Simplicity Bias in Human-generated Data

Jean-Louis Dessalles (dessalles@telecom-paris.fr)
LTCI, Institut Polytechnique de Paris
France

Giovanni Sileno (g.sileno@uva.nl)
University of Amsterdam
the Netherlands

Abstract

Texts available on the Web have been generated by human minds. We observe that simple patterns are over-represented: `abcdef` is more frequent than `arfbxg` and `1000` appears more often than `1282`. We suggest that word frequency patterns can be predicted by cognitive models based on complexity minimization. Conversely, the observation of word frequencies offers an opportunity to infer particular cognitive mechanisms involved in their generation.

Keywords: Simplicity; Kolmogorov complexity; Web frequency; Zipf's law

Introduction

The observation of word or number frequencies reveals a few spectacular patterns. The most obvious one is that simple items tend to be more frequent. For example, round numbers such as 1000 or consecutive strings like `abcde` appear orders of magnitude more frequently than items such as `1282`) or `arfbxg`). Intuition suggests that some items are more frequent *because* they are descriptively simple. This form of causality, however, requires hypotheses about the way words and numbers are generated. In particular, we must suppose that individuals are biased toward simplicity.

Simplicity is expected to play a central role in cognitive modeling (Chater, 1999). Any reasonable induction system should favor simplicity (Solomonoff, 1964), and the human mind is no exception. Solomonoff also links abduction to simplicity, a principle known as Occam's razor. Simplicity is essential to define subjective probability (Saillenfest & Dessalles, 2015; Maguire, Moser, Maguire, & Keane, 2018; Griffiths & Tenenbaum, 2003). Pattern recognition, ranging from basic shapes to abstract IQ tests, is guided by simplicity (Chater, 1999; Feldman, 2004; Gauvrit, Zenil, Delahaye, & Soler-Toscano, 2014; Amalric, Wang, Pica, et al., 2017; Sablé-Meyer, Fagot, Caparos, van Kerkoerle, & Amalric, 2020). Analogy making is best explained as complexity minimization (Murena, Al-Ghossein, Dessalles, & Cornuéjols, 2020). Simplicity is observed to play a decisive role in predicting phenomena like surprise (Dessalles, 2008), memory (Planton, van Kerkoerle, Abbih, Maheu, & Meyniel, 2021; Chekaf, Cowan, & Mathy, 2016; Lemaire, Robinet, & Portrat, 2012), interestingness, or emotional intensity (Dessalles, 2011). Moreover, simplicity can be used to re-interpret and extend Bayes' rule (Sileno & Dessalles, 2022).

Several studies have applied simplicity principles to text data, for example, to measure text resemblance (Benedetto, Caglioti, & Loreto, 2002) or semantic distance between words (Cilibrasi & Vitányi, 2007). In this paper, we focus on the frequency of numbers and words. We will make several attempts to support the idea that complexity minimization at the cognitive level may produce some of the spectacular frequency patterns that can be observed in texts or on the Web.

The paper is organized as follows. We begin by defining the notion of simplicity as it is applied in cognitive modeling, and we discuss the way it is usually linked to data frequency. Then we make three attempts to offer causal explanations of word frequencies. First, we propose a basic generative model to explain why the frequency of alphabetic sequences such as passwords obeys a simplicity bias. Second, we examine number frequency distributions and propose an additional model that partly accounts for their remarkable simplicity patterns. Third, we examine word statistics; we show how Zipf's law on word frequencies points to certain hypotheses about complexity minimization in word generation.

Defining simplicity

Simplicity is approached here in terms of Kolmogorov complexity. The theoretical notion of Kolmogorov complexity $C(s)$ corresponds to the size of the smallest program p from which one can retrieve an object s.

$$C(s) = min(|p| : M(p) = s). \quad (1)$$

where $|p|$ designates the length of p in bits. M is a universal Turing machine (an ideal computer) on which p is run or, equivalently, a programming language in which p is written. The sequence `0,1,2,3,4,5,6,7,8` can be seen as simpler than another sequence `7,1,9,1,2,0,2,8`, which means that it can be expressed with less information. For instance, it can be generated in Python using the `range` function, which (applying optimal encoding principles) can be referenced by its position in the list of Python's most-used instructions, applied to `9`. Coding a basic function and one number requires less information than coding eight numbers.

