Entropy Estimates for the Irish Language

Sai Chaithanya Kumar Palada
Department of Mathematics and Statistics, Maynooth University,
Kildare, Ireland.

David Malone
Hamilton Institute / Department of Mathematics and Statistics, Maynooth University,
Kildare, Ireland.

***Abstract*—Entropy estimates for natural languages are useful for a number of reasons. For example, they can be used to estimate the length of a translated text or the amount of text required to make a brute-force attack on an encrypted message feasible. This paper briefly reviews the development of techniques for entropy estimation and then applies modern techniques to Irish text. We believe this addresses a gap in the literature giving an entropy estimate for the Irish language. We discuss our results in the context of entropy estimates for equivalent English text.**

***Index Terms*—entropy, compression, estimation, natural language, Irish, English**

I. Introduction

The information entropy of a source was introduced by Shannon [1] in the 1940s. The entropy of a source of messages describes the average number of bits required to store a message from that source or to transmit such a message. Shannon considered the question of how one might estimate the entropy for English by considering the frequencies and predictability of words and characters [2].

More formally, a source produces message X that is random and $\mathbb{P}[X = m_i] = p_i$. The Shannon Entropy is given by

$$H(X) = -\sum_i p_i \log_2 p_i. \tag{1}$$

For a source with only two messages that occur with probability p and $1-p$ this is sometimes written $h(p) = -p\log_2 p - (1-p)\log_2 1-p$. When considering natural languages, the traditional quantity to consider has been the entropy per character. That is, if we consider a random message X_n of length n from the language, then we would calculate $H(X_n)/n$. In order to average out short term correlations in a language, one would ideally evaluate the limit as $n \to \infty$.

One could estimate the entropy directly using Shannon's formula by estimating the probabilities of various sequences of characters. Under assumptions, such as stationarity and ergodicity of the letters of the language, the entropy per character can be related to the predictability of the next in a sequence of characters of the language. This allows estimates using the predictability of the language [2], [3]. However, as techniques for data compression on computers improved (e.g. [4]), it became possible to exploit the links to data storage requirements and use compression to estimate entropy.

978-1-7281-2800-9/19/$31.00 ©2018 IEEE

While these techniques were originally applied to English, they have also been applied to other languages including Arabic, Chinese, French, Greek, Japanese, Hebrew, Korean, Russian and Spanish [5]–[7]. However, as far as we are aware, these techniques have not yet been applied to the Irish language. In this paper, we will apply data compression techniques to estimate the entropy of the Irish language using three different sources. We will also apply the techniques to the equivalent English documents and compare the results.

In addition to having applications in the storage and transmission of messages, knowing the entropy of a language has a number of other applications. For example, it can be related to the security of a cipher through the *unicity distance* [8] and can be related to other measures of security (e.g. [9], [10]). In a broader context, it is also used as a measure of randomness or unpredictability, which can be optimised when designing data analysis techniques (e.g. [11], [12]).

The paper is laid out as follows. In Section II we describe the method that we follow to carry out entropy estimation for natural language texts. Section III describes our source texts in Irish and English, and also describes synthetic data sources that we use for validation. Section IV begins by describing the results of validating the method, followed by the results of applying the method to our natural language data sets. Section V makes some observations on our results and the paper concludes in Section VI.

II. Method

We follow the method outlined in [7]. We begin by taking a sample of text in a language. We take the first n characters of the sample and compress them to get a file of size $R(n)$ and then calculate the compression rate $r(n) = R(n)/n$. If the compression is efficient for the language in question, then $R(n)$ should be close to the entropy of X_n, and so $r(n)$ will be close to the entropy per character. Consequently, we would like to know the value of $r(n)$ as $n \to \infty$.

A. Compressors

In practice, we have to choose a suitable compressor that will be efficient for Irish. We will consider using the common compressors zip (deflate), gzip, bzip2, xz and 7z in PPMd mode. The PPMd compressor [13] has proven particularly effective with other natural languages [5], [7] and we expect it to be similarly effective when applied to Irish. We also include the Zopfli compressor [14], which compresses data in a format

Compressor	Command Line
zip	`zip -9 -q`
gzip	`gzip --best`
bzip2	`bzip2 --best`
xz	`xz --best`
7z	`7z a -m0=PPMd -mo=9 -mmem=26 -si`
zopfli	`zopfli -c`

TABLE I
OPTIONS USED FOR EACH COMPRESSOR.

