Compression techniques
Text

Paolo Boldi, DSI, Università degli studi di Milano

Representing and compressing indices

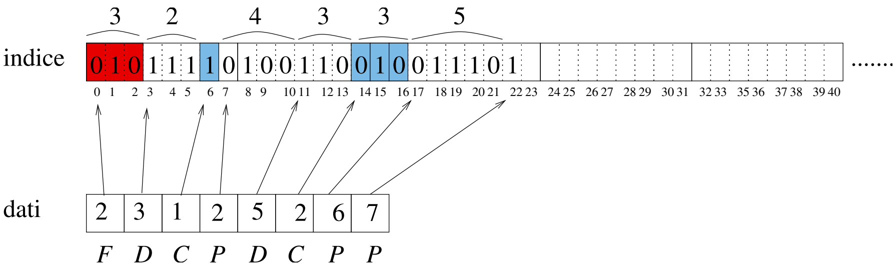
- Recall that an inverted index for a given *corpus* may *contain*:
  - frequence: in how many documents a term appears;
  - pointers: in which documents a term appears;
  - counts: how many times a term appears in each document;
  - positions: where a term appears in each document (*full text*);
  - possibly other global data, like overall number of occurrences, average document length etc.
  - of course, the list of terms!
- probably, it will *not contain*: *stopwords*, *hapax legomena*;
- which codings should we use?

Compression codes

- For frequence and counting, it is only a matter to choose the right coding (usually γ);
- pointers and positions are encoded as gaps;
- depending on the model (or reference distribution) we will have different encodings:
- the most common model is *Bernoulli local*: every term with frequence f appears independently in every document with probability f/N;
- under this assumption, gaps have a geometric distribution (the number of trials before having a success in a Bernoulli distribution), so we should use Golomb.

Compression codes (cont'd)

- alternatively, we can use γ and δ (that are also universal);
- other codes also work very well (e.g., *skewed Golomb*), but have no theoretical explanation;
- in some cases, arithmetic coding is good.



Interpolative encoding

A kind of encoding that works pretty well for positions is interpolative coding.

- To encode the sequence $p_0 < p_1 < \cdots < p_{n-1}$ with lower and upper bound 0 e s, we encode $p_{n/2}$ by its position in the interval $[0, s]$ using a (minimal) binary coding; then we encode recursively $p_0 < \cdots < p_{n/2-1}$ in $[0, p_{n/2} - 1]$ and $p_{n/2+1} < \cdots < p_{n-1}$ in $[p_{n/2} + 1, s]$. An empty list is encoded with ϵ.
- Problem (also in Golomb and arithmetic coding): it is necessary to know the size of the document beforehand.