# The Zipfian Challenge: Learning the statistical fingerprint of natural languages

**Christian Bentz**
Department of General Linguistics
University of Tübingen
chris@christianbentz.de

## Abstract

Human languages are often claimed to fundamentally differ from other communication systems. But what is it exactly that unites them as a separate category? This article proposes to approach this problem – here termed the *Zipfian Challenge* – as a standard classification task. A corpus with textual material from diverse writing systems and languages, as well as other symbolic and non-symbolic systems, is provided. These are subsequently used to train and test binary classification algorithms, assigning labels "writing" and "non-writing" to character strings of the test sets. The performance is generally high, reaching 98% accuracy for the best algorithms. Human languages emerge to have a *statistical fingerprint*: large unit inventories, high entropy, and few repetitions of adjacent units. This fingerprint can be used to tease them apart from other symbolic and non-symbolic systems.

## 1 Introduction

"If a Martian scientist [...] received from Earth the broadcast of an extensive speech [...] what criteria would [...]determine whether the reception represented the effect of an animate process on Earth, or merely the latest thunderstorm on Earth?" (Zipf, 1936, p. 187)

Zipf's ideas – condensed in the above quote – have spurred a whole research paradigm: the study of statistical laws of language. These have emerged as the best candidates for universals of language (Ferrer-i-Cancho, 2005, 2007; Bentz and Ferrer-i-Cancho, 2016; Takahira et al., 2016; Dębowski, 2020; G. Torre et al., 2021; Tanaka-Ishii, 2021; Petrini et al., 2023). Beyond languages, many other systems have been found to follow similar statistical laws – to the extent that their "meaningfulness" has been sometimes called into question (Miller, 1957; Li, 1992; Suzuki et al., 2005). Most recently, experimental investigations have shown that Zipfian distributions facilitate learning of linguistic and visual input (Lavi-Rotbain and Arnon, 2021, 2022, 2023), that they arise from human cognitive biases (Shufaniya and Arnon, 2022), and that they help with learning new word-referent mappings (Wolters et al., 2023). In this sense, such statistical laws are quite literally "meaningful".

However, the challenge posed in the quote above is still only partially addressed by research into statistical laws. Namely, a statistical pattern might universally occur across languages, but this does not entail that it is a *unique* feature of languages. The Zipfian Challenge is ultimately the search for a *statistical fingerprint*: a feature, or set of features, which uniquely identify human languages. This is related to an age-old controversy of the language sciences: What makes human language special – if anything?

This challenge is here broken down into a standard classification task. Assume you are provided with strings of characters:[1]

AALLAQQAASIUTA
SSSSCSOFSPPPFPP (1)

Is there an algorithm which robustly classifies these into "writing" and "non-writing"? – If yes, how? – If no, why not?

Beyond pure scientific curiosity, there would be concrete applications for such an algorithm: a) cleaning of contaminated corpora, especially when large and automatically crawled (Blevins and Zettlemoyer, 2022); b) measuring similarity of undeciphered scripts to known writing systems in order to help decipherement (Rao et al., 2009, 2010; Lee et al., 2010; Sproat, 2014); c) providing tools to systematically compare human language with animal communication (Kershenbaum et al., 2016).

[1]The first string is the beginning of the UDHR in Kalaallisut (West Greenlandic), the second is a transliteration of symbols in a wheather forecast.

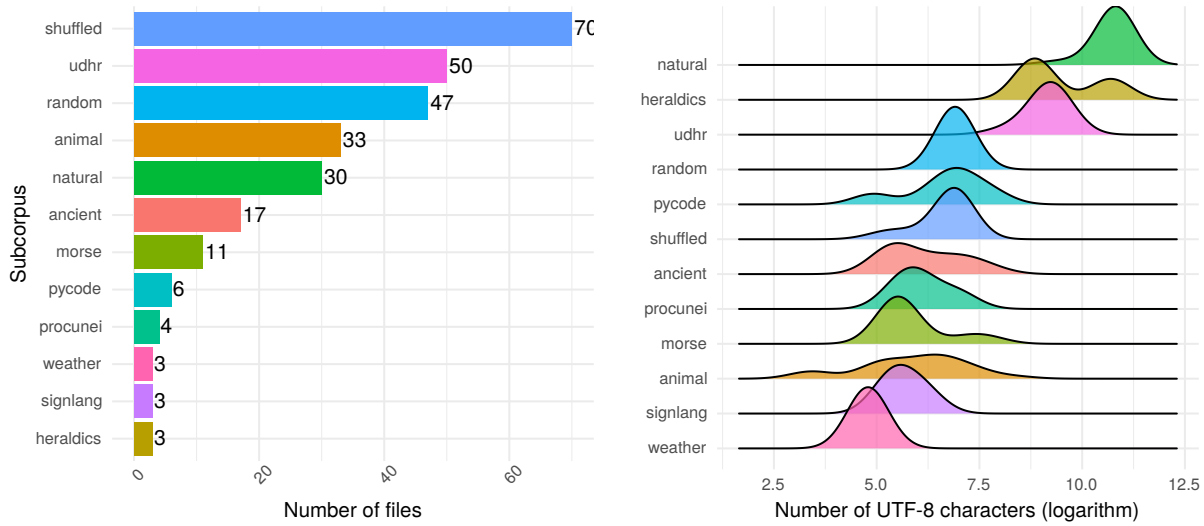


Figure 1: Number of files per subcorpus (left panel). Logarithm of the number of UTF-8 characters over files in a given subcorpus (right panel). Note that the natural logarithm of 50k is roughly 11, while for 500 this is roughly 6.

