

Outline

- Project Questions
 - File Space
 - Other Concerns?
- Outline for Today
 - Index Compression
- Homework
 - Read about AltaVista circa 1997
 - Read S4.1-4.3 of *Modeling the Internet and the Web*

Index Sizes

- Example dataset: newswire
 - Size: 1GB
 - Documents: 400,000
 - Word occurrences: 180,000,000
 - Distinct words (after stemming): 400,000
 - Distinct words per doc: 70,000,000
- Data structures
 - 12MB for 400,000 words, counts, etc.
 - 280MB for 70M document identifiers (4B each)
 - 140MB for 70M document frequencies (2B each)

Index compression techniques

- Compressing the index so that much of it can be held in memory
 - Required for high-performance IR installations (as with Web search engines),
- Redundancy in index storage
 - Storage of document IDs.
- Delta encoding
 - Sort Doc IDs in increasing order
 - Store the first ID in full
 - Subsequently store only difference (gap) from previous ID

Encoding gaps

- Small gap must cost far fewer bits than a document ID.
- Binary (“normal”) encoding
 - Optimal when all symbols are equally likely
 - Not true for gaps
- Unary code (n-1 ones followed by a zero)
 - Favors short gaps too strongly
 - Optimal if probability of large gaps decays exponentially (e.g., $\Pr(N=n) = 2^{-n}$)
 - 1=0, 2=10, 4=1110, 7=1111110, etc.

Encoding gaps

- Elias's Gamma code
 - Represent gap x as
 - Unary code for $1 + \text{floor}(\log x)$ followed by
 - $x - 2^{\text{floor}(\log x)}$ represented in binary (req. $\text{floor}(\log x)$ bits)
 - Total space is roughly $1 + 2\log x$ bits
 - Examples
 - $1=0:$, $2=10:0$, $4=110:00$, $7=110:11$, $9=1110:001$
 - Can you improve on this? (Elias's Delta code)
- Golomb codes
 - Further enhancement; relative to some constant other than 1
- Byte-wise codes also possible
 - Faster, but less efficient in space usage

Improvements Possible

- Term frequencies are typically small (median of 1 or 2) and are thus well-represented by unary or Gamma codes
- Document number gaps are well-represented by Golomb codes
 - newswire uses 7% for Golomb-Gamma coded index
 - or 11% for a bitwise-Gamma coded index
- Processing cost is more than offset by disk seek-time savings

Ordered Postings

- When collection is large, you may not want to read entire posting list
 - Instead, focus on most important postings
 - typically high term frequency (weight) documents
 - sort by frequency rather than by doc id
 - Example
 - Original: $\langle 12,2 \rangle \langle 17,2 \rangle \langle 29,1 \rangle \langle 32,1 \rangle \langle 40,6 \rangle$
 - Frequency: $\langle 40,6 \rangle \langle 12,2 \rangle \langle 17,2 \rangle \langle 29,1 \rangle \langle 32,1 \rangle$
 - Factored: $\langle 6,1:40 \rangle \langle 2,2:12 \ 17 \rangle \langle 1,2:29 \ 32 \rangle$
- If possible, use full impact score
 - Likely need to discretize a real-valued score into 10-30 values
 - As in paper on reverse-engineering AltaVista

Lossy compression

- Trading off space for time
- Collect documents into buckets
 - Construct inverted index from terms to bucket IDs
 - Document IDs can shrink to half their size.
- Cost: time overheads
 - For each query, all documents in that bucket need to be scanned
- Solution: index documents in each bucket separately into per-bucket indices
 - E.g.: Glimpse (<http://webglimpse.org/>)

Useful resources

- Some examples from SIGIR 2003 tutorial on “High-Performance Indexing and Query Evaluation for Information Retrieval” by Justin Zobel and Alistair Moffat
- See their Zettair search engine
 - <http://www.seg.rmit.edu.au/zettair/>
- See *Managing Gigabytes* text (on reserve)