Source	Lang	Characters	Bytes	Words	Naïve Entropy
Bible	GA	4866723	5220849	879623	4.445
Bible	EN	4164527	4164527	790020	4.405
Constitution	GA	142291	150890	23866	4.755
Constitution	EN	138135	139331	22679	4.770
GDPR	GA	395278	422247	63262	4.392
GDPR	EN	352067	352324	55124	4.400

TABLE II
A SUMMARY OF THE DATA SETS USED TO ESTIMATE THE ENTROPY. CHARACTER AND WORD COUNTS ARE CONDUCTED WITH THE `wc` TOOL IN A `UTF-8` LOCALE.

that is compatible with traditional formats such as gzip, but achieves more efficient compression at the cost of much higher compression times.

The compressors used are summarised in Table I, including the command line options used to invoke the compressors. The compressors have been used in aggressive modes, intended to achieve efficient compression. Also, where possible, the compressors have been run in a filter mode[1], so that extraneous information about filenames will not be stored with the compressed data.

B. Estimation of Entropy Rate

Another choice we have to make is how to estimate $r(n)$ as $n \to \infty$. One option is to simply take a large amount of text and find $r(n^*)$ for this text, where n^* is the size of the text. We will present results using this method. However, typically the compression rate improves as the compressor sees more data allowing it to learn the statistics of the language. This, combined with overheads associated with writing out file headers and other data structures, mean that the limiting value of $r(n)$ would be typically be expected to approach the entropy from above.

This leads to a second strategy: fitting a curve to $r(n)$ and using the fitted curve to extrapolate to the limit. Several possible parameterised curves[2] have been proposed, but we will fit $f(n) = An^{\beta-1} + h$, corresponding to the f_1 function from [7]. To fit the curve we use the points $n_i = 2^6, 2^7, \ldots$ and including powers of 2 up to and including the file size. The file size is included if it is not a power of two. To find the parameters, we follow [7] and find parameters that minimise

$$\sum_i \left(\ln r(n_i) - \ln f(n_i)\right)^2 .$$

This reduces the impact of larger absolute errors that may be seen for smaller i, as we really want to extrapolate beyond the largest n values. To take the limit of f_1 as $n \to \infty$, we simply use the fitted value of h.

C. Implementation

As the compression phase of this process involves Unix command-line tools, we implemented the compression phase using a shell script to automate the process. The plotting of data and fitting of curves is then implemented with the `gnuplot` package.

[1]A filter mode reads from C's `stdin` and outputs to `stdout`.

[2]These are sometimes referred to as an *ansatz*.

III. DATA SOURCES

A. Irish and English Texts

We work with three sets of data, available in both English and Irish. The first is the Bible. For an Irish version, we use *An Bíobla Naofa*, a 1981 translation of the Bible into Irish [15]. To produce suitable plain text, PDF files of each chapter were converted to plain text using the `pdftotext` command line tool with the `-layout` option. The resulting text was in UTF-8 and was cleaned by removing lines beginning with page feeds, removing digits that appeared to be verse numbers, removing leading/trailing blank space and removing duplicate spaces or blank lines. The resulting UTF-8 contains some accented characters outside the ASCII range and some other characters such as left- and right-quotation marks.

For an English version of the Bible, we took the Project Gutenberg version of the King James Bible [16]. Project Gutenberg provide a UTF-8 version, which is trimmed to begin at the text *The First Book of Moses* and end just before the text *End of the Project Gutenberg EBook of The King James Bible*. Verse numbers are removed and white space is regularised in a similar manner to that applied to the Irish text. This results in a plain ASCII file. Note that the books included in the King James Bible and An Bíobla Naofa are slightly different.