In the following, a corpus of character strings labelled as "writing" and "non-writing" is introduced in Section 2. Given this corpus, a sampling procedure is defined to retrieve strings of predefined lengths (10, 100, 1000). Subsequently, features from quantitative linguistics and information theory are described an calculated on the strings (Section 3). A series of classification algorithms are trained on a subset of the feature values. Section 4 then gives the results in terms of performance of the algorithms on the test sets. Section 5 discusses the results with regards to the original research question of a statistical fingerprint, as well as some follow-up questions which arise from the results.

## 2 Data

The data stems from a corpus of overall 377 files, split into "writing" (170 files) and "non-writing" (207).[2] The standard definition of *writing* is applied here. It refers to the tight link between spoken language structure and the graphemes representing it: "Broadly defined, writing represents speech. One must be able to recover the spoken word, unambiguously, from a system of visible marks in order for those marks to be considered writing," (Woods, 2010, p. 18). However, some transcriptions of sign languages are also included here. Arguably, unique structural features of a given sign language can be identified in a transcription system, in parallel to spoken language in its graphical form.

### 2.1 Writing

The writing files in this corpus consist of 50 parallel translations of the Universal Declaration of Human Rights (UDHR),[3] transcriptions of interactions in American Sign Language (ASL) and Sign Language of the Netherlands (SLN) according to the Berkeley system, as well as transliterations of ancient languages (Akkadian, Cretan Hieroglyphs, Proto-Elamite, Prakrit, and Sumerian).[4]

### 2.2 TeDDi sample

To increase the diversity of genres, registers, and modalities (spoken vs. written) for modern day languages beyond the UDHR, we furthermore draw 100 files randomly from the TeDDi (Text Data Diversity) sample (Moran et al., 2022). It includes more than 20K texts from overall 89 languages and 15 writing systems, and aims to maximize the diversity of families and areas represented.

### 2.3 Non-writing

The files classified as "non-writing" are further subdivided into songs of different bird species (animal), DNA strings (natural), python code

[2]Files and code can be found at https://github.com/christianbentz/NaLaFi.

[3]These were chosen to maximize the diversity of scripts. There are 36 different scripts in this sample according to the ISO 15924 standard.

[4]Mostly retrieved from https://cdli.mpiwg-berlin.mpg.de/.

(pycode), heraldics (heraldics), weather symbols (weather), morse code (morse), and proto-cuneiform (procunei). Examples are given in Table 1.

*Bird song* transcriptions of five different species (black-headed grosbeak, chickadee, Cassin's vireo, California thrasher, and zebra finch) are collected from an online database (Bird-DB).[5] It provides a "text" coding of recurrent phrases, identified by short pauses, and annotated with regular UTF-8 character strings in Praat (Arriaga et al., 2015).

*Heraldics* here refers to the description of heraldry (coats of arms) according to the so-called *Blazon* system. It has its own syntax, and uses a mixture of English and French words. It is here considered "non-writing" following the discussion in Sproat (2023). However, it is a borderline case. The usage of English words, inflectional morphemes, and noun phrase structures partially link it to the spoken language.

*Morse code* is another borderline case.[6] Graphemes of actual writing are here recoded into three morse characters (plus pause character). Hence, the actual writing can be recovered, and the underlying spoken language can be identified. However, this is a *two-stage* process. If we accept morse code as writing, we also have to accept, for instance, binary code. Such artificial coding schemes are here rather classified as "non-writing".

*Proto-cuneiform* is strictly speaking also "non-writing". Take, for instance, the transcription of a tablet from the Uruk III period (c. 3200-3000 BC)[7] as given in Table 1. *N14* and *N19* are transcriptions of sumerograms representing numbers (which are repeated several times for enumeration purposes), *SZE∼a* is an *iconic sign* which stands for the concept of "barley", and *LU2* for the concept of "person". In a strict sense, we do not know whether the scribe thought of the Sumerian spoken words for "barley" and "person" when they produced these iconic signs. They could have spoken any other language. As a consequence, the *language feature* of this tablet is assigned the value "undetermined" in the database.

Finally, two further sets of "non-writing" files are generated by a) randomly drawing up to 48 different characters from a uniform distribution, and b) randomly shuffling the characters of the "writing" files. Note that the latter process does not impact certain text statistics, e.g. the frequency distributions of characters. An overview of the file counts in this corpus, as well as distributions of file lengths in UTF-8 characters are given in Figure 1.

## 3 Methods

### 3.1 Preprocessing

The 377 files are preprocessed consistently to remove special characters which are used as annotations, rather than representing genuine information of the symbolic systems. For example, in Sumerian transliterations, curly brackets indicate so-called determinatives, as in *{d}nansze*, where *d* represents the star shaped sumerogram indicating that the next sumerogram is to be interpreted as the name of a deity, namely, the goddess *nansze*.[8] Note that the curly brackets are here already an interpretation of the person transliterating the original sumerograms, i.e. an annotation. The UTF-8 characters removed from all files include the tab character, as well as '{', '}', '(', ')', '[', ']', '+', and '*'. In fact, these characters also often cause problems in later processing steps, which is another – more practical – reason to remove them. Examples of preprocessed character strings are given in Table 1.

