A corpus for the evaluation of lossless compression algorithms

Ross Arnold and Tim Bell[*]

Department of Computer Science, University of Canterbury, Christchurch, NZ

1 Introduction

A large number of quite different lossless compression systems have been proposed and implemented over the years. Normally some sort of evaluation is provided by authors to establish the relative merits of their system. This paper investigates how the reliability of these evaluations can be ensured, particularly the repeatability of experiments, in line with scientific method.

The evaluation of compression methods can be analytical or empirical. Analytical results are generally expressed in terms of the compression of a system relative to the entropy of the source, which is assumed to belong to a specified class. Such results tend to have only asymptotic significance; for example, the LZ78 method [ZL78] converges to the entropy for very large inputs, but in practical situations files are far too short for this convergence to have any significance. For this reason empirical results are needed to establish the practical worth of a method.

The main factor measured in typical empirical experiments is the amount of compression achieved on some set of files. Researchers also often report the speed of compression, and the amount of primary memory required to perform the compression. The speed and memory requirements can be different for the encoding and decoding processes, and may depend on the file being compressed. This can result in a daunting number of factors that need to be presented.

A number of authors have used the "Calgary corpus" of texts to provide empirical results for lossless compression algorithms. This corpus was collected in 1987, although it was not published until 1990 [BCW90]. Recent advances with compression algorithms have been achieving relatively small improvements in compression, measured using the Calgary corpus. There is a concern that algorithms are being fine-tuned to this corpus, and that small improvements measured in this way may not apply to other files. Furthermore, the corpus is almost ten years old, and over

[*]`tim@cosc.canterbury.ac.nz`

this period there have been changes in the kinds of files that are compressed, particularly with the development of the Internet, and the rapid growth of high-capacity secondary storage for personal computers.

In this paper we explore the issues raised above, and develop a principled technique for collecting a corpus of test data for compression methods. A corpus, called the *Canterbury corpus*, is developed using this technique, and we report the performance of a collection of compression methods using the new corpus.

2 Evaluating compression performance

It would be tempting to think that the perfect evaluation of a compression method would be to evaluate it on every possible input file and record the average compression over all of those files. However, not only is this computationally infeasible, but the compression results would be disappointing because the average compressed file size must be greater than or equal to the average original size (since in this situation we can only assume that all possible files are equally likely, and thus are best coded using a flat representation). Compression can be thought of as a mapping of all possible input files of a given length to a set of files of different lengths, with shorter representations being given to files that are more likely to occur. As soon as one file is allocated a shorter representation, another file must be given a longer representation, otherwise there will be insufficient distinct codes to allow the files to be decoded unambiguously[1].

Consequently, we can only be concerned about evaluating a method on files that are *likely* to occur. In practice, even this relatively small subset would be impossible to collect, let alone evaluate, and so we must be content to use a small random sample of "likely" files. At this point it is very difficult to argue that one file is more relevant than another, since it is not usually possible to know what the files are that a compression system is likely to be presented with. If we did know then we would design a mapping specifically for them. For example, it is highly unlikely that any system would be presented with more than 2^{100} files to compress in its lifetime, and so all possible files could be mapped on to a 100-bit representation! The problem of predicting what files are "likely" is precisely the problem facing any developer of compression systems, and it is ironic that it also makes the evaluation challenging.

These arguments justify the common approach of choosing a few files that happen to be available as the basis of an empirical evaluation. However, some care needs to be taken over how these files are chosen in order to make the results valid and repeatable.

Validity is questionable if the files are of a specialised type that do not indicate how the compression system would perform in more general circumstances. An extreme case of this is an evaluation that uses files that are particularly well suited to the

[1]Curiously, there has recently been heated debate of what amounts to the same issue in the machine learning community. Wolpert's "no free lunch" theorem [Wol92, Wol94] and Schaffer's "Conservation law of generalization" [Sch94] both stated that for any machine learning algorithm, positive performance in some learning situations must be balanced by negative performance in others. This realization has caused a great deal of controversy in machine learning.

system. An example of this situation would be a compression system with a built-in dictionary that was constructed to suit the test file[2]. Another example is using a test file that contains trivial sequences, such as a test file that consists of "enough copies of the 26-letter alphabet to fill out 100,000 characters" [Abr89] (in this case the results are still valid because other more typical files are also used, but there is a risk that the method described could be reported as achieving compression of "up to 99.42%".)