Kolmogorov complexity can be regarded as the result of an ideal compression. A program p_0 of length $|p_0| = C(s)$ such that $M(p_0) = s$ appears to be a self-extracting compressed version of s. Simple objects are compressible, whereas complex objects, such as the second sequence above, are not and are considered random. Kolmogorov complexity can also

be seen as a context-dependent minimal description. For instance, the complexity of an integer N is about $\log_2(N)$ bits (using the binary coding of N). However, in a context in which another number M close to N has just been mentioned, $C(N)$ is about $C(N|M) \leq 1 + C(|N-M|) \leq 1 + \log_2(|N-M|)$ (the additional bit is used to code for the sign of the difference). Or, if N is stored in the machine's memory at the (binary) address a, then $C(N)$ is about length(a).

Kolmogorov complexity is often perceived as coming with two handicaps. The first one is the dependence on a machine M. It could be that there is no readily available function `range` in the machine or programming language. Conversely, `7,1,9,1,2,0,2,8` might be the owner's phone number and be already stored at a short address in the machine. This dependency on the machine is actually advantageous for cognitive modeling as it can be used to illustrate differences in background knowledge among observers. The "machine" to consider is the generic cognitive model used to represent an individual's capabilities for a given task (e.g. the ability to recognize a consecutive sequence), plus their specific knowledge (such as their phone number).

The second alleged drawback of Kolmogorov complexity is that it is proven to be incomputable: there can be no guarantee that a program of minimal length has ever been found, as it would require trying programs that won't halt. For scientific purposes, we need a resource-bounded version of Kolmogorov complexity (Buhrman, Fortnow, & Laplante, 2002). Human beings have limited computing power anyway and sometimes need time to detect hidden structures. Technically, we just need to limit recursion depth and computation time to obtain an operational approximation of $C(s)$.

Relation between complexity and frequency

The quantity $C(s)$ quite naturally measures the quantity of *information* contained in s, since it measures what is left when any redundant information has been removed in the operation of (ideal) compression. This notion of information generalizes Shannon's definition of information, which applies to repeated events and not to individual objects. If we no longer consider an object s in itself, but *occurrences* of s (eg. on the Web), then we expect the two notions to coincide:

$$C(s) \approx \log_2 \frac{1}{f(s)}, \tag{2}$$

where $f(s)$ is the observed frequency of s's occurrences. Inasmuch as it approximates the corresponding probability $p(s)$, $\log_2 {}^1\!/_{f(s)}$ approximates Shannon's information $\log_2 {}^1\!/_{p(s)}$ relative to s's occurrences. The right-hand term of (2) is purely observational. $C(s)$, on the other hand, refers to programs and naturally leads to considering the way data are generated. The present study takes (2) literally and attempts to show how frequency patterns emerge from the way data are generated. Our hypothesis is straightforward:

> (H): *human minds tend to minimize complexity; as a result, simpler items (words, numbers) tend to be more frequent.*

Password	Description	Complexity	Rank
111111	[repetition,[1],6]	8.8	9
333333	[repetition,[3],6]	9.8	28
12345	[increment,1,1,5]	10.2	7
123456	[increment,1,1,6]	10.4	1
555555	[repetition,[5],6]	10.4	17
1234567	[increment,1,1,7]	10.6	5
666666	[repetition,[6],6]	10.6	26
12345678	[increment,1,1,8]	10.8	6
777777	[repetition,[7],6]	10.8	32
123456789	[increment,1,1,9]	10.9	2
7777777	[repetition,[7],7]	11.0	19
888888	[repetition,[8],6]	11.0	21
121212	[repetition,[1,2],3]	11.6	48
654321	[increment, -1, repetition, [6], 2]	13.0	16
987654321	[increment, -1, repetition, [9], 2]	13.5	42
123123	[repetition,[increment,1,1,3],2]	14.2	10
aa123456	[a,a,increment,1,1,6]	14.4	41
abc123	[increment,1,a,3,increment,1,1,3]	18.2	11
lovely	[l,o,v,e,l,y]	34.5	18
qwerty	[q,w,e,increment,2,r,2,y]	36.5	3
1q2w3e4r	[merge,1,[increment,1,1,4],[q,w,e,r],0]	38.3	13
iloveyou	[increment,3,i,3,v,e,y,o,u]	42.5	8
sunshine	[s,u,n,s,h,i,n,e]	45.0	30
1q2w3e4r5t	[merge,1,[increment,1,1,5],[q,w,e,increment,2,r,2],0]	45.3	33
qwertyuiop	[q,w,e,increment,2,r,2,y,u,i,increment,1,o,2]	60.1	15

Table 1: Structure and complexity of popular passwords listed by frequency of use in (Keck, 2018).

What follows consists of three attempts to test (H) by showing how frequency distributions can be related to simplicity-oriented cognitive procedures. We will consider in turn alphabetic sequences, then numbers, and finally words.

