A Comparison of Huffman codes across languages

1 AUTHOR:

Luciano G Buratto
University of Brasília
18 PUBLICATIONS 55 CITATIONS

SEE PROFILE

A Comparison of Huffman codes across languages

Luciano G. Buratto*

April 19, 2002

Abstract

We present a study correlating encoding and compressibility across four languages: English, France, German and Spanish. We show that French presents the better compression rates among those languages for two sample files, the Biblical texts Psalm 23 and Genesis 1, even though it neither have the best encoding nor the least number of characters. We discuss the possible reasons for this result and the limitations of this approach.

Introduction

Motivation

The field of Information Theory was born motivated by the issue of how to communicate messages between a sender and a receiver. The main idea is to ignore the meaning of the messages and focus on their information content. The message has to be coded as a binary string in order to be transmitted through digital equipment. Sender and receiver are supposed to know how to decode the message. In this context, one of the problems that Information Theory addresses is how to reduce the size of the data being transferred without losing information.

An issue that arises is how to code linguistic information. A related question is how the codes extracted from different languages compare to each other. The goal of this work is to evaluate how successfully different languages with different codes compress the same semantic information.

*lburatto@student.uva.nl, #0127019

We carried out an experiment comparing the codes of four languages, namely English, French, German and Spanish. The codes were extracted from language-specific letter frequency tables (Pratt, 1939). To produce the codes, we implemented Huffman encoding algorithm (Cormen et al., 2001). The codes are based on Shannon-Fano code property, which assigns shorter codes to more frequent letters and longer codes to less frequent ones.

The idea is to compare the text-lengths and corresponding code-lengths for two sample texts, Psalm 23 and Genesis 1, translated into each language. Biblical texts may be used as a benchmark, since they were not written in any of the four languages, avoiding possible translation biases. In particular, these two texts were chosen because they reflect two different language styles: the first is poetic and less structured, whereas the second is more concrete and structured.

French showed better compression rates than English, even though it had longer text-lengths and less optimal encoding. When the encoding was produced from letter frequencies extracted from the sample texts, compression rates and code-lengths worsened across the four languages. Yet the main tendency remained: French showed better compression rates. In the following, we define precisely what we mean by an encoding, which kind of encoding Huffman algorithm outputs and what is it meant for such an encoding to be optimal.

Definitions

Let $E : \Sigma \to \{0,1\}^*$ be any mapping from the alphabet $\Sigma = \{A, B, C, \ldots\}$ to the set of binary strings of possibly variable length. We call the domain of E the set of *source-words* (e.g. 'A') and the range, the

set of *code-words* (e.g. '010'). Thus, the mapping $E(x) = y$, is defined as the *encoding function* that takes a source-word and returns a code-word.

One property we might want in a code is being prefix-free. For $x, y \in \{0,1\}^*$, we call x a *prefix* of y if there is a $z \in \{0,1\}^*$ such that $z = xy$. A set $A \subseteq \{0,1\}^*$ is called *prefix-free* if none of its elements is a prefix of any other element in the set. Thus, a *prefix-free code* can be defined as a function E whose range is prefix-free. In other words, a prefix-free code, or *prefix-code* for short, enables the decodification process to start at the beginning of the binary code-word and decode one code-word at a time. Since no code-word is the prefix of another, the decoder knows where one code-word ends and when the next one starts. Moreover, it facilitates codification, since what one only needs to create a message is to concatenate code-words (e.g. A=1,B=01,AB=101).

Every prefix-code is *uniquely decodable*. This is another property we expect from a good encoding. A code is uniquely decodable if, for each finite sequence of source-words, its sequence of code-words is different from the corresponding sequence of any other source-words. A uniquely decodable code, therefore, avoids ambiguity. Huffman algorithm produces such a prefix-free and uniquely decodable code.

An encoding E is uniquely determined by a probability distribution $P(x) : \Sigma \to [0,1]$. Huffman algorithm takes the letter frequencies as an estimator for the probability distribution. So, each language has a specific encoding determined by their letter frequencies. And, within the same language, the encoding may change if the letter frequencies change (e.g. Some texts in French have words with the letter 'Z' and some do not. These differences result in distinct codes, since the code-length for each code-word is based on the frequency of the source-word).

