Katharina Ehret*

Through the compression glass: language complexity and the linguistic structure of compressed strings

https://doi.org/10.1515/lingvan-2022-0140
Received November 23, 2022; accepted August 28, 2023; published online March 25, 2024

Abstract: Against the backdrop of the sociolinguistic-typological complexity debate which is all about measuring, comparing and explaining language complexity, this article investigates how Kolmogorov-based information theoretic complexity relates to linguistic structures. Specifically, the linguistic structure of text which has been compressed with the text compression algorithm *gzip* will be analysed. One implementation of Kolmogorov-based language complexity is the compression technique (Ehret, Katharina. 2021. An information-theoretic view on language complexity and register variation: Compressing naturalistic corpus data. *Corpus Linguistics and Linguistic Theory* (2). 383–410) which employs *gzip* to measure language complexity in naturalistic text samples. In order to determine what type of structures compression algorithms like *gzip* capture, and how these compressed strings relate to linguistically meaningful structures, *gzip*'s lexicon output is retrieved and subjected to an in-depth analysis. As a case study, the compression technique is applied to the English version of Lewis Carroll's *Alice's Adventures in Wonderland* and its lexicon output is extracted. The results show that *gzip*-like algorithms sometimes capture linguistically meaningful structures which coincide, for instance, with lexical words or suffixes. However, many compressed sequences are linguistically unintelligible or simply do not coincide with any linguistically meaningful structures. Compression algorithms like *gzip* thus crucially capture purely formal structural regularities. As a consequence, information theoretic complexity, in this context, is a linguistically agnostic, purely structural measure of regularity and redundancy in texts.

Keywords: Kolmogorov complexity; language complexity; corpus linguistics; information theory; English

1 Introduction

This article is situated at the nexus of information theory, corpus linguistics, and research on language complexity. Specifically, it is a methodological exploration of a Kolmogorov-based complexity measure and analyses the linguistic structure of text which has been compressed with the *gzip* algorithm. Text compression algorithms (like *gzip*) are sometimes used to approximate Kolmogorov complexity as a means of assessing language complexity (e.g. Ehret 2021; Juola 2008; Sadeniemi et al. 2008).

In theoretical linguistics, language complexity has been a widely researched and much debated topic for the past two decades (e.g. Baerman et al. 2015; Baechler and Seiler 2016; Kortmann and Szmrecsanyi 2012; Mufwene et al. 2017).[1] Initially, the central question of this debate was whether, overall, all languages are equally complex or not. Thus, in answer to this question, language complexity has been measured at various linguistic levels such as morphology, syntax, or phonology, and, in some cases, at the overall structural level. Recently, the focus of the debate has shifted to comparing (Bentz et al. 2016) and evaluating existing complexity metrics (Berdicevskis et al.

1 In fact, language complexity has also been extensively researched in applied linguistics, cognitive linguistics/neuroscience, and other fields. This paper contributes to the debate in theoretical linguistics.

***Corresponding author: Katharina Ehret**, Department of English, University of Freiburg, Rempartstr. 15, 79098 Freiburg, Germany, E-mail: katharina.ehret@anglistik.uni-freiburg.de

2018; Ehret et al. 2021), and exploring new and better ways of measuring language complexity (von Prince and Demberg 2018).

One such recently explored measure is based on the notion of Kolmogorov complexity (Kolmogorov 1965), which defines the complexity of a text as proportional to the length of the shortest possible description of this text and can be conveniently approximated with text compression algorithms. This article focuses on the implementation of the Kolmogorov-based measure dubbed the *compression technique* (Ehret 2017). The technique has been applied to a wide range of naturalistic corpora such as the International Corpus of Learner English (Ehret and Szmrecsanyi 2019), the British National Corpus (Ehret 2021), or the SFU Opinion and Comments Corpus (Ehret and Taboada 2021). The compression technique employs the *gzip* algorithm for text compression. Essentially, texts which can be compressed comparatively more efficiently are comparatively less linguistically complex. The compression technique can also be used with various text modification procedures to address morphological and syntactic complexity in text samples. The algorithm, of course, is agnostic about form-function pairings so that Kolmogorov-based complexity measures have been described as measures of structural surface redundancy and (ir)regularity (Ehret 2017).

