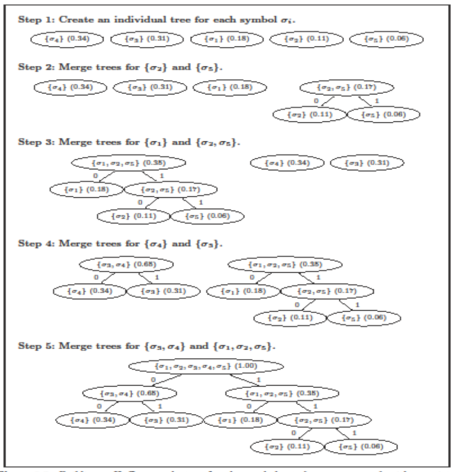

Figure 6.2 Building a Huffman code tree for the symbol set $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$ with associated probability distribution $\Pr[\sigma_1] = 0.18$, $\Pr[\sigma_2] = 0.11$, $\Pr[\sigma_3] = 0.31$, $\Pr[\sigma_4] = 0.34$, $\Pr[\sigma_5] = 0.06$.

Gamma (Elias) Encoding (γ)

X	γ
1	0
2	10 0
3	10 1
4	110 00
5	110 01
6	110 10
7	110 11
8	1110 000
63	111110 11111

To represent value X:

- $\lfloor \log_2 x \rfloor$ ones representing the highest power of 2 not exceeding X.
- a 0 marker.
- $\lfloor \log_2 x \rfloor$ bits representing the remainder $x - 2^{\lfloor \log_2 x \rfloor}$ in binary.
- Uses $2\lfloor \log_2 x \rfloor + 1$ bits to represent value x. The smaller the integer, the fewer the bits used to represent the value. Most *tf's* are relatively small.

Gamma (Elias) Encoding (γ) Example

X= 22

$\lfloor \log_2 22 \rfloor = 4 \qquad 2^4 \le x < 2^5$

4 is highest power of 2 not exceeding 22 => 4 bits unary: 1111

$x - 2^{\lfloor \log_2 22 \rfloor} = 22\text{-}2^4 = 6$

=> 4 bits binary to represent the remaining number 6: 0110

1111	0	0110
4 bits unary for 16	0 marker	4 bits binary for 6

- Decompression is in one pass.

15

Reordering Documents Prior to Indexing

- Reduce doc id gap for better compression
- Similar documents contain similar terms
- Thus, find similar documents and process in that order d_3, d_{50}, d_{200} will be d_1, d_2, d_3
- Methods:
  - Clustering
  - URL info (from same Web server, same directory,…)
  - …..

16

Compression Summary

- Pro
  - Reducing the storage requirements of inverted index
  - Reducing I/O for querying the inverted index
  - Reducing disk seek time
  - Store more data in memory cache, and expedite processing
- Con
  - Takes longer to build the inverted index.
  - Software becomes *much* more complicated.
  - Uncompress required at query time – note that this time is usually offset by dramatic reduction in I/O.

17