Methods

We describe here the frequency tables and the encoding algorithm. Next, we define some measures used to compare different codes as to efficiency and distance.

Letter frequency tables

The probability distributions for each encoding are given by a letter frequency table. The tables for each of the four languages were taken from Pratt (1939). They represent the frequency distribution of characters over 1,000 words. We call this frequency distribution the *standard distribution* and the code derived from it, the *standard code* ($Lang_{st}$).

$Char$	$F_{F_{st}}$	$Code_{F_{st}}$	$F_{E_{st}}$	$Code_{E_{st}}$
E	850	00	591	100
A	395	1110	368	1111
S	388	1101	275	0110
I	366	1100	286	0111
T	356	1011	473	001
N	355	1010	320	1100
R	305	1001	308	1011
U	295	0111	111	00010
L	278	0101	153	10101
O	255	0100	360	1110
D	200	11110	171	11011
C	148	10000	124	01010
M	145	01101	114	00011
P	144	01100	89	00000
V	75	100010	41	1101001
Q	61	1111110	5	1101000101
G	51	1111101	90	00001
F	46	1111100	132	01011
B	42	1000111	65	101000
H	35	11111111	237	0100
J	29	11111110	6	1101000110
X	17	10001100	7	1101000111
Y	10	100011010	89	110101
Z	8	1000110111	3	1101000100
K	2	10001101101	19	11010000
W	1	10001101100	68	101001

Table 1: Letter frequencies and encodings for French, F_{st}, and English, E_{st}, using the *standard distribution*. The frequency number is the number of times the letter showed up in a sample of 1,000 words. The encoding is the output of Huffman algorithm.

We also extracted frequency tables from the sample texts. In other words, we implemented an algorithm that reads in Psalm 23 (and Genesis 1)

and returns a frequency table corresponding to each text. We used these tables to create two different encodings for each language: $Lang_{S23}$ and $Lang_{SG1}$.

Huffman algorithm

The algorithm uses a bottom-up greedy strategy. In the beginning, the algorithm assigns to each letter of the alphabet a distinct node bearing the letter's frequency. Then, it successively merges the nodes with lower frequencies and assigns to the new node a frequency given by the sum of the frequencies of the merged nodes.

Let C be the set of frequencies. So $|C| = 26$, the size of the Latin alphabet. We are not considering 'space' as a letter since, in practice, it is possible to determine the meaning of a sentence in natural language even when the words are not separated by spaces. The attributes $left[z]$ and $right[z]$ are respectively the left and right child of the newly created node z. The object z takes as an attribute value the sum of the values $f[x]$ and $f[y]$, the frequency values of the nodes x and y, respectively. The pseudo-code is given by:

HUFFMAN(C)

1. $n \leftarrow |C|$
2. $Q \leftarrow C$
3. **for** $i \leftarrow 1$ **to** $n - 1$
4. **do** $z \leftarrow AllocateNode()$
5. $x \leftarrow left[z] \leftarrow ExtractMin(Q)$
6. $y \leftarrow right[z] \leftarrow ExtractMin(Q)$
7. $f[z] \leftarrow f[x] + f[y]$
8. $Insert(Q, z)$
9. **return** $ExtractMin(Q)$

The resulting tree output by the algorithm is the code. Each code-word is obtained by following the tree from the root to the leaves. Each left child adds '0' to the code-word and each right child adds '1'. The complete code is obtained when all paths from the root to the leaves are taken. The running time of the algorithm is $O(n\log n)$.

Compression rate

The code obtained from Huffman algorithm is a *variable-length code*, since the code-lengths vary from letter to letter. When there is a constant l_0 such that $l(c) = l_0$ for all $c \in C$, it is called a *fixed-length code*.

In order to calculate the compression rate obtained from the Huffman code, we compared the code-length of a given input text in a fixed-length code with the code-length of Huffman's variable-length code. The minimum number of bits necessary to code for the 26 letters of the Latin alphabet is $l_0 = 5$. So the compression rate of encoding E is given by

$$Comp_E = (1 - \frac{Codelength}{l_0 \times Textlength}) \times 100$$

where $Codelength$ is the length of the text in bits when coded using Huffman encoding and $Textlength$ is the number of letters in the text.