Our second source of text is the Constitution of Ireland. The text is available as a PDF with parallel Irish and English text. The pages were separated into two PDF files for Irish and English pages, including the index pages. The third text is EU Regulation 2016/679, the General Data Protection Regulation (GDPR) [17]. This document is available as a PDF in each official European language. For both sources the pdf was converted to plain text with `pdftotext -layout` and repeated white space, etc., was removed. The result for both was UTF-8 encoded text. The English version of both contains a small number of non-ASCII characters.

Table II shows a brief summary of our sources, giving the number of UTF-8 characters, bytes and words in the file. We also provide a naïve estimate of the entropy, using just the frequencies of the characters and equation 1. This would correspond to ignoring the correlation between characters and assuming that each character is drawn to be independent and identically distributed (IID).

B. Validation Data Sets

In addition to these sources of natural language, we also use two synthetic sources of text for validation. The first is a

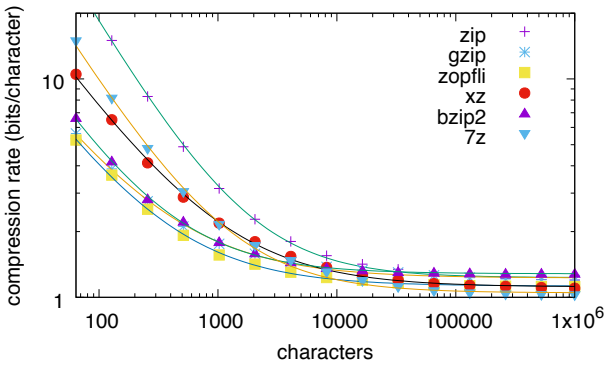


Fig. 1. Performance of various compressors on random Bernoulli 0/1 text.

list of one million ASCII 0 or 1 characters, chosen using a cryptographic random number generator[3]. We regard these as a Bernoulli random source and this source should ideally have an entropy of $h(0.5) = 1$ bit per character.

The second validation source is a list of one million ASCII 0 or 1 characters, where the n^{th} character is 1 if the n^{th} odd number is prime. Though the primes are not randomly distributed their structure is complex and so a generic data compressor is unlikely to identify the pattern. There are 148,932 odd primes less than two million, so a 1 character occurs with probability 0.148932. Thus, for a compressor that noticed no other structure, one would expect an entropy of around $h(0.148932) \approx 0.607$ bits per character.

More generally, there are approximately $n/\ln n$ primes less than n and only one is even [18]. Given that there are approximately $n/2$ odd numbers up to n, the density of odd primes up to n is roughly $2/\ln n - 2/n$. If they appear randomly distributed, the entropy will be approximately $h(2/\ln n - 2/n)$. We will call this the random estimate for the primes data.

IV. Results

A. Validation

To validate our framework, we first apply it to the random Bernoulli 0/1 data. The results are shown in Figure 1, where the values of $r(n)$ are shown for our different compressors. We see that from about 30,000 characters, the 7z PPMd compressor is most effective, as it has the smallest $r(n)$ values. Table III shows the $r(n^*)$ ratio for the full file size, n^*. We see that we get entropy estimates of almost 1.3 bits per character for the bzip2 compressor, around 1.2 bits per character for zip and gzip, around 1.1 bits per character zopfli and xz, and 1.02 bits per character for 7z. Ideally, we expect an entropy of 1 bit per character, so these estimates are relatively good with 7z getting closest to the ideal estimates.

Also shown in the figure are the fitted curves. We can see that they follow the points relatively closely. We are interested in the limiting value of the curves as n becomes large, and we see that the ordering of the curves is the same as the points for

[3]The generator was `/dev/random` on Mac OS X.

Source	Lang	7z	bzip2	xz	zopfli	gzip	zip
Bernoulli	none	1.022	1.282	1.100	1.119	1.199	1.200
primes	none	0.603	0.606	0.619	0.666	0.751	0.752
Bible	GA	1.736	2.068	2.118	2.677	2.813	2.813
Bible	EN	1.528	1.796	1.854	2.316	2.444	2.444
Constitution	GA	1.710	1.924	2.053	2.279	2.380	2.390
Constitution	EN	1.610	1.835	1.929	2.142	2.238	2.248
GDPR	GA	1.257	1.420	1.592	1.920	2.021	2.024
GDPR	EN	1.196	1.350	1.486	1.804	1.901	1.905

TABLE III
Entropy estimates in bits per character using $r(n^*)$ at the file size, n^*.