Against this backdrop, I describe and analyse compressed strings in order to arrive at a detailed linguistic definition of Kolmogorov-based language complexity. In particular, this article addresses the following questions: First, what kind of structures does the *gzip* algorithm actually capture and compress? Second, how do these structures, or compressed strings, relate to linguistic structures, in the sense of form-function pairings? As a case study, the compression technique is applied to Lewis Carroll's *Alice's Adventures in Wonderland* in English, and *gzip*'s lexicon output containing the compressed text sequences is subjected to an in-depth analysis.

This article is structured as follows: Section 2 provides some background on Kolmogorov complexity and the compression technique. Section 3 describes the general structure of *gzip*'s lexicon output. In Section 4 compressed strings are analysed and interpreted in linguistic terms. Section 5 concludes with a short summary.

2 Kolmogorov complexity

In linguistics, Kolmogorov complexity was first applied as a measure of language complexity by Juola (1998, 2008). Its implementation has since been extended and adapted for use with naturalistic, non-parallel corpora (e.g. Ehret and Szmrecsanyi 2016; Ehret 2021). The basic idea is to measure language complexity in texts by approximating their Kolmogorov complexity with compression algorithms.[2] The Kolmogorov complexity of a text is defined as the length of the shortest possible description of this text from which the original text can be reproduced (cf. Li et al. 2004: 3252). Take, for example, the two text strings in (1). Both strings consist of 10 characters, yet (1a) can be described by four characters, while the shortest possible description of (1b) is the string itself, with 10 characters. In terms of Kolmogorov complexity, then, the string in (1b) is more complex than the string in (1a).

(1) a. azazazazaz (10 symbols) → 5×az (4 symbols)
b. a4gh39fby8 (10 symbols) → a4gh39fby8 (10 symbols)

The implementation of the Kolmogorov-based measure described here has been dubbed the compression technique (Ehret 2017, 2021). It can be used to measure language complexity in texts at an overall level and, in combination with text modification procedures, at the morphological and syntactic level. The compression technique employs the open-source algorithm *gzip* for text compression and is available at https://github.com/katehret/compStrings.

Complexity at an overall level is measured by compressing original unaltered texts, and subsequently taking two measurements for each text: the size of the text file before compression and the size of the text file after compression. Based on these two measurements a score for overall complexity is calculated (for the exact computation, see Ehret [2021: 389–390]). Generally, comparatively larger scores indicate higher overall complexity in a text while comparatively lower scores indicate less overall complexity. This basic method can

2 To be precise, its upper bounds are approximated; see e.g. Li and Vitányi (1997).

be combined with two text modification procedures to indirectly measure morphological and syntactic complexity. Essentially, the morphological and syntactic information in texts is modified through random deletion before applying compression (for details, see Ehret [2021]). The morphological and syntactic complexity scores obtained with the compression technique thus basically index how well the compression algorithm deals with the noise created through the morphological or syntactic text modification procedures. In this article, the focus is put on the basic compression procedure as utilized to measure overall complexity.

3 Data

Text compression algorithms like *gzip* work on the assumption that texts always exhibit, to some extent, structural regularities and redundancies which can be reduced, that is, compressed. In very simplified terms, algorithms like *gzip* compress texts by describing and reducing upcoming text sequences on the basis of preceding text sequences which are temporarily stored in a kind of lexicon (Ziv and Lempel 1977: 337). It is this lexicon that is extracted and analysed in the following section.

In order to describe the general structure of *gzip*'s lexicon output, the English version of *Alice's Adventures in Wonderland* by Lewis Carroll is compressed, and the lexicon retrieved. The scripts and instructions for lexicon retrieval and processing are available at https://github.com/katehret/compStrings.