Determining the type of files that should be used for an evaluation is not simple, and will depend on the application of the system. For example, a compression system to be used in a modem might be evaluated on the kinds of files transferred from a personal computer, while one to be used as part of a transparent disk compression system would be evaluated on blocks of data from a "typical" disk.

As well as the set of test files being valid, it is important that experiments are repeatable. A simple way to achieve this with compression systems is to make both the compression program and test files available to others. However, often researchers are reluctant to distribute their system, and the test files may not be available for public distribution because of copyright problems.

A simple solution to most of the above problems is to develop a corpus of files that is publicly available, and also to publish the results of compression systems on the corpus. The main intent of the repeatability requirement can be achieved if the performance of a system is evaluated independently by a trusted site.

Such a corpus could still be taken advantage of to make a particular compression system look good. One very simple strategy to make an algorithm appear to work well is to detect whether one of the corpus files is being compressed, and to encode it specially. A less extreme example of this is to fine tune the parameters of a system to suit the files in the corpus. This sort of system would easily be discredited by testing it on other files; one could even envisage having a set of secret test files at the trusted site for this purpose. Furthermore, a full replication of the experiment (where the algorithm is re-implemented from its description) should always be possible.

Another deliberately misleading tactic used to "sell" a compression system was used by the OWS and WIC programs, which hid the original data in unused disk blocks or in another part of the file system. Such systems appear to be very impressive until the user attempts to decompress a copy of the compressed file on another computer. Other related strategies would be to store the file somewhere else using the Internet, or to look for another copy of the file elsewhere on the file system. Fortunately this kind of deception can be detected fairly easily by the compression community.

[2]Believe it or not, one of the authors has been presented with a paper to referee that described a dictionary-based compression scheme that was evaluated by deleting from test files all words that were not in the compression dictionary.

3 Survey of test files used in the past

In order to identify trends in the selection of test files, we have tabulated the files used in papers that were presented at the Data Compression Conference in the three years from 1994 to 1996, in Table 1. The table covers both lossy and lossless methods. Of the authors who performed empirical tests, 17.2% did not identify their test data, and a further 30.6% used data that was not used in any other paper. This leaves only about a half of the papers that report results that can conveniently be compared with others.

	1994		1995		1996		Total	%
	papers	posters	papers	posters	papers	posters		
No test data used	12	51	10	40	7	20	140	
Data not identified:								
— lossy	4	6	3	5	1	3	22	11.8
— lossless		2		1	2	5	10	5.4
Data not used by others	1	9	11	6	11	4	57	30.6
Lena	5	4	9	5	8	4	35	18.8
Calgary Corpus	3	1	2	2	3	3	14	7.5
USC	2	1	5	4	2	2	16	8.6
Simulated Source	3	1	1	2	4		11	5.9
CCITT	2		2	1	3		8	4.3
Miss America Video		1		1	1		3	1.6
CT medical	1			1	2		4	2.2
Football Video	1				1		2	1.1
JPEG Standard					2		2	1.1
JBIG Standard			1		1		2	1.1
Total	49	76	44	68	48	41	326	100.0

Table 1: Test data used in Data Compression Conference papers and posters.

The only test data that was used in more than one paper reporting lossless methods was the Calgary Corpus, although almost as often researchers used data that was not even identified. The lossy compression research community faces similar problems, although the "Lena" image is clearly popular. Despite the widespread use of this particular image, using it to compare results is still not straight-forward because the image exists in several versions, in different resolutions and in colour and gray-scale. It also has different names—Lena, Lenna, and WomanHat—and it is not in the public domain because it was taken from a magazine without permission [Opt92].

4 Criteria for choosing a corpus

We have established above that it is very difficult to identify which set of files will be the most suitable for a corpus. However, there are some criteria that are desirable.

- A corpus is should be *representative* of the files that are likely to be used by a compression system in the future. This means that it will contain a variety of different types of files.
- The corpus should be *widely available.* Making it available on Internet is the obvious way to achieve this.
- The corpus should only contain *public domain* material. This rules out a large number of "real" files that we would like to put in the corpus. For example, it would be desirable to have samples from a PC disk to evaluate transparent disk compression methods, but inevitably such a sample would contain proprietary code.
- The corpus should *not be larger than necessary.* If it is too large then it will be too slow or expensive to distribute.
- The corpus should be *perceived to be valid and useful.* This will encourage wide adoption. To create this perception the corpus should include widely used file types, and the procedure used to choose the corpus should be published.
- The corpus should be *actually valid and useful.* The performance of compression algorithms on the corpus files should reflect "typical" performance on other files. The remainder of the work reported in this paper is intended to establish this.

