2021

Properties of Text

ZIPF-ANALYSIS ON YORUBA & IGBO

ADEWUMI, IMMANUEL CID: 01071093

SUPERVISOR: DR TIM EVANS

WORD-COUNT: 5981

Contents

Abstract

The Zipf's Phenomenon expresses an inverse relationship between the frequency of occurrence of an observed variable in a specific study and the ranked (positional value) of that variable. In this report, the focus will be on the study of Zipf's Phenomenon in Natural Language. The results and findings of Zipfian analysis carried out on bodies of text in languages which previously may not have been analysed for Zipf's Phenomenon will also be presented.

1. INTRODUCTION

Zipf Law is a curious phenomenon which has been the centre of studies by social scientists for the past 70 years **(Zipf, 1932)**. Also known as the Pareto-Principle **(Vilfredo Pareto, 1898*)*** in business fields, it is a subset of a more general group of Power-law probability distributions.

1.1 - Theoretically Background - Power laws

Power laws represent a body of functions in which two or more quantities are related by a functional relationship where one quantity varies as a power of the other**.** The simplest example of this is the relation between the side-length *L* of a cube and its volume *V*, with *V* varying as L^3.

In Statistics Power-law probability distributions are also known as long-tailed or heavy-tailed: used interchangeably **(Sergey & Dmitry, 1970)**. Their characteristic properties are that for increasing values of observed variable x, the cumulative probability density (or mass) function does not converge. Thus, they are generally *scale-invariant* **(Mandelbrot, 1967)**. Scale-invariance implies that for all values of x (infinitely large or infinitesimally small) the general power-relation holds. In practice, this means that no matter how much one zooms in, or out, of a heavy-tailed probability distribution plot, it retains the same general long-tailed form.

The most b*asic* Power-law function is 1/x which; for integer values x, becomes the Harmonic series. 1/x is also the form in which G.K. Zipf **(1935)** first introduced the Principle of Zipf's Law .

Cumulative power-law probability distribution functions are generally of the form:

$$P(X > x) \sim L(x).x^{\beta} \qquad (1)$$

where ***β*** has a value greater than 1 and function *L(x)* is defined as a slowly varying function **(Bingham, 1994)**. This is any function that obeys the criterion *L(kx)* / L(x) = 1 as *x* → infinity, for all positive valued ***k.*** *L(x)* is required to be slowly varying so as to satisfy the scale-invariance property for long-tail distributions. In probability terms this means it ensures a relatively-constant probability function *p(x)* for all measurable values of *x*. ***β*** must have a value greater than 1 in order for the function to converge and be normalizable for all *x*-values. Thus, accumulating to the total probability of 1.

It is usually convenient to assume that there is an x_{min} cutoff below which the law does not apply, this is because in theory, many power-law distributions with an inverse relation in *x*, diverge as *x* → zero. An infinite-valued probability has no physical meaning (as measurement values are finite). The probability *p(x)* of *x* occurring can be written as the derivative of the cumulative distribution of *x*. Thus, taking the derivative of cumulative form eq (1), whilst accounting for x_{min} (with *x* → $x/xmin$) it can be shown that the power-law probability function has a general form;

$$p(x) = \frac{\alpha-1}{xmin}\left(\frac{x}{xmin}\right)^{-\alpha} \qquad (2)$$

Here G(x) = α – 1 / x_{min} is the normalizing component and would be constant for completely scale-invariant Power-law distributions. From Eq (2) p(x) can be fitted into the known general equation for calculating statistical moments: these are parameter estimates which describe key characteristics, such as mean, variance and skew of a fitted plot. The results can be seen below Eq (3).

$$\langle x^m \rangle = \int_{x_{min}}^{\infty} x^m.p(x).dx = \frac{\alpha-1}{\alpha-1-m} x_{min}^m \qquad (3)$$

From Eq (3) we can see that the nth statistical moment is only defined when ***m*** is less than ***β* - 1,** as the function for the moment diverges for larger moments (values of ***m***). This means that if **2 < *β* < 3**, and we define a variable ***α* = *β* - 1** then all statistical-moments of value ***m* > 1** will be undefined and diverge, as ***m*** would be greater than ***β* - 1.** Power-law distributions which satisfy this unique criterion: **2 < *β* < 3** and consequently; **1 < *α* < 2,** are classified as ***Zipfian*** distributions.

1.2 - *Zipf's Law - Historical Background*

Zipf's Law was first extensively popularised by American Linguist - George Kingsley Zipf in his 1935 publication *'The Psycho-biology of Language'*. Though the law was not originally discovered him, he undoubtedly was the first to carry out extensive and rigorous studies on it, uncovering its prevalence in language and other seemingly unrelated social-sciences. In his latter publications, clear references were made to Vilfredo Pareto **(1898)**, Alfred Lotka **(1926)** and Jean-Baptiste Estoup **(M. Petruszewycz, 1973)** who had previously identified different variants of the law.

Zipf first encountered the law in the context of linguistics and is described by Glottometrics ***(a scientific journal on language and text - 2002)*** as the founder of quantitative linguistics (known also as 'Zipfian' linguistics). He is seen as visionary for developing the extensive interdisciplinary body of work which would later serve as the reference point for much further study and development; in fields such as sociology, psychology, geography, economy, business and more generally data-science (info-metrics - information analysis).

In 1949, a year before his untimely demise, Zipf released his most referenced book *'Human Behaviour and the Principle of Least Effort'* **(1949)**, which has been particularly popular amongst psychologists and social scientists. It was a culmination of his lifelong work using Zipfian statistics. In it he attempted to explain the reason behind the occurrence of Zipfian-type distributions in Language Evolution, Social Science, and Nature at large; explaining these as a result of what he termed: energy (or in linguistics - effort) minimisation.