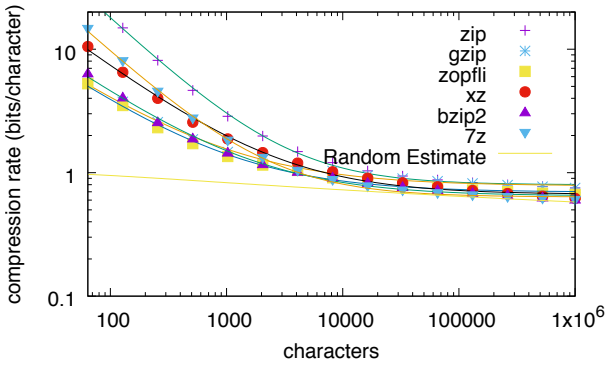


Fig. 2. Performance of various compressors on bitmap of odd primes.

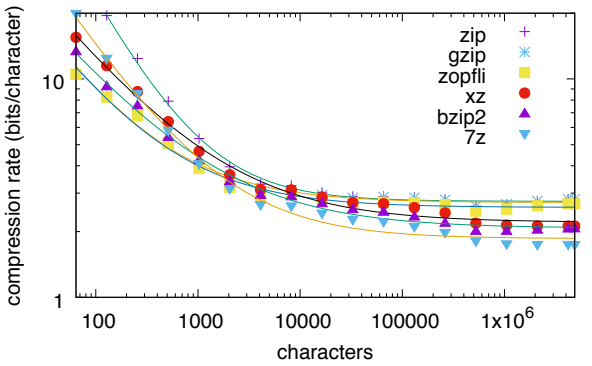
larger n. Table IV shows the results estimating the entropy via the fitted curve, and again they estimate an entropy of a little above 1 bit per character. Comparing the results to Table III we see that the results are broadly similar.

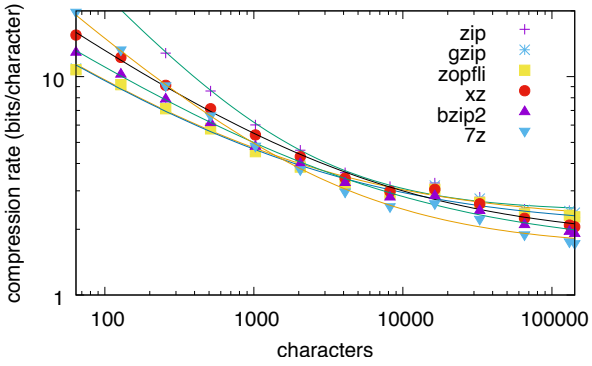
Figure 2 shows the results for our second validation data set, the bitmap of odd primes. By around 10,000 characters 7z is again the most efficient compressor, with the smallest $r(n)$ value. If we inspect Table III, the values of $r(n^*)$ are close to our estimate of 0.607, obtained by assuming that the distribution of the primes looks random to the compressors.

Figure 2 shows that the fitted curves fit well again, and Table IV shows that the fitted values for the entropy are close to 0.607, though a little higher than the values taken from $r(n^*)$. Also shown in the figure is the random estimate from Section III. We see that initially the entropy estimated by the compressors is well above this random estimate, however, by about 10,000 characters there is reasonable agreement.

Source	Lang	7z	bzip2	xz	zopfli	gzip	zip
Bernoulli	none	1.048	1.282	1.116	1.126	1.226	1.230
primes	none	0.635	0.648	0.667	0.695	0.781	0.798
Bible	GA	1.857	2.072	2.203	2.580	2.716	2.752
Bible	EN	1.632	1.820	1.945	2.213	2.338	2.362
Constitution	GA	1.654	1.620	1.786	2.039	2.160	2.414
Constitution	EN	1.527	1.451	1.654	1.892	1.997	2.280
GDPR	GA	1.170	1.040	1.322	1.592	1.709	2.041
GDPR	EN	1.181	1.059	1.386	1.637	1.759	1.981

TABLE IV
Entropy estimates in bits per character via fitting $f_1(n)$ and using h as the entropy estimates for various compressors.