### 3.2 Sampling

While the numbers of files in the "writing" versus "non-writing" categories are relatively balanced (170 versus 207), the average file lengths in terms of UTF-8 characters differ widely. These range from c. 100 characters in the case of weather symbols, to c. 50k characters in the case of DNA (see also Figure 1, right panel). In most cases, this is due to data availability issues.

To alleviate this problem, two strategies are applied: Firstly, a maximum number of 10 strings of characters is extracted from each file. Secondly, the lengths of strings (in terms of number of UTF-8 characters) are held constant: 10, 100, 1000. We thus achieve a consistent comparison of strings of a given length across the different types of writing and non-writing systems. Also, these lengths are chosen with potential later applications

[5] http://taylor0.biology.ucla.edu/birdDBQuery/

[6] Thanks to one of the reviewers for raising this issue.

[7] https://cdli.mpiwg-berlin.mpg.de/artifacts/5353

[8] We here use the transliterations of sumerograms into Latin script. Mapping these back to UTF-8 sumerograms is currently not feasible.

| Corpus | Subcorpus | File ID | Example |
|---|---|---|---|
| **Writing** | Ancient | akk_0001 | šum-ma a-wi-lum ba-wi-lam u-ub-bi-ir-ma |
| | Signlang | tsl_0001 | -clVP-clTL-golVP_TOP-pstSTRmount-cl |
| | UDHR | cmn_0001 | 序言鉴于对人类家庭所有成员的固有尊严及其 |
| | | eng_0001 | Preamble Whereas recognition of the inherent |
| | | kal_0001 | AALLAQQAASIUTA taqqinassusermik inuup |
| | | kor_0001 | 전 문모든 인류 구성원의 천부의 존엄성과 동등 |
| | TeDDi | eng_nfi_242 | It's not supposed to be like this.It's time. |
| **Non-Writing** | Animal | bhg_0001 | uj kd ro su sv sw sx gf jr dw kd tc jt ag ta |
| | Heraldics | bla_0001 | Or, a lion rampant within a double tressure |
| | Morse | moc_0001 | phh_pppp_p_hp_s_pp_hp_s_h_pppp_p_s_hphp |
| | Natural (DNA) | dna_0001 | GGTAGTTAGGGTCTGAAAAAGATTTTGCG |
| | Proto-Cuneiform | prc_0001 | N14 [...] N19 N19 N19 SZE∼a LU2 MUD3∼d |
| | Python code | pyc_0001 | class Person: pass p = Person() print(p) class |
| | Random | ran_10 | hihhe bh fif cd gbgdiiigc ghigbbg af icegeebiifg |
| | Shuffled | eng_0001 | swr a j e eitimii hfeooa ti i d qs sfi roeviebg ep |
| | Weather | wsy_0001 | SWCCSSSSSSSSSSCSOFSPPPFPPFPP |

Table 1: Examples of characters strings of genuine writing systems as well as systems here classified as non-writing.

in mind. For example, when aiming to classify undeciphered scripts, or comparing human communication with animal communication, the strings available are often rather limited in length, in some cases just a couple hundred characters. Methods which need large amounts of data are not useful in this context. The sampling procedure is further illustrated in Appendix A.

Given this sampling procedure, we arrive at several thousand character strings for each predefined length (Table 2). For each of these strings, values are calculated for four quantitative features outlined in the following.

### 3.3 Features

The focus is here on quantitative features which have been explicitly proposed to distinguish different natural languages, and other symbolic systems (e.g. in Rao et al., 2009, 2010; Lee et al., 2010; Sproat, 2014; Bentz et al., 2017). In particular, the measures chosen are the type-token ratio (TTR), the unigram entropy ($H$), and the entropy rate ($h$) of units (i.e. UTF-8 characters), as well as the repetition rate of adjacent units ($R$). The exact definitions for these measures are given below.

#### 3.3.1 Type-token ratio (TTR)

The *type-token ratio* is defined as

$$TTR = \frac{C}{\sum_{i=1}^{C} f_i}, \tag{2}$$

where $C$ is number of character types in an "alphabet" $\mathcal{A}$, such that $C = |\mathcal{A}|$, and $f_i$ is the token frequency of a given character type $c_i$.

#### 3.3.2 Unigram character entropy (H)

Compared to TTR, the *unigram character entropy* is a more nuanced measure of diversity, reflecting the distribution of units. In general, it is defined as (Cover and Thomas, 2006, p. 14)

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x), \tag{3}$$

where $X$ is a discrete random variable, $\mathcal{X}$ is the alphabet, and $p(x)$ is the probability of a given type of the alphabet. In our case, we estimate the entropy with the maximum likelihood or 'plug in' method for a given string of characters $S$, such that

$$\widehat{H}(S) = -\sum_{i=1}^{C} \widehat{p}(c_i) \log_2 \widehat{p}(c_i), \tag{4}$$

where $S$ is assumed to be an i.i.d discrete random variable drawn from the alphabet $\mathcal{A}$, and $\widehat{p}(c_i)$ is the estimated probability, i.e. the relative frequency of a character $f_i$ in $S$. The unigram character entropy takes values in the range $[0, \infty]$. For an example sequence *abcabcabc* we have $\widehat{H}(X) = (1/3 \times \log_2(1/3)) \times 3 = 1.58$ bits/unit.