Cost and Entropy

Different codes show different average 'lengths'. A measure of such 'length' is called *cost* and defined as

$$C(P) = \sum_{i=1}^{26} P(c)l(c)$$

where $P(c)$ is the probability of encountering the source-word 'c' and $l(c)$ is the length of the code-word corresponding to 'c'. The cost is a weighed average of the size of a given encoding. It represents its *average code-length* in bits.

The question that arises is how good the cost of an encoding is. In order to make comparisons, it is necessary to determine what an optimal encoding is and determine the distance between the optimal encoding and the encoding we are dealing with. According to the *Noiseless Coding Theorem* (Li and Vitányi, 1997), an encoding E is considered optimal if

$$H(P_E) \leq E \leq H(P_E) + 1$$

where $H(P_E)$ is the *entropy* associated with the probability distribution P_E that determines the encoding.

The entropy associated to a probability distribution P is defined in Information Theory as $H(P) =$

$-\sum_{i=0}^{n} P_i \log_2 P_i$, where $P_i = P(x_i)$. The motivation for such definition might be interpreted as follows. Suppose we want to send a message among a set of known messages. Suppose further that the receiver also knows the set of possible messages. So, the only information we need to transmit is that concerning which message we chose. If the set contains two equally likely messages, then we need only 1 bit to inform the receiver which message was chosen. Since the probability of taking either message is $\frac{1}{2}$, the information needed to transmit the message can be calculated by $-\log_2 \frac{1}{2}$. In this case, the information $I(P)$ associated to the given probability distribution is $-\log_2 \frac{1}{2} = 1$ bit.

The same reasoning can be applied to calculate the information contained in a string of characters. If the next character in the string has probability P_0 of being c_0, P_1 of being c_1, and P_i of being the ith possible character, then we have probability P_0 of gaining $-\log_2 P_0$ amount of information, P_1 of gaining $-\log_2 P_1$ bits and P_i of gaining $-\log_2 P_i$ bits.

The entropy, therefore, can be interpreted as an expectation value, which tells us that a symbol will give us *on average* $-\sum_{i=0}^{n} P_i \log_2 P_i$ amount of information (the average number of bits per character).

Thus, from the Noiseless Coding Theorem and the definitions of cost and entropy, it follows that, *an encoding E_1 is more optimal than an encoding E_2* if the respective costs C_1 and C_2 are related as

$$0 \leq H(P_1) - C_1 < H(P_2) - C_2 \leq 1$$

The definition of *less optimal than* is analog.

Distance measure

The distance function $d : E_1, E_2 \rightarrow \mathbb{N}$ between two encodings E_1, E_2 is defined as

$$d(E_1, E_2) = \sum_{i=1}^{26} |l(c_{1_i}) - l(c_{2_i})|$$

where $l(c)$ is the length in bits for each code-word $c_{1_i} \in E_1$ and $c_{2_i} \in E_2$.

The more different is the letter frequency table between two languages, the more different their encodings will be. So the distance $d(Lang_1, Lang_2)$ between the encoding of these two languages should be larger. Conversely, the more similar the frequencies, the 'closer' the encodings and thus the smaller the distance d.

Bible versions

The Bible was originally written in Hebrew, Aramaic and Greek. The translations used in this paper are specified below:

English The New International Version had the 'New Testament' published in 1973 and the 'Old Testament', in 1978. It is published by the International Bible Society (www.gospelcom.net/ibs/niv/).

French The Louis Segond version is the classic French equivalent of the English King James Version. It was published in 1910 by Alliance Biblique Universelle. (www.la-bible.net).

German The Elberfelder Bible was first published in 1871. Since then, there have been several revisions, including the ones in 1975 and 1985. It is published by the R.Brockhaus Verlag (www.brockhaus-verlag.de).

Spanish The Nueva Versión Internacional was produced by the International Bible Society and printed in 1999 (www.IBS-NVI.org).

Results

We split the results into two groups. The first group refers to the encodings corresponding to the *standard distribution*. The second group presents results obtained from text-specific encodings, i.e., codes extracted from the texts.

Standard distribution

The Huffman codes for English and French are presented in Table 1. There are a few remarks to make here. First, the shortest code-word for both languages correspond to the source-word 'E'. The code-lengths, however, are different. The code-word for 'E' has 2 bits in French and 3 bits in English. Second,

the distribution of the frequency mass along the 26 characters appears to be more balanced in English than in French.