Listing 1 provides the first 20 lines of the *Alice* lexicon output. The lexicon is a line-by-line output of compressed sequences, that is, back-referenced sequences with length-distance pairs, and some literal, uncompressed text strings. In other words, it contains (sub)sequences which have previously occurred in the text. These redundant (sub)sequences are listed together with a reference of their length and the distance to the previous identical sequence (in the program's buffer). The minimum length for these sequences is three characters including spaces, so that the lexicon does not contain any back-referenced sequences shorter than three characters including spaces (Salomon 2007: 230–240). This general structure of the *gzip* lexicon is independent of the input text. However, the specific (text) (sub)strings and the length of the lexicon vary according to the content and length of the original text. Text-specific structural details will therefore be discussed in Section 4.

Listing 1: First 20 lines of *Alice's Adventures in Wonderland* lexicon output.

```
alice was beginning to get very tired of sitt
\[29 ,4] ing by her
\[15 ,3] sist
\[7 ,3] er on the bank an
\[41 ,5] d of hav
\[40 ,4] ing noth
\[77 ,7] ing to do
\[40 ,3] on
\[102 ,3] ce or tw
\[111 ,4] ice s
\[51 ,3] he had peep
\[94 ,3] ed in
\[37 ,3] to
\[71 ,5] the book
\[94 ,12] her sister
\[151 ,4] was read
\[120 ,5] ing but it
\[55 ,5] had no pictures
\[84 ,4] or con
\[171 ,3] versations
```

For illustration, let us now take a closer look at the structure of the *Alice* lexicon output. The first line contains no compressed sequences as there are no preceding text sequences on whose basis the text of the first line could be compressed. The subsequent entries all start with a backslash followed by square brackets containing the distance to the previous identical sequence and the length of the compressed sequence. The square brackets are immediately followed by the compressed sequence of the specified length (and, if applicable) literal text. For instance, line 6 `\[40,4]ing noth`, needs to be read as follows: the first integer in the square brackets indicates the distance of the referenced sequence to the previous identical sequence in the program's search buffer on whose basis the referenced string is compressed. The distance is given in characters and includes spaces, which are part of compressed sequences.[3] The second integer in the square brackets indicates the length of the referenced, that is, compressed, sequence. This means that the referenced sequence in line 6 was first encountered 40 characters before, and is four characters long. Thus, the sequence which is referenced and compressed is `ing_` which is followed by the uncompressed text string `noth`.

4 Through the compression glass

In this section the *gzip* lexicon output of the original English *Alice's Adventures in Wonderland* is examined, and compressed strings are interpreted in linguistic terms. As pointed out in the previous section, the length and the compression frequency of the specific (text) sequences that are captured and compressed depend on the input text. Yet, the exemplary analysis of the *Alice* lexicon yields insights into the general kind of strings that compression algorithms like *gzip* recognize and how they are mapped to linguistic structures.

The lexicon of the original *Alice* consists of 16,991 entries in total, and counts 11,683 unique strings. In linguistic terminology, then, the lexicon contains 16,991 tokens but only 11,683 types. In general, the number of unique strings decreases with increasing compression frequency: while there are only 15 highly frequent strings, strings which have been compressed only twice or less constitute, with a total count of 10,586, the largest group in the lexicon. Thus, over 90 percent of all compressed strings are very rare and occur only once or twice (see Table 2 in the Appendix for a tabular overview of strings and their compression frequency). As a matter of fact, the distribution of compressed strings in the lexicon follows a Zipfian distribution (Figure 1). This is an interesting but not unexpected observation because word frequencies in natural human languages are known to follow Zipf's law: the frequency of a word decreases exponentially to its frequency rank; that is, the probability of word occurrences in a natural human language starts off high but gradually decreases. In other words, only a relatively small number of words occur with high frequency while the majority of words occur rarely (Cancho and Solé 2003; Zipf 1935, 1949). This also applies to the frequency distribution of strings in the lexicon.

The distribution of compressed strings according to their length follows a similar pattern: the number of strings decreases with increasing length of the back-referenced sequence (see Table 3 in the Appendix). Specifically, about 85 % of the strings contain between three and 10 characters. Another 13 % contain between 11 and 18 characters. Many of the longer strings occur only once or twice, such as, for instance, (2) which contains 148 characters (Table 4).

