Managing Gigabytes: Compressing and Indexing Documents and Images

I.H. Witten, A. Moffat, T.C. Bell

Van Nostrand Reinhold, New York, 1994
ISBN 0-442-01863-0

Errata as at May 11, 1999

Front Matter

page vii, last line: "Index 425" → "Index 423"

Chapter One

Chapter Two

page 25, para 2, line 5: "is is" → "it is"

page 34, Table 2.2: The two 18 bit codes are what `mg` actually produces, but strictly speaking they should be 17 bits long, since 100, 101, 102 etc all appear exactly once in the text. `Mg` extends two codes to reserve one codeword for other purposes, but this is not explained at all, and the example should have been hand edited to avoid confusion. If anything, the example is a little bit too realistic in the context of Chapter Two.

page 37, "How arithmetic coding works": we have recently re-examined the implementation of arithmetic coding, and a revised version is described in [?] and available via Alistair Moffat's home page at `http://www.cs.mu.oz.au/~alistair/arith_coder/`

page 42, para 2: The description of PPM should mention the idea of "exclusions". In the example, the context *lie* has occurred 201 times. However, 22 of these occurrences were followed by the letter "s", yet an "s" will never be coded in that context, so it can be *excluded* from the count. This means that only 179 of the occurrences of the context will be used as a sample, and an "r" will have an estimated probability of 19/180. Performing exclusion takes a little extra time, but gives a reasonable payback in terms of compression. There are several other variations of performing sampling that also offer different tradeoffs between speed and compression.

A recent variation of PPM [1] offers improved compression performance.

page 43, para 3, line 4: "is the number them that" → "is the number of them that"

page 46, line 6: "Kirchoff's law" → "Kirchhoff's law"

page 47, line –8: "such a capital letter following a period" → "such as a capital letter following a period".

page 53, para 3, line 1: "Gzip" → "GZip"

page 64, Table 2.5, line "progp": "43,379" → "49,379"

page 68, Table 2.6: "Mbyte/sec" → "Mbyte/min" twice in the body of the table, and in the caption "Mbyte/second" → "Mbyte/minute"

page 70, para 4, line 5: "Santos" → "Santis"

page 70, line 4, line 5: "Lelewer and Hirschberg" are the authors of the *Computing Surveys* paper; the *CACM* paper is by "Hirschberg and Lelewer" and is listed in the bibliography under this correct authorship order.

Chapter Three

page 88, Table 3.6: When $b = 4$, the upper bound on p, 0.1810 → 0.1808. And when $b = 7$, the lower bound on p, 0.0841 → 0.0885.

page 89, para starting "Using this method", line 2: "*hapax legomena*" → "*hapax legomenon*"

page 96, line 5: "a such a" → "such a"

page 98, line 6: "shows that in fact none is an answer to this query" → "shows that only document 2 is an answer to this query"

page 106, para 3, line 9: "the bitstring in Figure 3.7b" → "the bitstring in Figure 3.7c"

page 107, Figure 3.7: The coding shown in part (c) cannot be decoded unambiguously, so the example given is quite incorrect. For example, the sequence "1010 0000 0001 0000" would be represented as "1010, 00, 10, 11", but so would the sequence "1000 0000 0011 0000". One way to avoid this problem is to add an extra bit to the position codes to say whether or not they are the last one for their subsequence; another is to prefix, at each expansion, the number of non-zero subtrees.

page 108, Figure 3.8: The stemmer that was used for this example is proprietary, and so cannot be distributed with `mg`. Hence, if you try this experiment using the distributed `mg` software (which contains a much simpler stemmer) you are likely to get different stemmed forms.

A different stemmer again was added to mg-1.1, which is similar (but still not identical) to the one used for the printed examples.

page 115, para starting "The tradeoff": "MacKenzie" → "McKenzie"

page 115, para starting "Lovin": "Lovin" → "Lovins".

Chapter Four

page 160, line "and the code c": "code c for integer x" → "code c for x"

page 173, line 3: "Baye's Theorem" → "Bayes' Theorem"

page 174, line −5: "Sparc-Jones" → "Sparck-Jones"

Chapter Five

page 179, figure 5.2, line 2(b)iii: "if t" → "If t"

page 182, figure 5.4, line 2(b)iii: "if t" → "If t"

page 182, figure 5.4, line 5(b): "are the inverted file entry" → "are the inverted file entries"

pages 189–195: We have improved the algorithm described in this subsection, and the description here, while correct, is no longer up to date. Contact Alistair Moffat for more details if you are interested, or see [3].

page 195, para 1: "400×50 Kbytes $\times\ 1/2 = 1$ Mbyte" → "400×50 Kbyte $\times$ $1/2 = 10$ Mbytes"

page 201, formula 2, line 2: "$10f$" → "$10ft_r$"

page 204, Table 5.5: On the line "Sort-based, multi-way, in-place", "141" → "150"

Chapter Six

page 245, line 2: "Recall from Section 3.3 that the Golomb code has a" → "In Section 3.3 we described the Golomb code. It is controlled by a". (Section 3.3 is marked by a gray bar, and it seems unfair to ask the reader to recall something they may not have read.)

page 245, para 3, line 5: "For an 8-bit grayscale image, there are only 511 possibilities" → "For an 8-bit grayscale image, there are only 255 possibilities"

Chapter Seven

page 294, line 1: "Mohuiddin" → "Mohiuddin"

Chapter Eight

Chapter Nine

page 339, Figure 9.5: The implementation of canonical decoding has received attention recently, and it is possible to improve upon the mechanism described in this figure. See [**?**] for details.