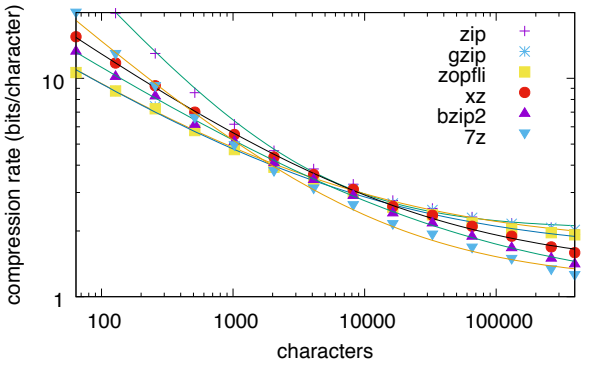
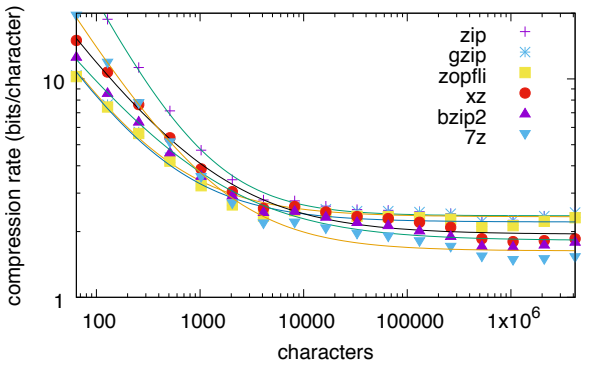
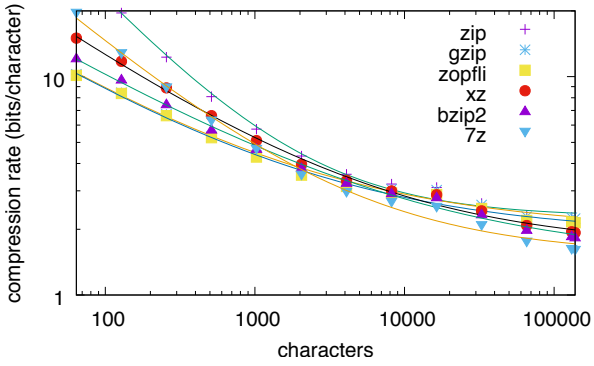


Fig. 3. Performance of various compressors on Irish text. The Bible is shown on the top, the Constitution of Ireland is shown in the middle and the GDPR is shown at the bottom.





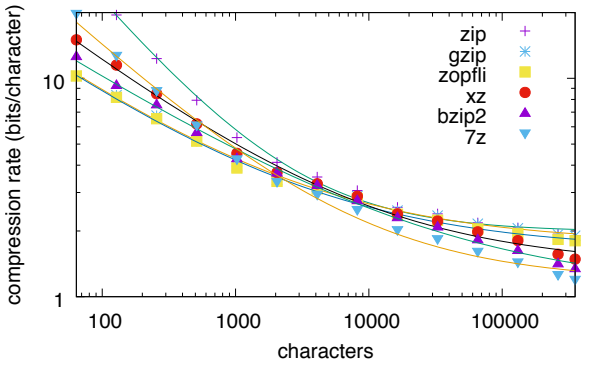


Fig. 4. Performance of various compressors on English text. The Bible is shown on the top, the Constitution of Ireland is shown in the middle and the GDPR is shown at the bottom.

B. Irish and English Texts

Having validated our framework, we now want to move to estimating the entropy per character for Irish. Figure 3 shows the result of applying our framework to the three Irish texts. In previous studies, the 7z PPMd has proven efficient for natural languages, so we expect that it may also be efficient for Irish. We see that for all three texts, by around 4,000 characters 7z is already the most efficient compressor.