Simplicity of short alphabetic sequences

In this section, we consider alphanumeric sequences such as `abc123` or `aa123456` that are used as passwords. Table 1 lists a few passwords taken from (Keck, 2018) and indicates their rank by frequency of use. We can see that `abc123` is 11$^{\text{th}}$ while `aa123456` comes in 41$^{\text{st}}$.

To test whether our hypothesis (H) obtains in this case, we must find a way to estimate the complexity of alphanumeric sequences. We designed a small "reasonable" model to capture some basic structural elements such as repetition or incrementation. The point is to compute plausible complexity values to see if they match observed frequencies. We selected six operators: `repetition`, `increment`, `periodic`, `merge`, `map` and `mirror`. Table 2 presents their syntax through examples. These operators have been manually selected, but they echo similar operations used in other proposals. For instance, the PISA compressor used in *Structural Information Theory* (SIT) (Leeuwenberg & Van der Helm, 2013) refers to *iteration* (similar to the `repetition` used here), *symmetry* (similar to `mirror`) and *alternation* (similar to `periodic`).[1]

Complexity is computed as follows: $C(N) = \log_2(1+N)$ for a positive integer; $C(N) = 1 + \log_2(1+N)$ for a nega-

[1] Compared to SIT, focusing only on structural aspects, our proposal also considers metrical components.

Instruction	Output
[repetition,[a],4]	[a,a,a,a]
[increment,2,a,4]	[a,c,e,g]
[periodic,1,[a],[increment,1,a,5],0]	[a,a,a,b,a,c,a,d,a,e]
[periodic,1,[a],[increment,1,a,5],1]	[a,a,b,a,c,a,d,a,e,a]
[merge,1,[increment,1,a,4],[increment,1,1,4]]	[a,1,b,2,c,3,d,4]
[map,[a,b],[repetition,4]]	[a,a,a,a,b,b,b,b]
[mirror,[increment,-1,d,4]]	[a,b,c,d,d,c,b,a]

Table 2: Examples to illustrate how our simple operators are applied to alphabetic sequences.

tive integer (the additional bit is paid for the sign); $C(X) = \log_2(1 + pos(X))$ for a character where $pos(X)$ is the position of the character X in the alphabet (starting from a); $C(O) = \log_2(1 + pos(O))$ for an operator where $pos(O)$ is the index of O in our list of six operators. Objects' type is known when in operand position. Otherwise, two bits are added to indicate which type (integer, character, operator) is chosen. Note that this code is space-delimited (like the Morse code)[2]. We tried to make reasonable choices when selecting and ordering operators or types to obtain an ordering of sequences by complexity. In our experiments, we found that variations in the order of operators had little impact on the output.

We implemented these computations into a Prolog program[3]. Complexity is computed through the recursive procedure presented in Algorithm 1.

Algorithm 1 Computation of String complexity

```
procedure: compress
Input: List
Output: description + complexity
if Operator can be applied to the beginning of List then
    call compress on Operator's arguments
    insert Operator and its compressed argument in description
else
    compute the complexity of the first element
    insert the first element in the description
end if
call compress on the rest of List
concatenate results and add complexities
```

We tested the procedure on the passwords of Table 1. The table mentions the complexity found by our program. Figure 1 shows complexity against password use for 8-letter alphabetic passwords, as retrieved from `https://haveibeenpwned.com/Passwords` in 2018[4]. The procedure does well in identifying the most frequent passwords as simple, except for "words" that our simple program has no means to make sense of (as it has no access to dictionaries). This result is consistent with the fact that what passwords Web users consider easy to remember are simple sequences, in the Kolmogorov sense. Note that the negative correlation between complexity and frequency of use is hardly apparent among the blue dots. It would be obvious if "good" passwords were included (maximal complexity and negligible use, all located in the top-left corner).

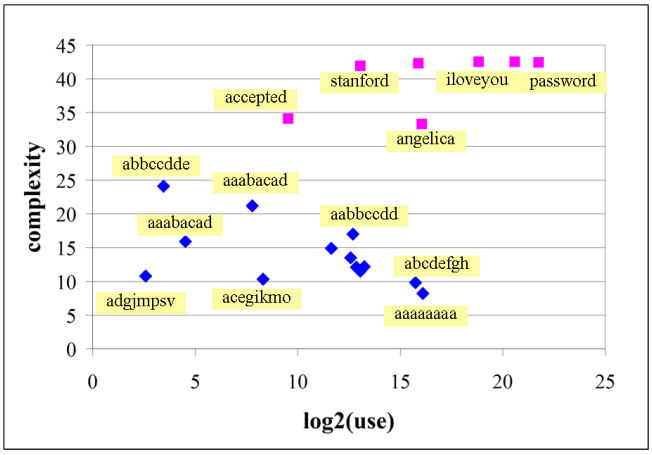

Figure 1: Complexity and use of a few 8-letter passwords, including some of the most popular ones (the complexity of words – pink dots – is overestimated, as the program has no access to a dictionary).