(2) ome and join the dance will you won't you will you won't you will you join the dance will you won't you will you won't you won't you join the dance_

What kind of strings, then, does *gzip* recognize and compress? To give an example, the 15 most frequent strings in the *Alice* lexicon are displayed in Listing 2. These unique strings were compressed between 13 and 16 times and consist of single- or multi-word sequences ending with a space, such as `quite_` or `said alice_`. Other strings seem to be substrings of words, such as `ing_`. Strings with other compression frequencies are structurally similar such as `was_` (compression frequency = 12), `uddenly_` (compression frequency = 4), or the string `down down down_`, which was only compressed once.

3 Spaces at the beginning and end of compressed sequences are represented by an open box '␣'. Spaces within compressed sequences are represented by themselves.

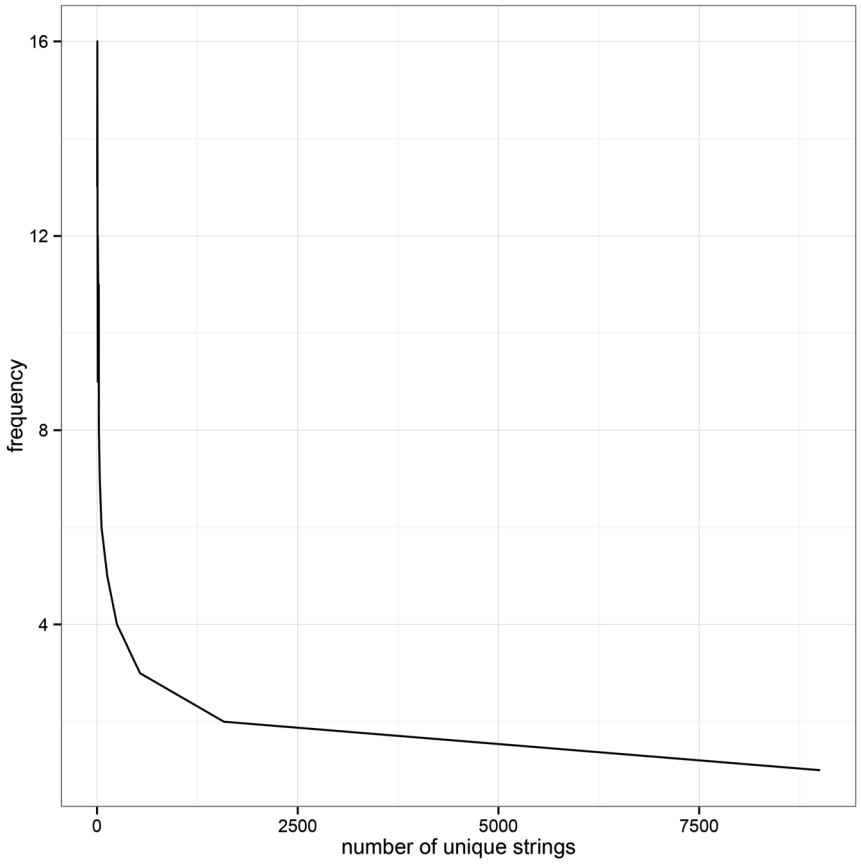

Figure 1: Distribution of unique strings in the lexicon of the original *Alice* text. The abscissa indexes number of unique strings, the ordinate indexes increased compression frequency.

Listing 2: Highly frequent unique strings in the *Alice* lexicon.

```
and she␣
and the␣
began␣
but she␣
had␣
ing␣
not␣
quite␣
said alice␣
said the␣
the␣
them␣
very␣
with␣
you␣
```

Having described the formal structural properties of compressed strings, let us now turn to the formal linguistic structure of strings. The referenced sequences in the *Alice* lexicon were manually analysed and, where possible, described in terms of common word class categories. In this spirit, the lexicon entries were grouped into four linguistically meaningful categories comprising the two the major word classes of (i) lexical words and (ii) function words, (iii) other linguistically interpretable strings, and (iv) meaningful phrasal patterns. Additionally, there are two categories comprising linguistically non-interpretable structures: (v) mixed strings and (vi) random

strings. Note that these categories do not distinguish between structurally identical but functionally or semantically different forms.