Inspecting the entropy estimates given by $r(n^*)$ in Table III, we see that the compressor efficiencies are consistently in the order 7z, bzip2, xz, zopfli, gzip and zip for the Irish text. The entropy of the Bible and Constitution text are about 1.7 bits per character, while the GDPR directive has a lower entropy per character of around 1.25 bits per character.

Interestingly, we observe that the fitted curves for the Irish text in Figure 3 do not follow the $r(n)$ values as closely as we saw for the synthetic text in Figures 1 and 2 For example, for the Bible from about 8,000 characters to 400,000 characters there is a significant gap between the 7z curve and the values of $r(n)$. A similar pattern is present in the results for the Constitution between about 10,000 and 80,000 characters.

The results for estimating the entropy via the fitted curves are shown in Table IV. While they are often within 0.1 bits per character of the values estimated by $r(n^*)$, there are some compressor and text combinations that show larger differences. With these estimates the smallest entropy estimates are given by 7z for the Bible (about 1.85 bits per character), and bzip2 for the Constitution and GDPR (1.6 and 1.0 bits per character, respectively).

For comparison, Figure 4 shows analogous results for the equivalent English texts. We see that many of the features seen

for the Irish texts are repeated for the English texts, including 7z being the most efficient compressor from around 4,000 characters, a consistent order of compressor efficiency and discrepancies between the curve fitting and the $r(n^*)$ values over similar ranges.

Consulting Table III, we see that the English text with 7z gives about 1.5, 1.6 and 1.2 bits per character for the Bible, Constitution and GDPR texts respectively. Interestingly, when the extrapolated values in Table IV are used, the smallest values are given by 7z for the Bible, but bzip2 for the other two texts, as in Irish.

V. Discussion

A first observation is that the entropy of the various texts that we are working with is different. This is unsurprising, as the style in which a text is written will have some impact on the entropy. For example, though written mostly in English, Joyce's Ulysses has been estimated to have a high entropy of around 2.2 bits per character, which is much higher than other English texts, presumably due to its literary style [7].

For our Irish text, it appears that the Bible has higher entropy per character and the GDPR appears to have quite low entropy. As the English version of the GDPR also has low entropy, this is likely to be due to subject matter and style that EU regulations are written in. Translation software is often used to ease translation of EU documents, which could increase the regularity of the text and so decrease entropy.

A. Higher Entropy Irish

Our second observation is that Irish seems to have a higher entropy per character than English, at least when calculated in terms of the compression ratio $r(n^*)$. This is potentially surprising for a number of reasons. First, Irish works with a 23 letter alphabet[4] compared to English's 26 letter alphabet, which would seem to reduce the number of options per character. Second, Irish texts are typically longer than their English translations, suggesting Irish requires more characters to convey similar information.

When we investigated this, we found that though the Irish texts contained more characters, the resulting compressed file was also larger[5]. One possible explanation for this is that in UTF-8 encoding, most characters used in English are represented as a single byte, however the Irish vowels with a fada are represented by multiple bytes, which count as a single input character. However, a compressor will store a small amount of extra information for a multi-byte character.

To determine if this explained the higher entropy for Irish, we recomputed the results using the number of input bytes instead of characters. The results are shown in Table V when the compression ratio is calculated using bytes. When compared to Table III we see the entropy reduced and the gap is closed significantly. The entropy per byte for the Bible and Constitution are still marginally higher for Irish, but the entropy per byte for the GDPR in English is marginally higher. For completeness, we also show the results of fitting curves in Table VI.

Source	Lang	7z	bzip2	xz	zopfli	gzip	zip
Bible	GA	1.619	1.928	1.974	2.495	2.622	2.622
Bible	EN	1.528	1.796	1.854	2.316	2.444	2.444
Constitution	GA	1.612	1.814	1.936	2.149	2.245	2.254
Constitution	EN	1.596	1.819	1.912	2.124	2.218	2.229
GDPR	GA	1.177	1.329	1.490	1.798	1.892	1.895
GDPR	EN	1.195	1.349	1.485	1.802	1.900	1.904

TABLE V
ENTROPY ESTIMATES IN BITS PER BYTE USING $r^b(n^*)$ AT THE FULL FILE SIZE, n^*.