Number simplicity

In this section, we consider numbers, which can be seen as particular character sequences or even "words". We expect round numbers (in base 10), such as 1000, to be much more frequent in texts than less simple ones such as 1244. This is indeed what we observe. Figure 2 shows results obtained when querying numbers between 0 and 2050 on a search engine and retrieving the number of results[5]. The over-representation of multiples of 10, 50, and 100 is manifest. Several patterns also emerge at the global scale. We can see how the frequency of small numbers decreases exactly as $\log_2(N)$ (red curve). The sharp decrease in both directions apart from the present ($\sim$ 2023) suggests that most numbers around this value represent dates. Their frequencies also follow a logarithmic shape (green curve), which is consistent with a proximity effect.

Syntax and semantics of numbers

To test hypothesis (H) on numbers, we first applied our `compress` program designed for alphabetic sequences. Figure 3 shows the expected simplicity of numbers based on their pure alphanumeric (or syntactic) form. We observe some convergence with Figure 2, in line with (H): multiples of 10 and 100 emerge as simpler; the overall logarithmic decrease of simplicity is also apparent. However, we observe unwanted outcomes, such as 1111, 1234, or 1001 which stand out above

[2] Space delimitation is more natural and simplifies the computation of Kolmogorov complexity. When objects need to be separated, e.g. in a random generation context, a *prefix-free* version of Kolmogorov complexity must be used (Li & Vitányi, 1993).

[3] Sources can be found at https://www.simplicitytheory.science/strings.

[4] The hierarchy of password use has recently changed, probably due to constraints imposed by Web sites. Although global correlation with simplicity remains manifest in 2024, statistics tend now to mask differences among the simplest passwords.

[5] Yahoo, Bing and Google provide the number of results for each query.

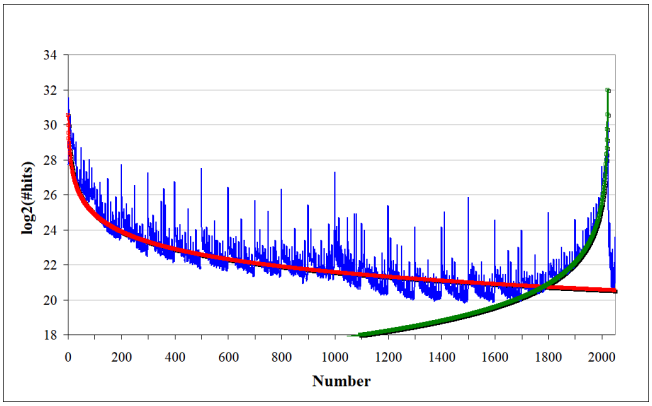

Figure 2: Querying numbers between 0 and 2050 on a search engine in logarithmic scale. The red curve is $\log_2(\#hits(0)) - \log_2(N)$, while the green curve is $\log_2(\#hits(2023)) - 1.42\log_2(\frac{2023-N}{0.35})$ (the results are displayed for Yahoo, but we obtained superposable results with Bing and Google).

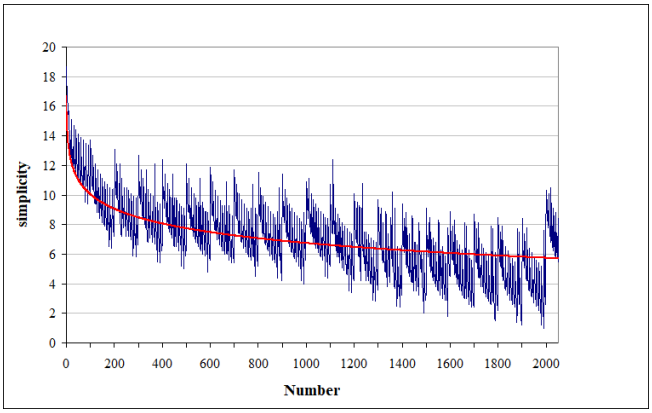

Figure 3: Simplicity (i.e. shifted opposite of complexity) of numbers between 0 and 2050, considered as mere character sequences (syntactic view).