(i) Lexical words: This category includes nouns, verbs, adjectives, and adverbs (Biber et al. 1999: 62–66) as well as *to*-infinitives (e.g. *to see*) and established phrasal verbs (e.g. *make out*). Examples from the *Alice* lexicon are *hedgehogs*, *considering*, and *dreadfully*. For practical reasons, auxiliary forms of the verbs *have*, *be* and *do*, and the borderline cases *ought to*, *used to*, and *have to* were included in this category.

(ii) Function words: This category comprises prepositions, determiners, pronouns, coordinators, subordinators, numerals, the negator *not*, adverbial particles, and *wh*-words as well as modal verbs (Biber et al. 1999: 69–91). Inserts were also subsumed under this category – despite the fact that they constitute an independent, if somewhat ambiguous, class of words (for a discussion, see Biber et al. [1999: 56–57]) – because the greetings and response words occurring in the lexicon (e.g. *yes*, *please*) are more or less a closed word class (Biber et al. 1999: 56). Furthermore, semi-determiners (e.g. *same*, *such*) as well as quantifiers (e.g. *every*) and subordinators (e.g. *yet*) with multiple word class membership were by default coded as "functional".

(iii) Other strings: This category includes word endings and linguistically interpretable sequences. Specifically, it contains noun suffixes such as *-ment* or *-ity* (for a complete list, refer to Biber et al. [1999: 321]), genitive *'s*, verb endings such as *-ing* or *-ed*, and adjectival/adverbial endings such as *-ly* or *-est*. Parts of contractions such as *'ll* or *'ve* and the endings *-self* and *-ward(s)*, as well as *-ion* and *-ish*, were included in this category. Furthermore, any of the above forms plus one or more sequences from category (i) and (ii), such as the sequence `'s no use_`, were coded as "other".

(iv) Phrasal patterns: These strings are defined as multi-word strings including combinations and phrases of two or more intact words (e.g. `do cats eat bats_`, `there was nothing_`, `her sister_`) as well as contractions (e.g. `that's_`, `can't_`). In this context, phrasal patterns are not identical with prosodic or grammatical phrases as the algorithm lacks knowledge of syntactic units. Phrasal patterns may therefore be combinations of words that, in the original text, belong to different syntactic units or were formerly separated by punctuation marks such as `child said the_` and `gryphon we`, or cut-off word sequences such as `you ever_` or `the best_`.

(v) Mixed strings: Mixed strings contain at least one intact pattern from categories (i), (ii), or (iii) which are mixed with random sequences such as, for instance, `the b` or `abbit was_`.

(vi) Random strings: This category consists of random sequences and linguistically non-interpretable phrases such as `cks_` or `ich w`.

Based on these categories it is possible to broadly distinguish between linguistically meaningful strings as in example (3) and strings which are only partially interpretable or linguistically non-meaningful as in example (4).

(3) a. their_
b. looked anxiously_
c. _opportunity_
d. 'd better_

(4) a. s to f
b. dance t
c. gree_
d. omet

As displayed in Table 1, most of the compressed strings in the lexicon are mixed, that is they consist of a linguistically meaningful part and a non-interpretable part. Around 22 % of all strings can be identified as linguistically meaningful phrases, yet another 20 % are random sequences that cannot be identified as a proper word or phrase and are also not straightforwardly linguistically interpretable. In contrast, only a relatively small number of sequences can be identified as lexical or function words in a linguistic sense.[4] Although this means that

4 As pointed out before, this distribution depends on the content of the input text.

Table 1: Compressed strings in the *Alice* lexicon by category, raw frequency, and percentage. Note that percentages have been rounded down.

Category	Raw frequency	Percentage
Functional	913	5
Lexical	2,558	15
Mixed	6,170	36
Other	175	1
Phrasal	3,730	22
Random	3,445	20
Total	16,991	99

not quite half of the lexicon entries are linguistically meaningful sequences, lexicon-based compression algorithms like *gzip* do capture recurring linguistic structures – at least to some extent.