#### 3.3.3 Entropy rate (h)

While TTR and unigram entropy only take into account the frequencies/probabilities of individual characters – independent of their co-text – the *entropy rate* is defined for a stochastic process $\{X_i\}$ reflecting the concatenation of random variables, which might or might not be independent of one another. In general, the *entropy rate* is defined as (Cover and Thomas, 2006, p. 74)

$$h(\mathcal{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, X_3, \ldots, X_n). \quad (5)$$

This can be seen as the per symbol entropy growth. Note that in the case of characters in natural language texts, we have co-occurence patterns which limit the entropy growth to a certain extent. To estimate the entropy rate we turn to an estimator proposed in Gao et al. (2008), and implemented in Bentz et al. (2017). It is defined as

$$\hat{h}(S) = \frac{1}{n} \sum_{i=2}^{n} \frac{\log_2 i}{L_i}, \quad (6)$$

where $n$ is the length (number of characters) in a given string $S$, and $L_i$ is the length (+1) of the longest contiguous substring starting at position $i$ which is also present in $i = 2$ to $i - 1$. The entropy rate also takes values in the range $[0, \infty]$. For our regular *abcabcabc* string we get $\hat{h} = 0.84$ bits/character. Notice that this is lower than the value for the unigram character entropy (1.58 bits/character). This is because the same substring *abc* is repeated several times. In a sense, this entropy rate estimator "penalizes" long substrings of repetitions when calculating the entropy of a given string.

#### 3.3.4 Repetition rate (R)

Finally, the *repetition rate* (for adjacent characters) is proposed in Lee et al. (2010) and Sproat (2014) as an alternative to entropy estimation for teasing apart writing from non-writing. The general idea is that consecutive repetitions of characters are dispreferred in genuine writing systems – probably reflecting the avoidance of adjacent repetitions of phonemes in spoken languages. While there are some extreme examples like *Schifffahrt* in Standard German, we rarely encounter more than two repetitions of the same character in adjacency, and even these are relatively infrequent. The repetition rate is calculated as

$$R = \frac{r}{\sum_{i=1}^{C} f_i - 1}, \quad (7)$$

| Length (Chars.) | Overall | Training | Test |
|---|---|---|---|
| 10 | 3741 | 2543 | 1198 |
| 100 | 3223 | 2194 | 1029 |
| 1000 | 1832 | 1261 | 571 |

Table 2: Number of character strings of a given length in the training and test sets.

where $r$ is the number of adjacent repetitions of characters $c_i$ in a given string, and the denominator is the possible number of adjacent repetitions. $R$ takes values in the range $[0, 1]$. In the string *abcabcabc* we have zero adjacent repetitions of the same character, while there could be $(3-1)+(3-1)+(3-1) = 6$ repetitions. The repetition rate is then $R = 0/6 = 0$. For comparison, in the string *baccbcaab* (which has the same TTR and $H$ as before), we have *cc* and *aa* as adjacent repetitions, and hence $R = 2/6 = 0.33$.

Overall, we thus have four vectors of feature values. The estimated values are visualized in Figure 2. Some general trends are already visible in these panels. For instance, the marginal density distributions of writing and non-writing overlap considerably for the TTR, such that it will be hard for a classification algorithm to distinguish these in this dimension. For the repetition rate $R$ (y-axes on the right panels), on the other hand, the values of writing cluster more strongly towards low values, and are more spread out for non-writing. Interestingly, the shuffled strings seem to move away towards higher values in the $R$ dimension compared to the original writing strings. This suggests that random shuffling of characters introduces systematically more adjacent repetitions than found in real text.

### 3.4 Training and test sets

The feature values along with their labels (writing vs. non-writing) are split into a training and test set by the ratio 67% to %33. The resulting numbers for the training and test sets per string length are given in Table 2. The same training and test sets are used for all algorithms.

### 3.5 Classification Algorithms

#### 3.5.1 K Nearest Neighbors (KNN)

The KNN algorithm computes euclidean distances for each data point in the test set with each data point in the training set. It then classifies a given target point in the test set based on a majority vote