1000. This shows that a purely syntactic approach (i.e. relative to form only) to number simplicity is insufficient.

Our second move was to consider the semantics of numbers. Numbers most often have a meaning as they represent quantities (even when numbers are used as ordinals instead of cardinals, they still mean a magnitude representing position). A typical computer knowing about numbers needs about $\log_2(N)$ bits to represent an integer N unambiguously. The same is true for human beings, at least verbally. We need four words, *twelve hundred [and] forty-four*, to represent 1244. The number of words (or digits) grows logarithmically with N. Representing number meaning as quantity involves logarithms as well (Feigenson, Dehaene, & Spelke, 2004). For instance, most individuals would negotiate a raise of their salary $x+dx$ not by considering the absolute value dx of the increase, but by evaluating its relative value dx/x when demanding $x(1+dx/x)$. The sensitivity to relative variations is

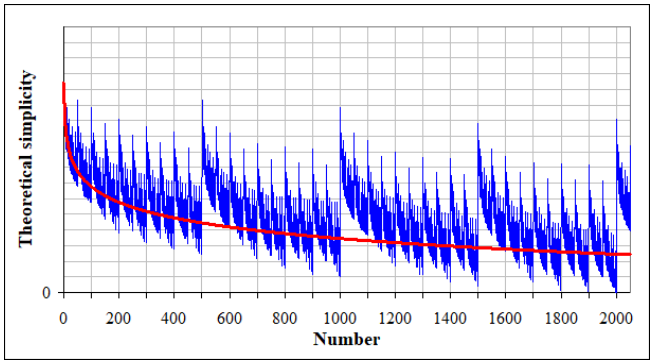

Figure 4: Simplicity when rounded numbers are systematically used as reference points to assess magnitude (semantic view).

a clear indication that human beings estimate magnitudes logarithmically (as $d(\log x) = dx/x$), and therefore represent numbers based on their complexity $C(N) \approx \log_2(1+N)$, keeping in mind that for most values of N this expression is close to the best estimate[6].

Round numbers are exceptions to this rule. Most contributors to texts found on the Web are using the decimal system. They rely on round numbers as *reference points* to locate precise numbers. They would use 1240 to locate 1244 and, if necessary, 1200 to locate 1240 and 1000 to locate 1200. As a result, we expect round numbers to become simple because of their role as reference points. For an arbitrary point N of the number line, the use of a reference point r verifies:

$$C(N) \leq C(r) + C(N|r) \tag{3}$$

This chain rule simply means that using r as a proxy is a reasonable, though possibly sub-optimal way to achieve the most concise description of N. In this writing, we implicitly suppose that the r giving the lowest upper bound has been picked. If r terminates with k zeroes in decimal base, we suppose that it is used as a reference point for the segment $[r, r+10^k[$ (subtracting r from N is a straightforward operation *above* the reference point r). We may suppose that using r as a proxy in the interval is not sub-optimal on average. As a result, $C(r)$ cannot deviate much from the average of $C(N) - C(N|r)$ for $N \in [r, r+10^k[$. If $C(N|r)$ within this interval is assessed as the magnitude of $(N-r)$, then $C(N|r) \approx \log_2(1+N-r)$. Taking the middle of the interval as average, $C(N)$ is averaged at $\log_2(1+r+10^k/2)$ while $C(N|r)$ is averaged at $\log_2 10^k/2$ (this approximation amounts to saying that $\log_2(N)$ is nearly constant in the interval). We obtain:

$$C(r) \approx \log_2(1+\frac{2r}{10^k}) \tag{4}$$

[6] As previously mentioned, this estimate of complexity works only if we consider numbers in isolation. When codes for numbers must be separated within a program, we have to use prefix complexity, which gives a typical estimate close to $\log_2(N) + 2\log_2(\log_2(N))$ for the complexity of $N > 0$ (Li & Vitányi, 1993).

Individuals use multiples of 10 as simple reference points r not only for locating absolute numbers N, but presumably also for assessing differences $(N-r)$. Evidence for this is provided by the tendency to celebrate "round" anniversaries of historical events, i.e. after 10, 50, or 100 years. If we use the above expression of $C(r)$ and use the chain rule recursively for r and $(N-r)$, we can compute the simplicity of numbers based on their proximity to multiples of 10. Figure 4 shows the resulting expected frequencies. The resemblance with real data (Figure 2) is again imperfect. However, the model succeeds in reproducing the overall logarithmic decrease (red line) and the simplicity effect observed after reference points. The obvious drawback is that the main reference points (500, 1000, 1500) are over-simple, while multiples of 10 and 100 are not simple enough. We can only conclude from our two attempts to reproduce the observations of Figure 2 that contributors to the Web are probably sensitive to both the syntactic and the semantic aspects of numbers, depending on context, in a way that remains to be explored.

Recency effects

Numbers around the present (~ 2023) are over-represented in Figure 2. The obvious interpretation is that most numbers in this zone should be viewed as dates. It seems "natural" that dates close to the present be more frequent. Could it be that they are frequent because they are cognitively simple?