The most common source-word in French ('E') is responsible for 17.5% of the frequency mass, as opposed to 13.1% in English. Furthermore, the sum of the frequencies of the 6 least common letters correspond to only 1.1% of the mass for French, whereas for English it rises to 1.8%.

This suggests that the more common source-words in French are more frequent than the more common source-words in English and the less common characters in French are less frequent than the less common characters in English.

Table 2 shows the costs and entropies of the codes obtained for each of the four languages analyzed. French showed the best cost, followed by Spanish, German and English. The lower the cost, the shorter the average code in bits to communicate a message.

Standard	$Cost(C)$	$Entropy(H)$	$(H - C)$
English	4.152941	4.129020	0.023921
French	4.026765	3.988209	0.038557
German	4.132139	4.096078	0.036062
Spanish	4.041112	4.013136	0.027977

Table 2: Cost and entropy measures (in bits) for the *standard encoding* across languages. The larger $(H-C)$ is for a given encoding, the *less optimal* it is. French presents the best cost but the least optimal encoding, whereas English shows the worst cost but the most optimal encoding.

When the cost for each language is compared to its corresponding entropy, however, English takes the lead. It presented the more optimal code followed by Spanish, German and French. The difference between cost and entropy indicates how close an encoding is to the theoretical lower bound given by the entropy. Thus, French encoding had the worst result.

Table 3 shows results on compressibility rates. Although French had longer absolute text-length and code-length when compared to the other languages, it presented the best compression rates (19% for Psalm 23 and 20.1% for Genesis 1). The rates of German were 18.9% and 18.5%, respectively, followed by Spanish (17.9%, 17.2%) and English (14.6%, 16.8%).

Surprisingly, Spanish showed a much shorter text-length and code-length for both texts than the other languages. German, for instance, needs 166 more letters than Spanish to convey the full meaning of Psalm 23 and 468 more characters to do so for Genesis 1.

Figure 1: Frequency tables of French and German using the standard code. The numbers from 0 to 25 correspond to the Latin alphabet ('A'-'Z').

The dissociation between text-length and code-length can be clearly observed when comparing results between English and French for Genesis 1. Even though French needed 105 more letters than English to express the same semantic content, its code-length was 203 bits shorter than the English one. In summary, French could transmit the same semantic information than English using less bits.

The differences between the encodings were quantified with a distance measure. The results agree with the expected relative distance between the languages. English is closer to German ($d(E_{st},G_{st}) = 17$) than to French ($d(E_{st},F_{st}) = 32$) and Spanish ($d(E_{st},S_{st}) = 28$). French, on the other hand, is closer to Spanish ($d(F_{st},S_{st}) = 16$), than to English and German ($d(F_{st},G_{st}) = 37$).

Figure 1 shows the code-length distribution for French and German. The former appears to have a more unbalanced frequency table, with two peaks of 11 bits, whereas the latter shows an overall more balanced set of code-lengths along the alphabet.

The main conclusion from this section is that having a better encoding, as shown for English, does not guarantee a better compressibility rate.

Distributions from sample texts

The results presented so far may be biased, since they are all based on the so-called *standard distribution*. This distribution has no guarantee of representing the true letter distribution of the languages analyzed. In some contexts, a set of words is more frequent than in other contexts. Therefore, the relative letter frequencies might change according to the contexts defined by different texts.

To further test the consistency of the results thus far, we extracted frequency tables and Huffman codes from the sample texts Psalm 23 and Genesis 1. We then took the codes, $Lang_{S23}$ and $Lang_{G1}$ respectively, and calculated the compressibility rates across languages. In short, the encoding for French F_{S23} was used to measure the compressibility rate for Psalm 23 and the code F_{G1} was used for Genesis 1.

The results again show that French has the best compressibility rate and English has the worst for both texts (18.6% and 21.2% vs. 16.8% and 20.2%). Note that, overall, text-specific codes showed more efficient compressibility than the standard code. This is so since the individual code-lengths reflect the letter frequency distributions given by the texts.

Codes derived from texts can be very different. Table 4 shows the distances between text-specific codes and the distances between either specific encoding and the standard code. The distances between text-specific codes in the same language can sometimes be even higher than the distances between standard encodings of different languages.