This poses the question of why *gzip* captures some instances of linguistic structures but not others. For instance, the preposition *with* occurs 180 times in total in *Alice's Adventures in Wonderland*, yet it was only recognized and referenced once. At the same time, the lexicon includes other strings containing the sequence `with` such as `with a_` or `with the_`. The reason for this is that lexicon-based algorithms like *gzip* compress texts by applying maximum length compression, that is the algorithm matches redundant sequences by back-referencing them to the longest previous identical sequence (Salomon 2007; Ziv and Lempel 1977: 377). In simple words, this means that the algorithm chooses longer sequences over shorter sequences no matter whether these sequences are words, linguistically meaningful structures, or linguistically non-interpretable sequences.

Example (5) illustrates maximum length compression. The text highlighted in bold serves as reference for the compression of the upcoming text passages (technically speaking, this is the text in the search buffer). The unmarked text represents the text which is supposed to be compressed (this is the text in the look-ahead buffer). The back-referenced sequences (i.e. the matches) and the previous identical sequence (i.e. the referent) are enclosed in square brackets. During text compression, *gzip* searches for the longest possible match to any sequence of characters and spaces stored in the search buffer. In (5a) the text in the search buffer is *alice was beginning to get very tired*. The longest possible match in the look-ahead buffer is the sequence `ing_` containing four characters. On the other hand, in (5b) there are two possible matches. The first possible match, `ing_`, contains four characters while the second possible match, `ing to the_`, contains 11 characters. In this case, *gzip* compresses the second match because it is the sequence of maximum length.

(5) a. **alice was beginn[ing␣]$_{\text{referent}}$ to get very tired** of sitt[ing␣]$_{\text{match}}$ by her sister on the bank …
b. **and began bow[ing to the␣]$_{\text{referent}}$ king** … and then turn[ing to the␣]$_{\text{match}}$ rose-tree

In example (5) both compressed sequences are linguistically interpretable and contain meaningful linguistic structures. However, algorithmically compressed strings, in particular strings containing sequences such as `ing`, which formally resemble linguistically meaningful structures, are not always actually linguistically meaningful. Compression algorithms, after all, are agnostic about form-function pairings and do not possess any knowledge of the meaning of the text. Therefore, algorithms neither distinguish between different functions of linguistic structures, for example, between gerunds and present participles as in (5a), nor do they distinguish between linguistically meaningful structures and formally identical, non-meaningful structures as in (6). Furthermore, in finding matches of maximum length, compression algorithms also reference linguistically non-meaningful sequences such as in example (7).

(6) **alice was beginn[ing to␣]$_{\text{referent}}$ get very tired** … and of having noth[ing to␣]$_{\text{match}}$ do

(7) **she found herself fa[lling␣]$_{\text{referent}}$ down a very deep well** … to drop the jar for fear of ki[lling␣]$_{\text{match}}$ somebody

As can be seen from this analysis, the *gzip* algorithm matches any sequence of characters and white space that is of maximum length – irrespective of its linguistic function or meaning. Therefore, a large number of sequences in the lexicon cannot be related to any linguistically meaningful structures. However, in many cases, these sequences coincide with linguistically interpretable and meaningful structures such as lexical and function words, suffixes, or multi-word phrases. In some cases, however, these sequences are only superficially similar to linguistically meaningful structures, in that they are similar in their form but not their function.

5 Summary

This article has explored the linguistic structure of compressed strings in order to gain a better understanding of Kolmogorov-based language complexity. As a case study, the compression technique (Ehret 2017, 2021), a Kolmogorov-based complexity metric, was applied to *Alice's Adventures in Wonderland* and a line-by-line lexicon output of compressed text sequences was extracted and analysed. Thus, the following two questions were addressed: First, what kind of structures does *gzip* capture and compress? Second, how do these structures relate to linguistic structures?