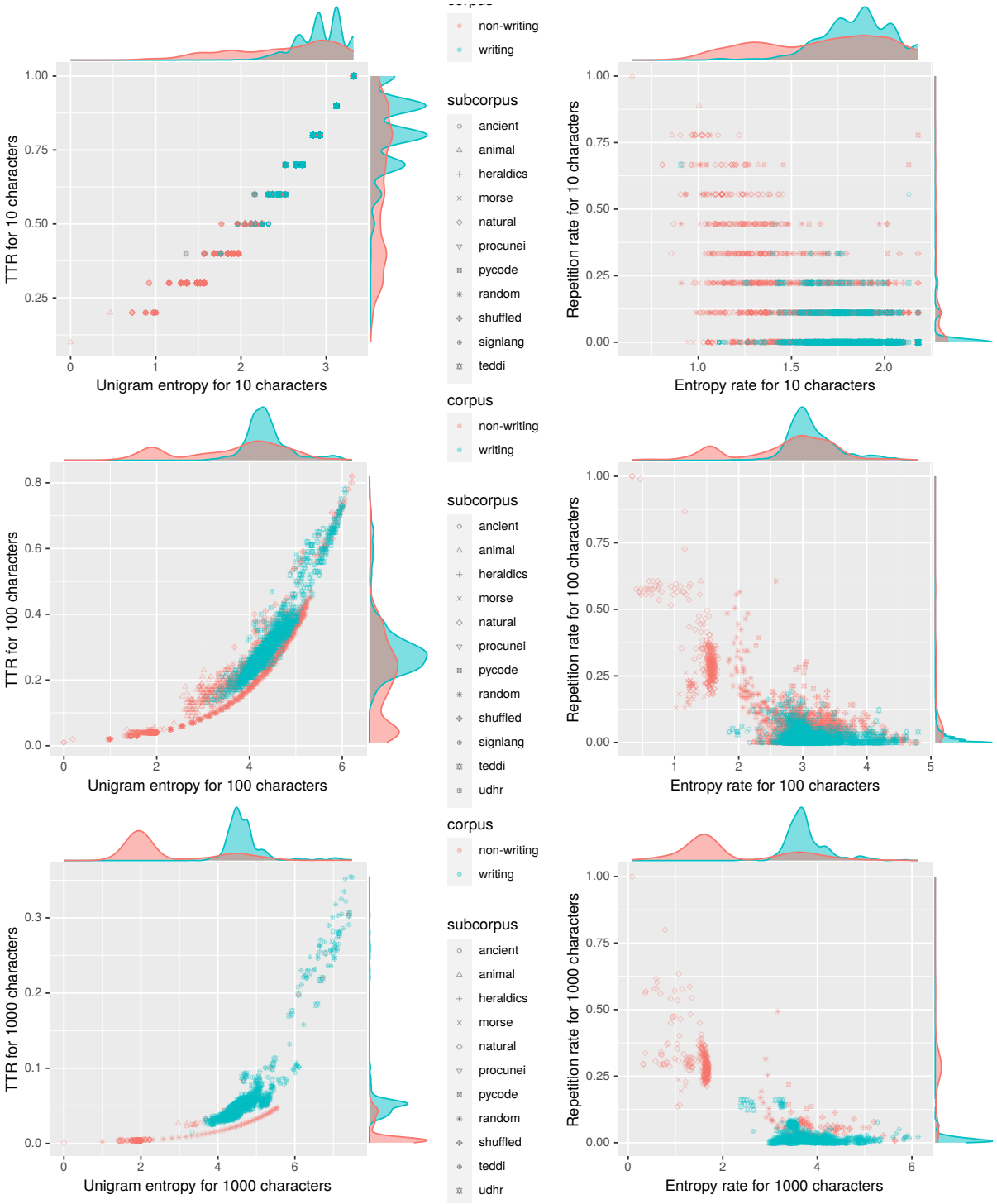


Figure 2: Distributions of feature values for strings of length 10, 100, and 1000 respectively. The main distinction between writing and non-writing is color-coded (blue and red). The subcorpora are indicated by different shapes of the dots.

of the class labels which the $k$ neighbours nearest to the target point have. Ties are broken at random. This is a non-parametric and fast classification algorithm. It was proposed already in Fix and Hodges (1952), and is still competitive today for general classification problems such as the XOR distribution of data points.[9] The only hyperparameter to tune is $k$, which is here assumed to range in between 1 and 10.

[9] See leader board at https://paperswithcode.com/task/classification (last accessed 29/06/2023).

#### 3.5.2 Logistic regression

Logistic regression is a parametric technique which was widely used in statistical learning for binary classification before the advent of neural networks. It is still used today in experimental studies in psychology and psycholinguistics (Baayen, 2013). For binary classification using feature values, we first need to estimate the coefficients of the logistic model, which is specified in our case as

$$\text{logit}(Y) = \log(\frac{P(Y=1)}{1-P(Y=1)}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4, \quad (8)$$

where $X_1, \ldots, X_4$ are random variables representing the feature values, $Y$ is the binary outcome variable we want to predict, and $\beta_0, \ldots, \beta_4$ are the parameters (coefficients) of the model which are learned (estimated) using the feature values and labels of the training set. Once these parameters are estimated, we use them for prediction of labels in the test set given the formula

$$P(Y=1) = \frac{1}{1+e^{-(\widehat{\beta_0}+\widehat{\beta_1}X_1+\widehat{\beta_2}X_2+\widehat{\beta_3}X_3+\widehat{\beta_4}X_4)}}, \quad (9)$$

with the decision rule: if $P(Y=1) > 0.5$, then assign label "writing", otherwise assign label "non-writing".

#### 3.5.3 Support Vector Machines

A support vector machine (Cortes and Vapnik, 1995) uses the input vectors of the training set – in our case $(\boldsymbol{x}_{TTR}, \boldsymbol{x}_H, \boldsymbol{x}_h, \boldsymbol{x}_R)$ – to find the hyperplane with dimensions $n-1$ (where $n$ is the number of features, i.e. $n-1=3$), which maximizes the margins to the nearest data points (i.e. support vectors). Data points in the test set are then classified according to the position of the hyperplane established with the training set. If the training data cannot be separated without error (which is almost always the case), then instead the number of errors is minimzed. As pointed out by Goodfellow et al. (2016, p. 141), the original formulation of SVMs is very similar to the logistic regression model given in Equation 8. However, it was subsequently shown that the so-called *kernel trick* can be used to allow non-linear mappings. The main hyperparameter is then the type of kernel used. Here, the *linear*, *radial basis*, *sigmoid*, and *polynomial* kernels are tested.

#### 3.5.4 Multilayer Perceptrons (MLP)

Multilayer perceptrons (deep feedforward networks) are the archetype of deep learning (Bengio et al., 2000; LeCun et al., 2015). In its simplest form, a feedforward network for binary classification consists of the input units (four in our case), a single hidden unit, and an output unit. See Figure 3 (upper panel) for an illustration. Note that this is mathematically equivalent to the logistic regression model in Equation 8. Namely, the vector of weights ($\boldsymbol{w}$) – multiplied with the input values of features ($\boldsymbol{x}$) – is equivalent to the coefficients $(\beta_1, \ldots, \beta_4)$, and the bias (indicated in blue in the figures) is equivalent to $\beta_0$.