A cognitive model based on simplicity would consider that any magnitude, including duration, is represented using a minimal code. As a result, in the absence of a reference point other than the present T_0, the cognitive complexity of an event located at time t must grow logarithmically with the time distance to the event:

$$C(t) = \log_2(1 + \frac{T_0 - t}{\tau}) \tag{5}$$

where τ is the characteristic timescale at which the class of events is considered (eg. days for seeing a friend, months for being sick, decade for moving house). This expression of $C(t)$ has been used to predict unexpected occurrences, as when the same type of event (eg. getting sick) occurs twice in a row independently, making the second occurrence "too simple" (Dessalles, 2008). If the dates of occurrence are t_1 and t_2, the coincidence is unexpected if $C(t_2|t_1) = log_2(1 + \frac{t_2 - t_1}{\tau})$ is close to 0.

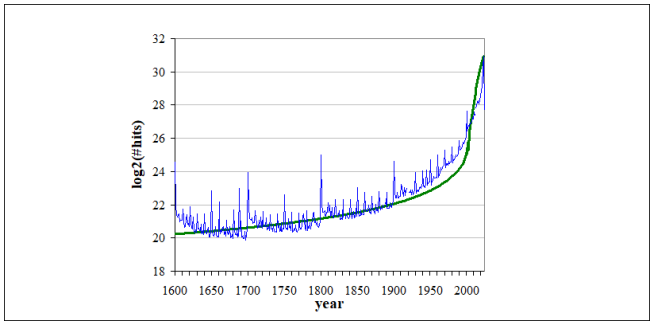

Figure 5: Model approximation of the frequency of years. The fitting curve is shifted vertically down by 4 bits to match actual Web frequencies.

We obtained a good numerical fit with empirical data close to 2023 in Figure 2 with the following logarithmic curve:

$$\log_2 f(T_0) - 1.42 \log_2(\frac{T_0 - t}{0.35}) \tag{6}$$

where $T_0 = 2023$. This fitting expression is reminiscent of (5) with $\tau = 0.35$. However, even if recent dates refer to events, it is not clear how the different values of τ would aggregate to an average value of 0.35. Moreover, the Web accumulates pages about the recent past while growing in size. From (*Total number of Websites*, 2019) we estimated the growth of the Web as $W(t) = 3500 \times (t - 1992)^4$. Then we estimated the number of pages created at time t about year y as $\frac{W(t)}{(t-y)/\tau}$ for $t > y$, or else 0 (with the previous estimate of $\tau = 0.35$). Then we integrated over t to get the cumulated number of pages about y created since 1992. The fit we obtain, shown in Figure 5, is not perfect but is at least reasonable.

Word frequency

After considering alphabetic sequences and numbers, we examine simplicity effects in word statistics. An empirical law first discovered by the physicist Edward Condon (Condon, 1928) and then known as Zipf's law reveals a systematic link between the frequency $f(w)$ of words in texts and their rank $r(w)$ by frequency in the same texts:

$$f(w) = \frac{k}{r(w)} \tag{7}$$

This relation translates into a straight line of slope -1 in log-log coordinates: $\log_2 f(w) = \log_2(k) - \log_2 r(w)$. Figure 6 shows that the law is robust (we may mix texts, even in different languages) and is a signature of natural texts (eg. slightly tampering with a text by merely removing so-called "stop words" perturbs the phenomenon, as shown bottom left of the figure). Zipf's law is not any power law: the slope is very close to -1 (Montemurro, 2001).

Here as well, we may gain a better understanding of this spectacular phenomenon by acknowledging that the human mind tends to minimize complexity (hypothesis (H)) when processing language. A cognitive model of language includes the ability to retrieve words for expressing meaning. Let's extend the machine metaphor and suppose that the meaning of a word w is accessible through its "address" $a(w)$. We are not considering binary addresses literally. Partial contents that trigger associative links can be regarded as addresses (a meaning can therefore be accessible through many of such "addresses"). The shortest address to a word w, available in the current context, offers concise access to its meaning. As such, its size $|a(w)|$ is an approximation of the word's cognitive complexity: $C(w) \approx |a(w)|$. Following George Zipf (Zipf, 1936, p. 19), we submit that addresses evolve, through