In order to assess how these differences affect compressibility, we crossed each text-specific code with the alternative text, i.e., we took Psalm 23-derived code and calculated the compression achieved in Genesis 1 using that code (and vice-versa).

Once more, French showed the best compression (18.5% for Psalm 23 using code derived from Genesis 1 and 19.1% for the reverse). Accordingly, English presented the worst rate (12.9% and 14.6%). The overall rates decreased, as expected. This is so because the frequencies extracted from Psalm 23 do not match the ones from Genesis 1 (the distance between them is high for each of the tested languages).

One example of the existing differences between text-specific codes comes from the letters 'Y' and 'Z', whose frequency was 0 in the French version of Psalm 23 and 21 and 8 respectively in Genesis 1. This different is reflected in the code-lengths: 'Y' and 'Z' require 11 bits in the text-specific code from Psalm 23, whereas only 7 and 8 bits respectively are necessary for Genesis 1 code.

To sum up, French showed the best compression rates among the group of languages analyzed. In addition, the rates are conserved across different encodings derived from texts. Interestingly, the results are conserved even when these text-specific encodings are very different from each other.

Discussion

Why is French more compressible?

The findings described in the previous sections appear to be counterintuitive. English has a better encoding than French and the remaining languages. Moreover, the average length of English words, extracted from the letter frequency table, is 4.5 letters per word, as opposed to 4.86 in French and 5.92 in German. On top of that, English texts for Psalm 23 and Genesis 1 are shorter than the French and German equivalents. Nevertheless, the compressibility achieved by the English binary code was overcame by French, German and Spanish codes.

This interpretation, however, may be misleading. First, having a better encoding does not entail better compressibility rates. The concept of *optimal encoding* is built on an average result: the entropy of the code. So, it would be expected that English outperforms the other languages when the sample space of texts increases considerably. That is not the case in this study, where only two texts were used.

Second, average shorter word-lengths do not guarantee shorter codes. It might be the case that the texts we analyzed contain many unusual words, which would not match the probabilities laid out in

$Comp_{E_{st}}$	$Psalm23$			$Genesis1$		
$Language$	$Textlength$	$Codelength$	$Comp$	$Textlength$	$Codelength$	$Comp$
English	441	1883	14.6%	3029	12593	16.8%
French	467	1891	19.0%	3134	12390	20.1%
German	468	1898	18.9%	3326	13546	18.5%
Spanish	422	1732	17.9%	2858	11825	17.2%

Table 3: Compression rates across languages using the *standard distribution* for the two sample texts. $Textlength$ is measured in number of letters and $Codelength$, in bits. French shows better compression rates than English for both texts, even though it needed more words to deliver the same meaning. Spanish presents the shorter length for both texts, but had the second worst compression rate.

Distances within language between text encodings		
$d(E_{S23}, E_{G1}) = 21$	$d(E_{S23}, E_{st}) = 17$	$d(E_{SG1}, E_{st}) = 14$
$d(F_{S23}, F_{G1}) = 23$	$d(E_{S23}, E_{st}) = 16$	$d(E_{G1}, E_{st}) = 13$
$d(G_{S23}, G_{G1}) = 19$	$d(E_{S23}, E_{st}) = 15$	$d(E_{G1}, E_{st}) = 16$
$d(S_{S23}, S_{G1}) = 14$	$d(E_{S23}, E_{st}) = 13$	$d(E_{G1}, E_{st}) = 13$

Table 4: Comparisons between encodings extracted from the letter frequency tables of the sample texts and the encoding from the *standard distribution*. The distances between the text codes are higher than the distance between either of them and the standard code. As the text-length increases, from Psalm 23 to Genesis 1, the distance between their encodings and the standard code decreases.

the frequency tables. The same reasoning could be applied to the other languages as well, since all of them are Biblical texts with possibly unusual words. Therefore, the distortions caused by unusual words should be cancelled out for each language. But, since languages evolve at different paces, the frequency of words which are considered rare for Biblical texts across languages may be different.

Third, having shorter text-lengths is distinct from having shorter code-lengths. As illustrated before, French had longer text-lengths than English but shorter code-lengths throughout the experiment.