page 340, lines 9 and 10: "and since the the array *symbol*" → "and since the array *symbol*"

page 340, section "Building a Huffman code": New space-efficient techniques for calculating minimum-redundancy (Huffman) codes have been developed since the book went to press, see [4, 7].

page 341, figure 9.6, item 4(a): "Set $A[1] \leftarrow A[h]$" → "Set $A[1] \leftarrow A[h]$,"

page 349, para 2, line 8: "52" → "53", twice. The Sun uses a floating point format with a 52-bit mantissa, but a normalised value always has a leading "1" bit and this is not stored, hence, to be completely accurate, there are 53 bits of integer accuracy available. The same correction applies also to page 351, para 3, line 5.

page 351, para 1: Length-limited prefix codes were not discussed in detail, as the existing algorithms were impractical for use on large alphabets because of their requirement for large amounts of memory. However, since the book went to press we have devised improved methods for implementing length-limited codes, and it is now possible to calculate them in pretty much the same space and time as ordinary Huffman codes. Hence, much of the material in this section starting on page 345 should now be replaced by a description of how to implement length-limited Huffman codes. For details of the new algorithm, contact Alistair Moffat or see [2].

page 355, para 2, line 6: "*huffword* decodes faster than the *pack*" → "*huffword* decodes faster than *pack*"

page 366, line 3: "on a someone else's" → "on someone else's"

page 367, section 9.5, line 10: "Table 4.8 on page 149" → "Table 4.9 on page 156"

page 372, line 3: "used is as an aid" → "used as an aid"

Chapter Ten

Appendix

page 391, last line: `Mg` is also available as a `gzip` archive. In this case, copy the file `mg.tar.gz` and use the command

```
gunzip < mg.tar.gz | tar xf -
```

pages 393–398: Since completing the book we have developed an X-windows interface. This is not described at all in the appendix. To try this interface, execute `xmg` and then click on the help button. Queries are typed in the top text-box, and executed when enter is typed. Inspect documents by clicking on their first line.

There is a manual page for `xmg` (as there is for all of the `mg` commands), type `man xmg` as a Unix shell command.

page 394, Figure A.1: The output format of `mgstat` has changed since this figure was prepared. Also, note that the different stemmer included with `mg` will give rise to different statistics for the `alice` collection. You are unlikely to be able to reproduce the numbers in Figure A.1(a).

page 406, Table A.4, line 1: A further `.set mode` option has been added:

```
.set heads_length 20
.set mode heads
```

displays just the first 20 characters of each answer to subsequent queries, with whitespace characters compacted. To select and examine one returned document in full, use the document number (33, say) from the desired line and return to `text` mode and change to `docnums` as the query mode:

```
.set query docnums
.set mode text
33
```

This mode is used to good effect in `xmg`.

Bibliography

page 413, line 1: "A. De Santos" → "A. De Santis"

page 414, "Fiala, E.R. and D.H. Green": "Green" → "Greene"

page 417, "Morgenstern, H.B. and S. Morgenstern": "Morgenstern, H.B." → "Barlow, H."

page 418, ref "Robertson": "Sparck Jones" → "Sparck-Jones"

page 421, "Zobel and Moffat (1992)": This paper is now available as [8].

Index

References

[1] J. G. Cleary, W. J. Teahan, and I. H. Witten. Unbounded length contexts for PPM. In J. A. Storer and M. Cohn, editors, *Proc. 1995 IEEE Data Compression Conference*, pages 52–61. IEEE Computer Society Press, Los Alamitos, California, March 1995.

[2] J. Katajainen, A. Moffat, and A. Turpin. A fast and space-economical algorithm for length-limited coding. In J. Staples, P. Eades, N. Katoh, and A. Moffat, editors, *Proc. International Symposium on Algorithms and Computation*, pages 12–21, Cairns, Australia, December 1995. Springer-Verlag. LNCS 1004.

[3] A. Moffat and T. A. H. Bell. In-situ generation of compressed inverted files. *Journal of the American Society for Information Science*, 46(7):537–550, August 1995.

[4] A. Moffat and J. Katajainen. In-place calculation of minimum-redundancy codes. In S. G. Akl, F. Dehne, and J.-R. Sack, editors, *Proc. Workshop on Algorithms and Data Structures*, pages 393–402. Springer-Verlag, LNCS 955, August 1995. Source code available from `http://www.cs.mu.oz.au/~alistair/inplace.c`.

[5] A. Moffat, R. Neal, and I. H. Witten. Arithmetic coding revisited. In J. A. Storer and M. Cohn, editors, *Proc. 1995 IEEE Data Compression Conference*, pages 202–211. IEEE Computer Society Press, Los Alamitos, California, March 1995.

[6] A. Moffat and A. Turpin. On the implementation of minimum-redundancy prefix codes. In J. A. Storer and M. Cohn, editors, *Proc. 1996 IEEE Data Compression Conference*, pages 170–179. IEEE Computer Society Press, Los Alamitos, California, April 1996.

[7] A. Moffat, A. Turpin, and J. Katajainen. Space-efficient construction of optimal prefix codes. In J. A. Storer and M. Cohn, editors, *Proc. 1995 IEEE Data Compression Conference*, pages 192–201. IEEE Computer Society Press, Los Alamitos, California, March 1995.

[8] J. Zobel and A. Moffat. Adding compression to a full-text retrieval system. *Software—Practice and Experience*, 25(8):891–903, August 1995.