Source	Lang	7z	bzip2	xz	zopfli	gzip	zip
Bible	GA	1.730	1.932	2.054	2.402	2.528	2.563
Bible	EN	1.632	1.820	1.945	2.213	2.338	2.362
Constitution	GA	1.550	1.508	1.666	1.906	2.020	2.271
Constitution	EN	1.511	1.428	1.633	1.869	1.973	2.262
GDPR	GA	1.094	0.976	1.239	1.489	1.598	1.909
GDPR	EN	1.181	1.059	1.385	1.637	1.758	1.980

TABLE VI
ENTROPY ESTIMATES IN BITS PER BYTE VIA FITTING $f_1^b(n)$ AND USING h AS THE ENTROPY ESTIMATES FOR VARIOUS COMPRESSORS.

At an intuitive level, one could make the argument that the information contained in a document in a particular language is the information in the document plus the information about how to express that document in a particular language. Thus, one might expect a certain consistency in the size of compressed documents in different languages [5]. Indeed, Behr et al. noted that the compressed size of the Bible is within about 15% of the size of a compressed English bible [5].

Thus, observing larger compressed files for a language is interesting, For our high-entropy Irish bible, it is approximately 30% larger than the compressed English bible when using 7z. This suggests that while multibyte characters play some role in the higher entropy per character of Irish, this may not be a full explanation.

One possible explanation is that certain features of Irish prove challenging for compressors, for example English has a simpler system of noun declension, and in Irish nouns can often be modified by a séimhiú (e.g. *Constitution* translates as *Bunreacht* but *my Constitution* is *mo Bhunreacht*) or an urú (e.g. *our Constitution* is *ár mBunreacht*) in different circumstances, which may reduce opportunities for matching previous strings.

B. Choice of Compressor

For both English and Irish we consistently saw that the 7z PPMd was the most efficient compressor, while bzip2 came in second. Interestingly, bzip2 found random data a challenge to compress and was outperformed by some of the other compressors. It is possible that one could tune the compressors further to match the language of a particular document. Our

[4]Irish is usually written using the letters *a á b c d e é f g h i í l m n o ó p r s t u ú*, however some loan words use other letters such as *v*. The accent on the vowels is known as a *fada*.

[5]Though a direct comparison between the Bibles is not possible, as they contain a different subset of books.

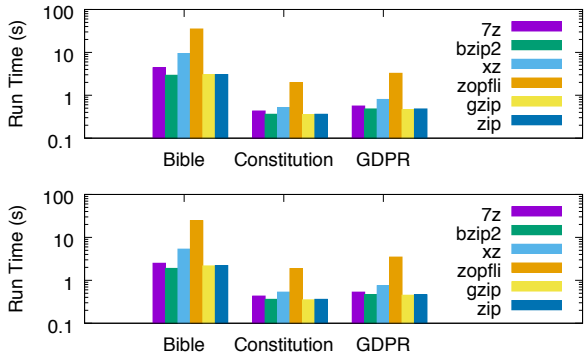


Fig. 5. Runtimes for the compression phase (on 2.3GHz Core i5 on OS X Mojave). Irish texts are shown on the top, English texts below.

choice of parameters for 7z was guided by some further experimentation [19].

While runtime of the compressors is not of direct interest when estimating entropy, it will be of general interest where data is compressed on-the-fly. We show the runtimes of the compression phase in Figure 5 on a log-scale to emphasize the relative runtimes. We observe that the compression times for both Irish and English are similar. As noted in Section II, the run time of the zopfli compressor is expected to be high, and it is substantially higher than all other compressors. Zip, gzip and bzip2 have low runtimes and 7z in PPMd mode appears only slightly more expensive.

C. Curve Fitting and Extrapolation

While 7z was the most efficient compressor and gave the smallest entropy estimates when using $r(n^*)$, it did not always give the smallest estimate when a curve was fitted to $r(n)$. However, for the natural languages, we did observe that the curve fitting was not always satisfactory, and so there is reason to regard these results more critically.

