Compression of the dictionary and posting lists
Summary of class discussion 3/12/08

Remarks on Zipf's law:

General law: f_i = frequency of the i^{th} most frequent item = $i^{-\theta} f_1$

for some constant θ. For our application, items are terms that appear in the documents of a collection. The law is observed to hold for other applications with varying values of θ. The text *Introduction to Information Retrieval* focuses on θ = 1.

Comparative growth along regions of the curve: $f_i / f_j = j^{\theta} / i^{\theta}$.
Therefore, if i/j = p/q then $f_i / f_j = f_p / f_q$.

f_i could refer to either the fraction of the total number of occurrences or an actual count of occurrences. If f_i is the actual count of occurrences, *t* is the number of distinct terms and *n* is the total count of occurrences of all items, then

$$f_i = \frac{n}{\sum_{j=1}^{t} j^{-\theta}} i^{-\theta} .$$

($\sum_{j=1}^{t} j^{-\theta}$ is a well-known mathematical quantity: the order θ harmonic number of t.)

Dictionary compression:
The dictionary compression we considered in class is covered in Section 5.2 of *Introduction to Information Retrieval*. Ankur suggested a trie-based data structure for the dictionary. This could work, but each interior node of the trie cannot be of a fixed size in memory because this would require allocating enough space for the maximum possible fanout. The resulting data structure would waste too much space, much as allocating enough space in each array element for the maximum-length term wastes too much space in the straightforward sorted array representation. Both the "dictionary-as-a-string" representation of Section 5.2.1 and a compact trie representation would require much shifting of data on an insertion or deletion. I leave the details of a trie-based data structure as an exercise for those who are interested.

Posting-list compression:
We departed from the treatment in section 5.3 of *Introduction to Information Retrieval* when we discussed bit-level variable-length codes for positive integers.

Let x be a positive integer.
Unary representation of x: 11….10 with x 1's (same as in §5.3).

Elias γ-code for x:
[unary rep. of floor(log x)] ◦ [floor(log x)-bit binary rep. of $(x-2^{\text{floor(log x)}})$]
where ◦ denotes binary string concatenation.

Elias δ-code for x:
[Elias γ-code for floor(log x)] ◦ [floor(log x)-bit binary rep. of $(x-2^{\text{floor(log x)}})$]

The Elias γ-code for x is of length 2*floor(log x) +1, essentially twice the optimal length. The Elias δ-code for x is of length 2*floor(log (floor(log x))) +1 + floor(log x), which has an overhead in additional bits of essentially 2 times the log of the optimal length (i.e. 2loglogx) – a relatively small quantity for large x.

Some compression numbers we did not get time to look at in class:

Reuters-RCV1 collection:
see Table 5.6 in *Introduction to Information Retrieval*.

TREC-3 collection as compressed by Moffat and Bell
(reference: A. Moffat and J. Zobel, Self- indexing inverted files for fast text retrieval,† *ACM Transactions on Information Systems,* Vol. 14, No. 4 (Oct. 1996), pgs 349-379. See also Section 7.4.5 of *Modern Information Retrieval* on reserve in Engineering Library.):

2 GB of document data
1,743,848 documents of size at most 1KB (larger docs chopped into multiple docs)
538,244 terms in dictionary
Inverted index size without compression : 1.1 GB
Entries of the posting list for a term contain only (docID, term frequency in doc) pairs, not a list of occurrences within the document.
Compressed: 184 MB, a 6:1 compression
Gaps between document IDs in the posting lists are compressed used the Golomb code, a code similar in general idea to the Elias γ-code. (For this application, the Golomb code was shown to be slightly better than the Elias δ-code, which is better than the Elias γ-code.) The term frequency values for each document are compressed using the Elias γ-code.

The early (1998) Google index
(reference: S. Brin and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine,† *Proceedings of the Seventh International WWW Conference (WWW 7)*, 1998.) :

14 million terms in the dictionary
24 million documents using 147.8 GB
inverted index using custom, sometimes lossy, compression: 53.5GB

† Links provided on "Schedule and Assignments" Web page.