First, compression algorithms like *gzip* capture any sequence of characters and white space with a minimum length of three characters as long as this sequence can be matched to an identical sequence in the program's buffer. In other words, any type of structural regularity or redundancy in a text is captured irrespective of its meaning or function. As a consequence, algorithmically recognized strings can coincide with (i) linguistically meaningful structures, (ii) strings which formally resemble linguistically meaningful structures, and (iii) strings which are not linguistically interpretable.

Second, only some compressed strings are related to linguistically meaningful structures. Specifically, lexical words and function words, or multi-word patterns are sometimes – but not always and not systematically – captured. In some cases, suffixes but also formally identical sequences, that is, structures which superficially resemble suffixes such as *-ing*, are compressed.

These findings come with some important implications.

Kolmogorov-based language complexity is largely unrelated to the meaning, function, or grammar of the analysed text. This, of course, may bring into question the meaningfulness of information theoretic measurements as a whole and their relevance for assessing the complexity of a language. It might be argued, for instance, that the complexity of a language cannot be assessed without considering the meaning and function of linguistic features. In theory, this may certainly be true. However, a substantial number of publications (among others, Ehret and Taboada 2021; Ehret 2021; Ehret and Szmrecsanyi 2019) demonstrate that structural patterns are a reliable predictor for measuring a language's complexity.

In fact, the inherent agnosticism of the method is also an asset. Most measures of language complexity are based on the selection of linguistic features. This can potentially bias complexity measurements: On the one hand, such selections are inherently subjective as some features are selected over others, and, on the other hand, often entail the *a priori* categorization of linguistic features into simple and complex. In contrast, the compression technique, and information theoretic measures in general, do not suffer from such bias because they are holistic and unsupervised measures. In addition, the compression technique is universally applicable as it is not restricted to a single language but can readily be applied to many different and typologically diverse languages.

In short, the compression technique is a useful and reliable method and constitutes a valuable contribution to the quest for new and better ways of measuring language complexity.

Acknowledgments: I am grateful for feedback from Benedikt Szmrecsanyi on early versions of this article, and helpful comments by three anonymous reviewers. My thanks also go to the Cusanuswerk e.V. for funding this research.

Appendix

See Tables 2 and 3.

Table 2: Overview of unique strings in the *Alice* lexicon according to frequency of compression. The first column categorizes the strings according to their frequency.

Category	Unique strings	Compression frequency
Highly frequent	4	16
	4	15
	5	14
	2	13
Frequent	10	12
	21	11
	18	10
	9	9
	23	8
	34	7
	56	6
Rare	127	5
	248	4
	536	3
Very rare	1,580	2
	9,006	1

Table 3: Length and number of compressed strings in the *Alice* lexicon.

Length	Strings
3	826
4	1,961
5	2,700
6	2,465
7	2,228
8	1,940
9	1,408
10	985
11	582
12	467
13	316
14	279
15	175
16	147
17	112
18	100
19	57
20	59
21	28
22	44
23	20
24	14

Table 3: (continued)

Length	Strings
25	12
26	9
27	16
28	4
29	4
30	2
31	3
32	2
33	5
34	3
37	2
39	2
40	1
41	1
46	1
47	1
50	2
51	1
53	1
54	2
55	1
57	1
85	1
148	1

References

Baechler, Raffaela & Guido Seiler (eds.). 2016. *Complexity, isolation, and variation*. Berlin: De Gruyter.

Baerman, Matthew, Dunston Brown & Greville G. Corbett (eds.). 2015. *Understanding and measuring morphological complexity*. New York: Oxford University Press.

Bentz, Christian, Tatyana Ruzsics, Alexander Koplenig & Tanja Samardžić. 2016. A comparison between morphological complexity measures: Typological data vs. language corpora. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*. Osaka, Japan. Available at: http://www.aclweb.org/anthology/W16-4117.

Berdicevskis, Aleksandrs, Çağrı Çöltekin, Katharina Ehret, Kilu von Prince, Daniel Ross, Bill Thompson, Chunxiao Yan, Vera Demberg, Gary Lupyan, Taraka Rama & Christian Bentz. 2018. Using Universal Dependencies in cross-linguistic complexity research. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, 8–17. Association for Computational Linguistics.