However, a crucial question is which hidden layer architecture, activation function, error function, and backpropagation algorithm yield the best results for a given data set. These are the hyperparameters to tune. Here, a search of the space of possible architectures is performed by randomly drawing natural numbers in the range $[1, 4]$ for the hidden layers, and numbers in the range $[1, 5]$ for the number of units in each hidden layer. The maximal values are guided by local regression analyses of model performance (F1 score) given the depth and size of networks (see Appendix B). Overall, one hundred random values are drawn for the depth and size, yielding one hundred different architectures (out of $5^4 = 625$). Moreover, different activation functions (logistic, ReLU, softplus, tanh), error functions (SSE, cross entropy), and backpropagation algorithms (Rumelhart et al., 1986; Riedmiller and Braun, 1993; Hinton et al., 2006) are considered.

## 4 Results

For all classification algorithms the accuracy, precision, recall, and F1 score on the test set are reported alongside the respective hyperparameters. A condensed overview of classification results are given in Table 3. The best model overall is an MLP trained on feature values of strings with 1000 characters. It achieves an F1 score of 0.96, and an accuracy of 98%. In other words, for the 571 strings of the test set it assigns the correct label (writing vs. non-writing) in 560 cases, erring only in 11 cases. This performance drops to 93% accuracy when feature values of strings of length 100 are supplied, and to 73% with strings of length 10. The performance of the best KNNs is very similar, differing only by a max amount of 0.01. In gen-

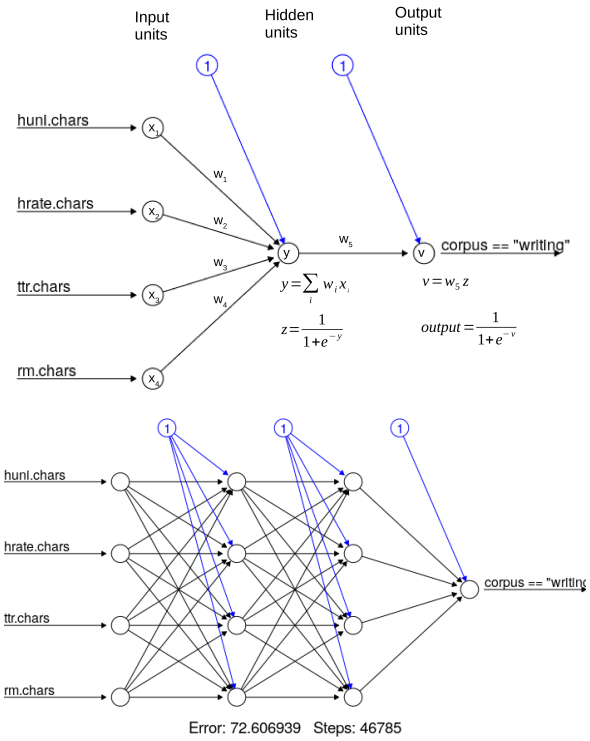


Figure 3: Upper panel: A forward pass with logistic activation and output functions with the simplest possible MLP architecture for binary classification, with one hidden layer, consisting of a single hidden unit. Lower panel: MLP architecture with two layers of hidden units (four each) and a logistic output unit. This is the architecture which performs best on strings of 100 characters.

eral, the KNN and MLPs show very similar performance, while the performance of SVMs and logistic regression models is lower across the board.

## 5 Discussion

Overall, the classification results suggest that the Zipfian Challenge is indeed a solvable problem. Namely, given strings of characters of length 100, KNNs and MLPs reach performance values of 0.92 and 0.93 respectively. With 1000 characters, they are almost at the ceiling of performance. In fact, it is questionable whether humans would be able to correctly classify the respective strings with 100% accuracy. Mind you that more than 36 different scripts and 90 different languages are represented in this data sample. It would be an interesting project for future research to establish human performance on this task. In the following, some further follow-up questions are briefly discussed.

### 5.1 Why do algorithms perform differently?

It is surprising to see a simple, non-parametric classification algorithm like KNN outperform other, much more complex algorithms such as logistic regression and SVMs, and perform on a par with the best MLPs. This is certainly related to the data set and problem at hand. The KNN has no parameters to "learn" from the training data. It directly assigns a label to a given vector of feature values by finding the vector of feature values closest to it in the training set. In comparison, the currently best MLP given in Figure 3 has $4\times4+4\times4+2\times1 = 34$ weights and $4+4+1 = 9$ biases to adjust. This amounts to overall 43 parameters to optimize in the "learning" process. In fact, few of the deeper networks with three or four hidden layers actually reach convergence with this data. And when they converge, they do not necessarily perform better than the simpler architectures (see Appendix B).

### 5.2 Why do longer strings yield better results than shorter strings?

The main reason for this is that the respective feature values have not converged for short strings of length 10. For strings of length 100, they start to converge in most cases, and at 1000 characters they have converged across the board. The convergence behavior of the different measures is given in Appendix C.