Therefore, these findings are not counterintuitive when one consider statistical arguments. The results are simply a counterexample to the general trend towards better compressibility of English texts.

It should be emphasized, however, that these results are highly dependent on the coding paradigm chosen. In this experiment, for example, we chose codes with the Shannon-Fano coding property (Li and Vitányi, 1997) and, in particular, an instance of this approach called Huffman code.

One question follows from the above: why French outperformed English in the specific examples considered, namely Psalm 23 and Genesis 1? A possible reason is that French is highly redundant when it comes to letter frequencies. As pointed out earlier, the probability distribution of characters in French has very high peaks and low valleys. Since shorter codes are assigned more frequent letters and since these letters show up more often, the code-length tend to shorten.

This reasoning, however, is general and does not answer the question as to why those Biblical texts allow for better compression. It seems that the particular texts have nothing to do with the enhanced compressibility of French. Rather, it may be a property of the language itself: French appears to be more redundant than English.

One mechanism of the language to reduce redundancy consists of eliminating consecutive vowels by means of a quote (e.g. le amour becomes l'amour). Yet, as Pratt (1939) points out, texts in French tend to repeat letters, words and phrases several times even in the course of short messages.

Conditional events and sample biases

A hidden assumption behind letter frequency tables is that the probability distribution is the same throughout the message. A message (e.g. text) with such a property is called *ergodic*. This assumption, however, does not apply thoroughly to natural languages, since the probabilities of the various outcomes (e.g. what is the next letter) depend on the position of the letter in the message.

In Spanish, for instance, it is more likely to have a vowel after a consonant than another consonant. The same goes to 'wh-' constructions in English (is hard to imagine a consonant following 'wh'). Thus, the event space shrinks when it comes to the next letter after a consonant in Spanish or after a 'wh-' construction in English. Summarizing, certain *combinations of letters* are more or less likely than the simple product of the probabilities for the letters separately.

All this information about about language structure (character and word combinations) is not captured by the probability distribution that we used. This information may help to improve the precision of such distributions and, therefore, the compression rates of new encodings. Once such improvement is carried out, it would be interesting to test the new codes across languages and evaluate whether or not the better compressibility of French is conserved.

Another statistical issue from our experimental design is the sample size. Since it is so small, the results obtained might not represent a general trend. Instead, they may be amplifications of sample biases.

An example of such biases are the unexpected short text- and code-lengths of Spanish for both standard and text-specific codes. Genesis 1 consists of 12,593 letters in English and only 11,825 in Spanish. This goes against the general trend of Spanish texts being longer than English ones (the average word-length of Spanish is 4.96 letter per word). The result above can be explained by a characteristic of the text, and not by a general property of the languages.

Genesis 1 is a highly structured text. It exhibits several repetitions of the coordinating conjunction 'And' at the beginning of sentences. Both English and Spanish versions of the text have 71 occurrences of 'and' and 'y' (which means 'and' in Spanish). The code-word corresponding to these repetitions, however, is much shorter in Spanish than it is in English.

'And' requires 13 bits in the standard code, whereas 'y' takes 7 bits. So, the total code-length corresponding to the repetitions of the conjunction requires 923 bits in English and only 497 in Spanish. Since the difference of total code-length between the two texts is 768 bits, it follows that the difference in code-length between the pair 'and'/'y' accounts for 55.4% of the difference in code-length between the two versions of Genesis 1.

Thus, experiments across languages should take larger samples to avoid sample biases. Moreover, they should use more detailed distributions to achieve better codes and compressibility results.

Conclusion

Different languages communicate the same message using distinct codes. These messages vary in length. Moreover, the messages vary in structure – some present more regularities and redundancies than others. These text- and language-specific properties allow for variable-length codes, such as Huffman codes, to compress information. We carried out an experiment to assess how compressibility rates vary across languages using as a sample two texts from the Bible: Psalm 23 and Genesis 1. The main finding is that French showed better compression rates than the remaining languages even though it presented a less optimal code and longer text-lengths.

References

The Bible Gateway. http://bible.gospelcom.net/.

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest (2001). *Introduction to algorithms.* MIT Press.

Li, M. and P.M.B. Vitányi (1997). *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, New York.

Pratt, Fletcher (1939). *Secret and Urgent: the Story of Codes and Ciphers.* Blue Ribbon Books.