Biber, Douglas, Stig Johansson, Geoffrey Leech, Susan Conrad & Edward Finegan. 1999. *Longman grammar of spoken and written English*. Harlow: Longman.

Cancho, Ferrer i Ramon & Ricard V. Solé. 2003. Least effort and the origins of scaling in human language. *Proceedings of the National Academy of Sciences of the United States of America* 100(3). 788–791.

Ehret, Katharina. 2017. *An information-theoretic approach to language complexity: Variation in naturalistic corpora*. Freiburg, Germany: University of Freiburg PhD thesis. Available at: https://freidok.uni-freiburg.de/data/12243.

Ehret, Katharina. 2021. An information-theoretic view on language complexity and register variation: Compressing naturalistic corpus data. *Corpus Linguistics and Linguistic Theory* 17(2). 383–410.

Ehret, Katharina, Alice Blumenthal-Dramé, Christian Bentz & Aleksandrs Berdicevskis. 2021. Meaning and measures: Interpreting and evaluating complexity metrics. *Frontiers in Communication* 6. 640510.

Ehret, Katharina & Benedikt Szmrecsanyi. 2016. An information-theoretic approach to assess linguistic complexity. In Raffaela Baechler & Guido Seiler (eds.), *Complexity, isolation, and variation*, 71–94. Berlin: Walter de Gruyter.

Ehret, Katharina & Benedikt Szmrecsanyi. 2019. Compressing learner language: An information-theoretic measure of complexity in SLA production data. *Second Language Research* 35(1). 23–45.

Ehret, Katharina & Maite Taboada. 2021. The interplay of complexity and subjectivity in opinionated discourse. *Discourse Studies* 23(2). 141–165.

Juola, Patrick. 1998. Measuring linguistic complexity: The morphological tier. *Journal of Quantitative Linguistics* 5(3). 206–213.

Juola, Patrick. 2008. Assessing linguistic complexity. In Matti Miestamo, Kaius Sinnemäki & Fred Karlsson (eds.), *Language complexity: Typology, contact, change*, 89–107. Amsterdam & Philadelphia: John Benjamins.

Kolmogorov, Andrej N. 1965. Three approaches to the quantitative definition of information. *Problemy Peredachi Informatsii* 1(1). 3–11.

Kortmann, Bernd & Benedikt Szmrecsanyi (eds.). 2012. *Linguistic complexity: Second language acquisition, indigenization, contact* (Lingua & Litterae). Berlin: Walter de Gruyter.

Li, Ming, Xin Chen, Xin Li, Bin Ma & Paul M. B. Vitányi. 2004. The similarity metric. *IEEE Transactions on Information Theory* 50(12). 3250–3264.

Li, Ming & Paul M. B. Vitányi. 1997. *An introduction to Kolmogorov complexity and its applications*. New York: Springer-Verlag.

Mufwene, Salikoko, Christophe Coupé & François Pellegrino. 2017. *Complexity in language: Developmental and evolutionary perspectives*. Cambridge: Cambridge University Press.

Sadeniemi, Markus, Kimmo Kettunen, Tiina Lindh-Knuutila & Timo Honkela. 2008. Complexity of European Union languages: A comparative approach. *Journal of Quantitative Linguistics* 15(2). 185–211.

Salomon, David. 2007. *Data compression: The complete reference*, 4th edn. London: Springer-Verlag.

von Prince, Kilu & Vera Demberg. 2018. POS tag perplexity as a measure of syntactic complexity. In *Proceedings of the First Shared Task on Measuring Language Complexity*, 20–25. Torun. Available at: http://www.christianbentz.de/MLC2018/proceedings.pdf#page=26.

Zipf, George Kingsley. 1935. *The psycho-biology of language: An introduction to dynamic philology*. Boston: Houghton-Mifflin.

Zipf, George Kingsley. 1949. *Human behavior and the principle of least effort: An introduction to human ecology*. Cambridge, MA: Addison-Wesley Press.

Ziv, Jacob & Abraham Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory* IT-23(3). 337–343.