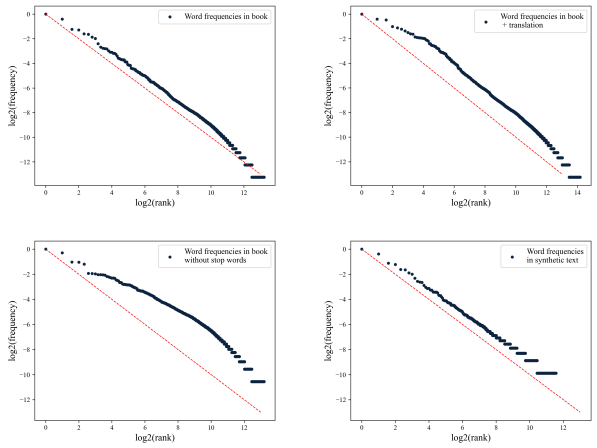

Figure 6: Word relative frequency *vs.* frequency rank for one of the authors' books (top left), the concatenation of the book and its translation (English + French) (top right), the book without English usual stop words (bottom left), and a synthetic text based on the book's vocabulary (bottom right).

learning, to be optimal. Being optimal for a system of addresses means that address lengths correspond to a Shannon-Fano code, i.e. they should be close to the logarithm of the word's frequencies: $|a(w)| = -\log_2(f(w))$. This principle is used in communication technology to guarantee minimal codes on average (e.g. with Huffman coding). To sum up:

$$\log_2(1/f(w)) \approx |a(w)| \approx C(w) \tag{8}$$

The preceding account goes from left to right: people learn word frequency from their language; they store meanings with optimal addresses whose size offers a measure of the meanings' cognitive complexity (averaged over different contexts). However, language is not only learned but also generated. To close the loop, we must have a right-to-left reading of (8). In line with hypothesis (H), we suppose that cognitive complexity controls meaning and word *generation*: meanings with smaller complexities trigger the use of the corresponding words with higher frequencies in a way that matches equation (2). As suggested by Zipf, each word w in a given context is in competition with synonyms, hypernyms and hyponyms. The winner tends to be the word compatible with the intended meaning that has the shortest address $a(w)$ in the context.

We just formulated two hypotheses to formalize Zipf's original intuition (Zipf, 1936). (H1): optimal length of mental addresses, and (H2): complexity-biased generation of words (which is a version of (H)). Now we can derive Zipf's law. Technically, the logarithm of the rank $\log_2 r(w)$ is sufficient to retrieve a word w. Hence it also offers an approximation of Kolmogorov complexity. We may write (up to an additive constant):

$$\log_2 \frac{1}{f(w)} \approx |a(w)| \approx C(w) \approx \log_2 r(w) \tag{9}$$

By using cognitive complexity $C(w)$ as a go-between, (9) shows how Zipf's law is an expected consequence of complexity minimization. If words are generated through optimal (i.e. Shannon-fano compliant) addresses, the probability for a word w accessible through address a to be chosen is (for binary addresses) $2^{-|a|}$. Texts generated this way will systematically verify Zipf's relation (7). As an illustration, Figure 6 (bottom right) shows the frequency-rank relation for a synthetic text generated with the lexicon of the same book and with the same frequencies.

Discussion

Equation (2) is often taken for granted, for it is widely cited in the scientific literature. It is nevertheless surprising as it establishes a coincidence between two separate notions. Following (Solomonoff, 1964), equation (2) can be read as a prediction. If the machine is optimal in the Kolmogorov sense, output frequencies should match complexities.

The present study goes the other way around. We start from frequency patterns in human-generated data and try to infer the cognitive mechanisms that generate them, while assuming that these mechanisms minimize complexity, i.e. achieve maximal compression. We did so on three types of data: alphabetic sequences, numbers, and words. In each case, we could make reasonable assumptions that make the frequency patterns less surprising. Notably, we were able to explain:

- the high frequency of simple passwords;
- the frequency pattern of numbers, which perfectly matches the logarithm in base 2 (red curve in Figure 2), where the logarithm in base 2 is the baseline for the Kolmogorov complexity of numbers (Li & Vitányi, 1993);
- the overrepresentation of round numbers and neighbors;
- the recency effects for dates close to the present;
- the inverse proportionality between frequency and frequency rank in the vocabulary (Zipf's law).

We are not aware of any principle, for example, one based on pure probability or Bayesian computations, which could account for all these phenomena without any reference to simplicity, in the sense of optimal compression.

Of course, much work still needs to be done to improve the accuracy of our simplicity-based explanations. Our goal was to highlight the relevance and feasibility of the approach. We are convinced, in line with (Chater, 1999), that simplicity in the sense of optimal compression remains under-exploited in Cognitive Science and should be considered a fundamental principle in cognitive modeling.

References

Amalric, M., Wang, L., Pica, P., et al. (2017). The language of geometry: Fast comprehension of geometrical primitives and rules in human adults and preschoolers. *PLOS Computational Biology*, *13*.