### 5.3 Which is the best feature?

When feature value vectors are input separately – rather than together – into the KNN algorithm (with $k = 1$), then the repetition rate $R$ performs best for 100 characters (F1-score: 0.8), followed by TTR (0.66), with unigram entropy and entropy rate at only 0.63. For 1000 characters, $R$ and TTR are similar (0.83 and 0.81), again with entropy measures yielding lower F1-scores (0.7 and 0.72). This squarely confirms the argument in Sproat (2014), namely, that the repetition rate $R$ is better than entropic measures for distinguishing writing from non-writing. However, if we remove entropic measures for the best KNN at 100 characters ($k = 5$), then the performance drops from 0.92 to 0.82. So they still considerably contribute to performance. For instance, for some natural language writing, e.g. the Kalaallisut string AALLAQQAASIUTA in Table 1, the repetition rate can be relatively high due to writing conventions

| Classifier | Chars. | Hyperparam. | Acc. | Prec. | Rec. | F1 |
|---|---|---|---|---|---|---|
| Baseline (only TTR) | 10 | k = 1 | 0.69 | **0.89** | 0.48 | 0.63 |
| KNN | 10 | k = 6 | 0.71 | 0.73 | **0.72** | **0.73** |
| | 100 | k = 5 | 0.92 | 0.92 | **0.92** | 0.92 |
| | 1000 | k = 7 | 0.98 | 0.98 | 0.92 | 0.95 |
| LogRegr. | 10 | – | 0.72 | 0.77 | 0.67 | 0.72 |
| | 100 | – | 0.79 | 0.84 | 0.71 | 0.77 |
| | 1000 | – | 0.93 | 0.95 | 0.75 | 0.84 |
| SVM | 10 | kernel: linear | 0.72 | 0.83 | 0.60 | 0.70 |
| | 100 | kernel: radial | 0.88 | 0.87 | 0.90 | 0.89 |
| | 1000 | kernel: radial | 0.92 | **1.00** | 0.70 | 0.82 |
| MLP | 10 | hidden: 5, 4; tanh; SSE; rprop+ | **0.73** | 0.78 | 0.69 | **0.73** |
| | 100 | hidden: 4, 4; tanh; SSE; rprop+ | **0.93** | **0.93** | **0.92** | **0.93** |
| | 1000 | hidden: 4, 5, 2; tanh; SSE; rprop+ | **0.98** | 0.99 | **0.94** | **0.96** |

Table 3: Classification results organised by number of characters and method. Only the best models (by F1 and Accuracy) for each number of characters is given. The baseline is the KNN algorithm (k=1) with strings of 10 characters and only TTR as a feature for training and testing.

for long vowels (aa), lateral glides (ll), and ejectives (qq). In such cases, the other measures will help with correct classification.

### 5.4 How are the results influenced by subcorpora?

The corpus of strings is not fully balanced. To get an idea of the degree to which particular subcorpora influence the performance, they are removed individually in a *post hoc* experiment with the best KNN model ($k = 5$) for 100 characters. The results are given in Appendix D. Generally, classification results are robust to removal of subcorpora. The strongest decrease in performance is associated with the removal of DNA (natural) strings. These have generally low entropies, and high repetition rates, and are hence easily classified as non-writing. The inverse effect holds for shuffled data. Shuffling the characters of genuine writing does not change the unigram entropy and TTR, and only marginally changes the entropy rate of strings. Hence, in this case, the repetition rate is the only feature useful for identifying the resulting strings as non-writing. Removing the shuffled strings increases the overall performance.

## 6 Conclusions

Compared to other symbolic and non-symbolic systems, natural languages seem to exhibit a unique fingerprint: relatively large unit inventories, relatively high entropy, and relatively few repetitions of adjacent units. This statistical fingerprint can be used to identify written language with high accuracy when more than 100 characters are provided. Interestingly, this seems to hold not only for writing reflecting spoken language but also for transcriptions of sign languages (though only small samples of ASL and SLN were used here). This suggests that humans have evolved the capacity of encoding information with a diverse, non-repetitive succession of units in three modalities: speech, manual signs, and graphical codes. If these results hold true, then it is not a single feature, and not a single modality, which defines human language, but a set of features related to rapid information transmission in the face of space and time limitations.

## Acknowledgements

Thanks to Tanja Samardžić, the *CrossLingference* working group, as well as the members of the *Language Learning and Processing Lab* led by Inbal Arnon for comments on earlier versions of these analyses. Thanks to Ximena Gutierrez-Vasques, Julia Łukasiewicz-Pater, Olga Peroni, and Steven Moran for their collaboration on the TeDDi sample and other projects directly relevant to this topic. The help of Clara Garcia Baumgärtner and Tim Wientzek with collecting non-linguistic data is also gratefully acknowledged. Finally, thanks to the anonymous reviewers who have helped to improve the manuscript.

## References


Julio G Arriaga, Martin L Cody, Edgar E Vallejo, and Charles E Taylor. 2015. Bird-db: A database for annotated bird song sequences. *Ecological Informatics*, 27:21–25.

R. Harald Baayen. 2013. Multivariate statistics. In Robert J. Podesva and Devyani Sharma, editors, *Research methods in linguistics*, pages 337–372. Cambridge University Press, Cambridge.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems*, 13.

Christian Bentz, Dimitrios Alikaniotis, Michael Cysouw, and Ramon Ferrer-i Cancho. 2017. The entropy of words – learnability and expressivity across more than 1000 languages. *Entropy*, 19(6):275.

Christian Bentz and Ramon Ferrer-i-Cancho. 2016. Zipf's law of abbreviation as a language universal. In *Proceedings of the Leiden Workshop on Capturing Phylogenetic Algorithms for Linguistics*. University of Tübingen.

Terra Blevins and Luke Zettlemoyer. 2022. Language contamination helps explains the cross-lingual capabilities of English pretrained models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3563–3574, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20:273–297.