5 Development of a new corpus

The following procedure was used to collect a new corpus of test files, with the aim of selecting the files that give the most accurate indication of the overall performance of compression algorithms.

First, a large number of candidate files (about 800) were identified as being relevant for inclusion in a corpus. The files selected were all public domain, and we tried to obtain as large a variety of types and sizes as possible. These were divided into groups according to their type, such as English text, C source code, UNIX object code, and HTML. Although only a small subset of possible file types was covered, we shall see later that this does not necessarily compromise the validity of results from the corpus.

Figure 1 shows a partial plot of the amount of compression on two of the file groups, the *executables* and the *C source code* groups, for the Unix utility *compress.* This is an example of how different file types consistently exhibit different compression ratios — the file types form distinct linear groups with only the occasional outlier. This behaviour is typical.

For each group we want to select a small number of representative files, with regard to compression. To do this, we compressed each file in the group with a number of compression methods. For each compression algorithm used, a scatter plot of file size before and after compression was obtained. Because files in the group had similar

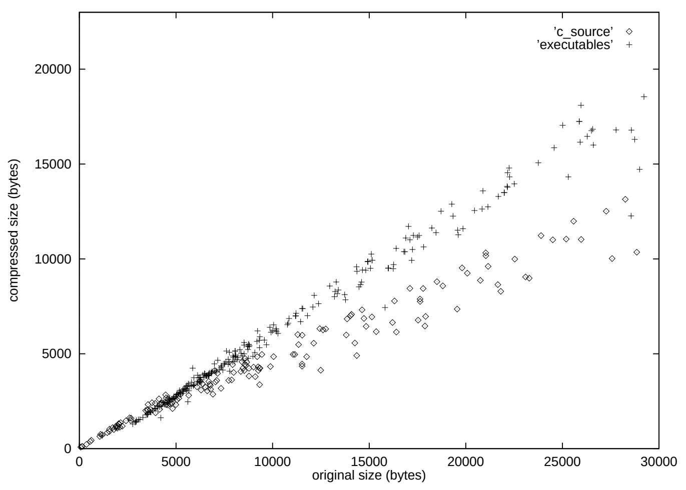

Figure 1: Results from applying the *compress* program to files of two different types. Each data point represents a file.

File Set	Description	N	average correlation	best file	average normalised deviation from line
guten_prose_all	English text	68	0.990	alice29.txt	0.00164
CCITT Test Set	Fax images	8	n/a	ptt5	0.00122
c_source	C source code	164	0.983	fields.c	0.00017
excelfiles	Spreadsheet Files	29	0.995	kennedy.xls	0.00039
gnu_exe	SPARC Executables	188	0.996	sum	0.00106
guten_technical	Technical documents	15	0.992	lcet10.txt	0.00070
guten_poetry	English Poetry	5	0.998	plrabn12.txt	0.00001
html1	HTML	19	0.984	cp.html	0.00045
lisp	lisp source code	76	0.948	grammar.lsp	0.00025
manpages	GNU Manual pages	150	0.998	xargs.1	0.00022
shakesp	Plays	27	0.995	asyoulik.txt	0.00001

Table 2: Correlation in groups of files of the same type

characteristics, the relationship was approximately linear. A straight line was fitted to the points using ordinary regression techniques, so the slope of this line gives the average compression of that file set for that algorithm.

For each group of files we identified a single file that was consistently close to the regression line, where distance from the line was measured as the square of the normalised distance above or below the line.

The results of this experiment are summarised in Table 2. N is the number of files in each group, and the "best file" is the one selected to be in the new Canterbury corpus. Note the high correlation between files in the same group (all of the files in the CCITT test suite were the same size, so no correlation could be obtained).