Note that we see features in the raw $r(n)$ values that any curve with a small number of parameters might have trouble following. We investigated the original texts to see if there was an obvious cause (e.g. a change of style) that might explain these features. For example, 80,000 is in the middle of Genesis in the bible, and there is no obvious reason why behavior might change at this point. For the Constitution, 10,000 is approximately the beginning of the list of articles of the constitution after the forward matter. However, the change in style does not seem substantial at this point.

We did also investigate changing the range over which the curve is fitted, and found that this could vary the predicted h values. This indicates that trying to extrapolate as $n \to \infty$ requires considerable care. Indeed, when we extrapolated with our synthetic primes dataset, the fitted curve followed our random estimate well, but then failed to identify that the random estimate goes to zero as $n \to \infty$. In a sense, this is unsurprising, as $1/\ln n$ goes to zero much more slowly than $f_1(n)$ can. However, the extrapolation actually gave larger estimates than our $r(n^*)$ value, contradicting our expectation that the limiting value would be approached from above.

VI. Conclusion

In this paper we have reviewed a method to estimate the entropy per character of natural language. We validated our implementation of the method and then applied it to three texts available in both Irish and English. Our estimates for entropy per character vary from about 1.0–1.8 bits per character. When compared to English, it appears that Irish has a slightly higher entropy per character. Some of this difference appears to be explained by the presence of multi-byte characters in Irish, however the entropy estimates are still relatively high.

In future work, it would be useful to analyse more texts in English and Irish, compare the entropy of Irish with other languages using multi-byte encodings and also to investigate the impact of using other encodings on entropy estimation.

References

[1] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.
[2] ——, "Prediction and entropy of printed english," *Bell System Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951.
[3] T. Cover and R. King, "A convergent gambling estimate of the entropy of english," *IEEE Transactions on Information Theory*, vol. 24, no. 4, pp. 413–421, 1978.
[4] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.
[5] F. Behr Jr, V. Fossum, M. Mitzenmacher, and D. Xiao, "Estimating and comparing entropy across written natural languages using PPM compression," Tech. Rep., 2002. [Online]. Available: \url{http://nrs.harvard.edu/urn-3:HUL.InstRepos:25104999}
[6] L. Levitin and Z. Reingold, "Entropy of natural languages: Theory and experiment," *Chaos, Solitons & Fractals*, vol. 4, no. 5, pp. 709–743, 1994.
[7] R. Takahira, K. Tanaka-Ishii, and Ł. Debowski, "Entropy rate estimates for natural language — a new extrapolation of compressed large-scale corpora," *Entropy*, vol. 18, no. 10, p. 364, 2016.
[8] C. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.
[9] D. Malone and W. Sullivan, "Guesswork is not a substitute for entropy," in *Proceedings of the Irish Information Technology and Telecommunication conference*, 2005.
[10] M. Christiansen and K. Duffy, "Guesswork, large deviations, and Shannon entropy," *IEEE Transactions on Information Theory*, vol. 59, no. 2, pp. 796–802, 2013.
[11] A. Holzinger, C. Stocker, B. Peischl, and K.-M. Simonic, "On using entropy for enhancing handwriting preprocessing," *Entropy*, vol. 14, no. 11, pp. 2324–2350, 2012.
[12] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *IJCAI-99 workshop on machine learning for information filtering*, vol. 1, 1999, pp. 61–67.
[13] A. Moffat, "Implementing the PPM data compression scheme," *IEEE Transactions on Communications*, vol. 38, no. 11, pp. 1917–1921, 1990.
[14] J. Alakuijala and L. Vandevenne, "Data compression using Zopfli," Google, Tech. Rep., 2013.
[15] P. Ó Fiannachta, Ed., *An Bíobla Naofa*. Irish Bible Society.
[16] P. Gutenberg, Ed., *The King James Version of the Bible*. [Online]. Available: http://www.gutenberg.org/ebooks/10
[17] European Union, "Regulation (EU) 2016/679 of the european parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46," *Official Journal of the European Union (OJ)*, vol. 59, no. 1-88, p. 294, 2016.
[18] G. Hardy and E. Wright, *An introduction to the theory of numbers*. Oxford University Press, 1975.
[19] S. C. K. Palada, "Entropy estimation and compression of languages using machine learning," Tech. Rep., 2018.