Thomas M Cover and Joy A Thomas. 2006. *Elements of information theory*. John Wiley & Sons, New Jersey.

Łukasz Dębowski. 2020. *Information theory meets power laws: Stochastic processes and language models*. John Wiley & Sons.

Ramon Ferrer-i-Cancho. 2005. The variation of Zipf's law in human language. *The European Physical Journal B*, 44:249–257.

Ramon Ferrer-i-Cancho. 2007. On the universality of Zipf's law for word frequencies. In Peter Grzybek and Reinhard Köhler, editors, *Exact methods in the study of language and text*, pages 131–140. Mouton de Gruyter, Berlin & New York.

Evelyn Fix and Joseph L Hodges. 1952. Discriminatory analysis - nonparametric discrimination: Small sample performance. Technical report, California Univ Berkeley.

Iván G. Torre, Łukasz Dębowski, and Antoni Hernández-Fernández. 2021. Can Menzerath's law be a criterion of complexity in communication? *Plos one*, 16(8):e0256133.

Yun Gao, Ioannis Kontoyiannis, and Elie Bienenstock. 2008. Estimating the entropy of binary time series: Methodology, some theory and a simulation study. *Entropy*, 10(2):71–99.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Arik Kershenbaum, Daniel T Blumstein, Marie A Roch, Çağlar Akçay, Gregory Backus, Mark A Bee, Kirsten Bohn, Yan Cao, Gerald Carter, Cristiane Cäsar, et al. 2016. Acoustic sequences in non-human animals: a tutorial review and prospectus. *Biological Reviews*, 91(1):13–52.

Ori Lavi-Rotbain and Inbal Arnon. 2021. Visual statistical learning is facilitated in Zipfian distributions. *Cognition*, 206:104492.

Ori Lavi-Rotbain and Inbal Arnon. 2022. The learnability consequences of zipfian distributions in language. *Cognition*, 223:105038.

Ori Lavi-Rotbain and Inbal Arnon. 2023. Zipfian distributions in child-directed speech. *Open Mind*, 7:1–30.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

Rob Lee, Philip Jonathan, and Pauline Ziman. 2010. Pictish symbols revealed as a written language through application of Shannon entropy. *Proceedings of the royal society a: Mathematical, physical and engineering sciences*, 466(2121):2545–2560.

Wentian Li. 1992. Random texts exhibit Zipf's-law-like word frequency distribution. *IEEE Transactions on information theory*, 38(6):1842–1845.

George A Miller. 1957. Some effects of intermittent silence. *The American journal of psychology*, 70(2):311–314.

Steven Moran, Christian Bentz, Ximena Gutierrez-Vasques, Olga Pelloni, and Tanja Samardzic. 2022. TeDDi sample: Text data diversity sample for language comparison and multilingual nlp. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 1150–1158.

Sonia Petrini, Antoni Casas-i-Muñoz, Jordi Cluet-i-Martinell, Mengxue Wang, Christian Bentz, and Ramon Ferrer-i-Cancho. 2023. Direct and indirect evidence of compression of word lengths. zipf' s law of abbreviation revisited. *Glottometrics*, 54:58–87.

Rajesh P. N. Rao, Nisha Yadav, Mayank N. Vahia, Hrishikesh Joglekar, R. Adhikari, and Iravatham Mahadevan. 2009. Entropic evidence for linguistic structure in the Indus script. *Science*, 324(29):1165.

Rajesh PN Rao, Nisha Yadav, Mayank N Vahia, Hrishikesh Joglekar, Ronojoy Adhikari, and Iravatham Mahadevan. 2010. Entropy, the Indus script, and language: A reply to R. Sproat. *Computational Linguistics*, 36(4):795–805.

Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE international conference on neural networks*, pages 586–591. IEEE.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Amir Shufaniya and Inbal Arnon. 2022. A cognitive bias for zipfian distributions? uniform distributions become more skewed via cultural transmission. *Journal of Language Evolution*, 7(1):59–80.

Richard Sproat. 2014. A statistical comparison of written language and nonlinguistic symbol systems. *Language*, 90(2):457–481.

Richard William Sproat. 2023. *Symbols: An Evolutionary History from the Stone Age to the Future*. Springer, Cham, Switzerland.

Ryuji Suzuki, John R. Buck, and Peter L. Tyack. 2005. The use of Zipf's law in animal communication analysis. *Animal Behaviour*, 69(1):F9–F17.

Ryosuke Takahira, Kumiko Tanaka-Ishii, and Łukasz Dębowski. 2016. Entropy rate estimates for natural language – a new extrapolation of compressed large-scale corpora. *Entropy*, 18(10):364.

Kumiko Tanaka-Ishii. 2021. *Statistical Universals of Language: Mathematical Chance vs. Human Choice*. Springer Nature.

Lucie Wolters, Ori Lavi-Rotbain, and Inbal Arnon. 2023. Zipfian distributions facilitate learning novel word-referent mappings. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 45.

Christopher Woods. 2010. The earliest Mesopotamian writing. In *Visible language: Inventions of writing in the ancient Middle East and beyond*, pages 33–50. The Oriental Institute of the University of Chicago, Chicago.

George Kingsley Zipf. 1936. *The psycho-biology of language. An introduction to dynamic philology*. George Routledge & Sons, Oxon, UK.

## Limitations

See separate pdf file (part of the Appendices).

## Appendices

See separate pdf file for Appendices.