Unfortunately it is very difficult to find files that are consistently close to the regression line. Despite the relatively high correlation, files can still deviate considerably from the average. It would be desirable to find a representative file whose compression performance would be equivalent to looking up the average compression for the whole group in a table, so that the compression performance on that file could be used as an absolute measure of the compression performance of a method on a

File group	All files (average±s.d.)	Corpus file
lisp source code	2.58±0.74	1.81
C source code	2.38±0.83	1.93
GNU Manual pages	3.37±0.66	2.97
HTML	3.40±0.86	2.32
Average	2.93	2.26

Table 3: Compression performance on all test files compared with corpus files, for *PPM**

large number of files. However, this does not appear to be possible. For example, Table 3 shows the compression performance of a new method (*PPM** [CTW95]) on some of the groups of files. The compression averaged over all files in a group has a relatively high standard deviation, and although the compression on the particular file selected for the corpus is relatively close to the mean, it is too far away to be considered a reliable estimate.

We have experimented with increasing the representative files to five of each type, but this is still highly subject to variation. The average over all file types reduces this effect somewhat, but (as shown in Table 3), the value 2.26 could hardly be considered a good estimate for the correct value of 2.93. Thus we conclude that absolute estimates are unreliable.

Fortunately, it turns out the *relative* compression performance is much more reliable. For example, comparing the *PPM** method to *gzip* for the four file sets in Table 3 indicates that *PPM** files are, on average, 88.3% of the size of *gzip* ones for all test files, while the corpus subset averages 88.1%. Comparing *PPM** with *PPM* gives figures of 97.5% and 97.7% respectively. The standard deviations involved are considerably smaller than the ones for the absolute measurements, and the relative figures on the corpus are very similar to those for the entire set of files available. This indicates that relative measurements on the corpus are reasonably reliable, and a relative improvement of even a few percent is significant.

6 Results

The results of compressing the files in the new Canterbury Corpus are shown in Table 4. Compression is measured in bits per character as this is a convenient unit to work with, and relates closely to entropy measures. It also avoids dependency on the representation of the input file (for example, it is the same whether the input uses an 8- or 7-bit character code.) The average is taken unweighted over all file types (a weighted average, that is, one calculated from the total compressed size compared with the total uncompressed size of the corpus, could give undue weight to a particular class of files). The methods evaluated are *pack* (Huffman coding), *compress* (a public domain utility derived from LZW [Wel84]), *DMC* (Dynamic Markov Com-

File Set	File	pack	compress	DMC	gzip	bred	PPMC
English text	alice29.txt	4.62	3.27	2.38	2.86	2.55	2.30
Fax images	ptt5	1.66	0.97	0.82	0.88	0.82	0.98
C source code	fields.c	5.12	3.56	2.40	2.25	2.17	2.14
Spreadsheet Files	kennedy.xls	3.60	2.41	1.43	1.61	1.21	1.01
SPARC Executables	sum	5.42	4.21	3.03	2.70	2.77	2.71
Technical documents	lcet10.txt	4.70	3.06	2.31	2.72	2.47	2.18
English Poetry	plrabn12.txt	4.58	3.38	2.66	3.24	2.89	2.46
HTML	cp.html	5.30	3.68	2.69	2.60	2.50	2.36
lisp source code	grammar.lsp	4.87	3.90	2.84	2.65	2.69	2.41
GNU Manual pages	xargs.1	5.10	4.43	3.51	3.31	3.26	2.98
Plays	asyoulik.txt	4.85	3.51	2.64	3.13	2.84	2.52
Average		4.53	3.31	2.43	2.54	2.38	2.19

Table 4: Compression (in bits per character) using the Canterbury Corpus on six compression methods.

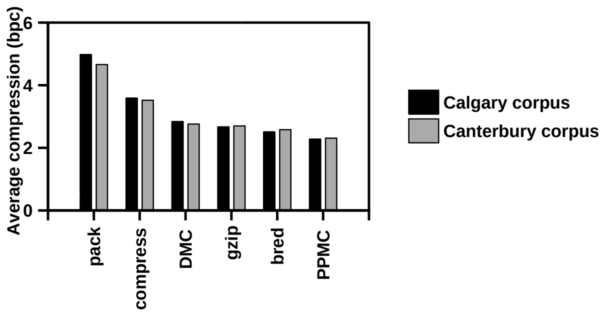

Figure 2: Comparison of compression performance on the Canterbury and Calgary corpora

pression [CH87]), *gzip* (a public domain utility derived from LZ77 [ZL77]), *bred* (a sort based compression system [BW94]), and *PPMC* (Prediction by Partial Matching [Mof90]).