Benedetto, D., Caglioti, E., & Loreto, V. (2002). Language trees and zipping. *Physical Review Letters*, *88*.

Buhrman, H., Fortnow, L., & Laplante, S. (2002). Resource bounded kolmogorov complexity revisited. *SIAM Journal on Computing*, *31*, 887–905.

Chater, N. (1999). The search for simplicity: A fundamental cognitive principle? *The Quarterly Journal of Experimental Psychology*, *52*, 273–302.

Chekaf, M., Cowan, N., & Mathy, F. (2016). Chunk formation in immediate memory and how it relates to data compression. *Cognition*, *155*, 96–107.

Cilibrasi, R., & Vitányi, P. (2007). The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, *19*, 370–383.

Condon, E. U. (1928). Statistics of vocabulary. *Science*, *67*, 300.

Dessalles, J.-L. (2008). Coincidences and the encounter problem: A formal account. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proc. of the 30th annual conference of the cognitive science society* (pp. 2134–2139). Austin, TX: Cognitive Science Society.

Dessalles, J.-L. (2011). Simplicity effects in the experience of near-miss. In L. Carlson, C. Hoelscher, & T. F. Shipley (Eds.), *Proc. of the 33rd annual conference of the cognitive science society* (pp. 408–413). Austin, TX: Cognitive Science Society.

Feigenson, L., Dehaene, S., & Spelke, E. (2004). Core systems of number. *Trends in cognitive sciences*, *8*, 307–314.

Feldman, J. (2004). How surprising is a simple pattern? quantifying 'eureka! *Cognition*, *93*, 199–224.

Gauvrit, N., Zenil, H., Delahaye, J.-P., & Soler-Toscano, F. (2014). Algorithmic complexity for short binary strings applied to psychology: a primer. *Behavior Research Methods*, *46*, 732–744.

Griffiths, T. L., & Tenenbaum, J. B. (2003). Probability, algorithmic complexity, and subjective randomness. In R. Alterman & D. Kirsh (Eds.), *Proc. of the 25th annual conference of the cognitive science society* (pp. 480–485). L.E.A.

Keck, C. (2018). *It's time to nervously mock the 50 worst passwords of the year*. Gizmodo.

Leeuwenberg, E., & Van der Helm, P. A. (2013). *Structural information theory: The simplicity of visual form*. Cambridge University Press.

Lemaire, B., Robinet, V., & Portrat, S. (2012). Compression mechanisms in working memory. *Mathématiques & Sciences Humaines*, *199*, 71–84.

Li, M., & Vitányi, P. (1993). *An introduction to kolmogorov complexity and its applications (3rd ed.)* (, ed. 1997 ed.). New York: Springer Verlag.

Maguire, P., Moser, P., Maguire, R., & Keane, M. T. (2018). Why the conjunction effect is rarely a fallacy: How learning influences uncertainty and the conjunction rule. *Frontiers in Psychology*, *9*, 1011.

Montemurro, M. A. (2001). A generalization of the zipf-mandelbrot law in linguistics. *Physica A: Statistical Mechanics and its Applications*, *300*, 567–578.

Murena, P.-A., Al-Ghossein, M., Dessalles, J.-L., & Cornuéjols, A. (2020). Solving analogies on words based on minimal complexity transformation. In *Int. Joint Conference on Artificial Intelligence* (pp. 1848–1854).

Planton, S., van Kerkoerle, T., Abbih, L., Maheu, M., & Meyniel, F. (2021). A theory of memory for binary sequences: Evidence for a mental compression algorithm in humans. *PLOS Computational Biology*, *17*.

Sablé-Meyer, M., Fagot, J., Caparos, S., van Kerkoerle, T., & Amalric, M. (2020). Sensitivity to geometric shape regularity in humans and baboons: A putative signature of human singularity. *PsyArXiv*.

Saillenfest, A., & Dessalles, J.-L. (2015). Some probability judgements may rely on complexity assessments. In *Proc. of the 37th annual conference of the cognitive science society* (pp. 2069–2074).

Sileno, G., & Dessalles, J.-L. (2022). Unexpectedness and bayes' rule. In A. Cerone (Ed.), *3rd international workshop on cognition: Interdisciplinary foundations, models and applications (cifma)* (pp. 107–116). Springer Nature.

Solomonoff, R. J. (1964). A formal theory of inductive inference. *Information and Control*, *7*, 1–22.

Total number of websites. (2019). Retrieved from `https://www.internetlivestats.com/total-number-of-websites/`

Zipf, G. K. (1936). *The psycho-biology of language: An introduction to dynamic philology*. Routledge.