Figure 2 compares the average values obtained for the different methods on the Canterbury and Calgary corpora. The rankings are identical, and the average compression amounts are very similar. This indicates that performance on the Calgary corpus may still be a suitable measure of compression. The main advantage of the Canterbury corpus is that it has files that are more typical of contemporary situations, and that its selection is more principled. The existence of two corpora is useful for checking that any fine tuning of algorithms is not applicable just to one corpus; for example, an algorithm could be have its parameters fine tuned on one corpus, and then the other could be used to check that the parameters still give good performance.

The files in the Canterbury corpus are not particularly large, and sometimes researchers are interested in the performance of compression systems on longer, homogeneous files. For this reason we will add some extra files to the corpus that may

File type	Speed ($\mu s/KB$)	Compression (bpc)
lisp source	1.4	2.66
English Text	2.0	2.48
Binary	4.4	3.59

Table 5: Results of compressing three different file sets with the PPMC algorithm.

be useful for this purpose. However, for the sake of making comparison simple, the subset shown in Table 4 will be used to provide an overall average to give a single figure for a quick comparison.

Measuring the speed of a method is difficult. The immediate problem is that it depends on the speed and architecture of the machine that the program runs on. However, it also depends on the input data. Table 5 shows how the speed can vary using different files for the same program—the speed varies by a factor of more than 3, which is far more than the compression performance varies! However, the difference in speed between different algorithms is also considerably greater than the difference in compression, and so reporting the speed on just one particular file still gives a useful indication of the algorithm's performance.

7 Conclusion

Despite its age and somewhat arbitrary process for construction, the Calgary corpus appears to still be a good indicator of compression performance. However, the Canterbury corpus offers a selection of more contemporary file types that have been chosen using a more justifiable process. Experiments with the new corpus show that while it still does not give a useful *absolute* compression figure, *relative* figures obtained from the corpus are accurate to within a few percent. This means that a measured improvement of as little as 0.1 bits per character is significant for high performance methods.

The web page `http://www.cosc.canterbury.ac.nz/~tim/corpus`[3] provides access to the test files and results of compression for both the Canterbury and Calgary corpora. The associated web pages give details about the results, including the programs and parameters used, and program availability.

References

[Abr89] D.M. Abrahamson. An adaptive dependency source model for data compression. *Communications of the ACM*, 32(1):77–83, January 1989.

[BCW90] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression.* Prentice Hall, Englewood Cliffs, NJ, 1990.

[3]For users in North America, the test files are available more conveniently from `ftp://dna.stanford.edu/pub/canterbury`.

[BW94] M. Burrows and D.J. Wheeler. A block-sorting lossless data compression algorithm. Technical report, Digital Equipment Corporation, Palo Alto, California, 1994.

[CH87] G.V. Cormack and R.N. Horspool. Data compression using dynamic Markov modeling. *Computer Journal*, 30(6):541–550, December 1987.

[CTW95] J. G. Cleary, W. J. Teahan, , and I. H. Witten. Unbounded context lengths for ppm. In J A Storer and J H Reif, editors, *Data Compression Conference (DCC 91)*, pages 52–61, Snowbird, Utah, 1995. IEEE Computer Society Press.

[Mof90] A. Moffat. Implementing the PPM data compression scheme. *IEEE Transactions on Communications*, 38(11):1917–1921, November 1990.

[Opt92] Editorial. *Optical Engineering*, 31(1), 1992.

[Sch94] C. Schaffer. A conservation law for generalization performance. In *Proc. International Conference on Machine Learning*, pages 259–265, 1994.

[Wel84] T.A. Welch. A technique for high performance data compression. *IEEE Computer*, 17:8–20, June 1984.

[Wol92] D. H. Wolpert. On the connection between in-sample testing and generalization error. *Complex Systems*, 6:47–94, 1992.

[Wol94] D. H. Wolpert. Off-training set error and a priori distinctions between learning algorithms. Technical report, Santa Fe Institute, Santa Fe, NM, 1994.

[ZL77] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23:337–343, 1977.

[ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable rate coding. *IEEE Transactions on Information Theory*, IT-24:530–536, 1978.