1.3 - Zipf's Law in Natural Language

1.3a - The Law

Zipf's Law in linguistics (sometimes known as Zipf's general law) is of the form;

$$F(r,\alpha,N) = \frac{1}{r^{\alpha}}\left(\frac{1}{H_{N,\alpha}}\right) \qquad (4)$$

where F(r) is the frequency of occurrence of the rth most frequent word, $\mathbf{H}_{N}, \boldsymbol{\alpha}$ is the Nth generalised harmonic $\sum_{\mathbf{n=1}}^{N} \mathbf{1/n^{\alpha}}$, (where *N* is the total number of unique words in the source body of text) and *r* is the rank; (the hierarchical position of each of these N unique words in terms of their frequency magnitude). In equation (4) $\boldsymbol{\alpha}$ **= beta - 1** as stated in section 1.1 equation (3). Thus $\boldsymbol{\alpha}$ is a parameter of its own which, in practice, is determined by the source data being analysed. It must satisfy the conditions $\mathbf{1 < \alpha < 2}$**,** for a truly Zipfian fit. The $\mathbf{1/(H_{N}, \alpha)}$ coefficient of the $\boldsymbol{1/r^{\alpha}}$ term is often represented as a constant and can be absorbed into the $\mathbf{L(r)/r_{min}^{n+1}}$ coefficient of $1/r^{\alpha}$ in equation (3), since it satisfies the condition of a slowly varying function. Thus, Zipf's generalised law; equation (4), states that for a large enough body of text, the most frequent word occurs as a fixed number (usually approximately equals the reciprocal of the Nth Generalised harmonic), the second most frequent word, *r*=2, occurs roughly half as frequently as the first, the 3rd, r=3, roughly a third as frequently as the first, and so on. Taking the log of both sides yields a straight-line equation of the general form;

$$log\,(F) = -\alpha \log(r) + C \qquad (5)$$

1.3b - Zipf's Law Origins & Critical Analysis

In Zipf's original formulation of the law (1932) he analysed both Latin from Plautus' works and fragments of Chinese obtained from 20 unique sources. As early as 1932 he had demonstrated the existence of Zipf's phenomenon using these 2 languages. In his 1932 book, he made use only of Lotka's inverse-square law principle which can be shown to be mathematically equivalent to Zipf's Law with $\boldsymbol{\alpha}$ **= 1 (Glottometrics, 2002)**. He did this without reference to Lotka. His 1935 publication of *'The Psycho-biology of Language'* ushered in his use of the log frequency versus log-rank method shown in equation (5) as a means of ascertaining further information on the true data-based value of $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is not hypothesized to have a value of 1.

Although Zipf claimed, using the $\boldsymbol{\alpha}$ **=1** assumption, that the data (Latin & Chinese) followed this hypothesis when observed by eye, Rousseau and Egghe **(1990)** later showed, using more rigorous *goodness-of-fit* statistical methods that the assumption was not purely statistically correct. However, the more generalised form seen in equations (4) and (5), with $\boldsymbol{\alpha}$ defined as a parameter to be determined by the observed data, did hold.

Since its discovery, Zipf's Law has been reviewed, adapted and tested against many more languages and it has proved consistent across an overwhelming number of largely disparate ones.

Most worthy of note, is the fact that despite the prevalence of Zipf's law in most known languages, and it's existence as originally identified by Zipf in written word-segmented Chinese, the law has been shown not to hold in written *non-word-segmented* variants of Chinese, Korean and Japanese **[p] (from Yoruba paper)**.

Thus, the reasonable ensuing question would be whether Zipf's phenomenon is a fundamental result of how we as humans carry out processes of classification and segmentation. Mandelbrot first proposed a purely stochastic model which demonstrated this may very well be the case.

1.4 - Zipf Law – Mandelbrot Generalization & Further Reviews

1.4a - Mandelbrot's Proof & Generalisation

As justification for Zipf's law's prevalence Zipf proposed a law he termed: 'The Principle of Least Effort'. This principle was heavily based on the nuances of language evolution, such as its history, word-contexts and ease of communication. In stark contrast, Benoit Mandelbrot **(1954)**, known for his invention of fractal theory, proposed a purely statistical approach, in which he used ideas from information theory (Claude Shannon, **1948)** to explain the occurrence of Zipf's Law. The idea behind Mandelbrot's approach which used a few ideas from Zipf's original postulates was that words have an information/cost ratio (that is a metric for the effectiveness of their use in communication). This ratio is determined by the number of letters in the words and the spaces between them. The cost was said to increase with increasing number of letters. By minimising the information/cost ratio. Benoit Mandelbrot showed that his first statistical approximation produced Zipf's law: Equation (4). It is quoted by **Rousseau (2000)** that the **alpha** exponent in Equation (4) turned out to be the Fractal Dimension of Language; a property which since has been proven by various means to be the likely reason for the shared attribute of Scale-invariance present in both Mandelbrot's fractal geometry and Zipf distributions **[p, q, r]**. Mandelbrot's second approximation for information/cost minimisation is even more accurate and is often known as the generalised Zipf-Mandelbrot equation. It takes the form:

$$log\ (F) = -\alpha \log(r + b) + C. \qquad (6)$$

This form will be examined in further detail in later sections of this paper.

1.4b - Reviews

More recent analytical reviews of the law and Zipf's proposed justifications have revealed that perhaps Zipf's law may indeed be the result of fundamentally statistical processes governing language formation. Bruce M. Hill **(1970)** and Michael Woodroofe **(1975)** both derived a statistical form of Zipf's law by utilising Bose-Einstein and Maxwell-Boltzmann statistics. Yuji Ijiri and H.A. Simon **(1975)** offered similar derivations in their publication "Some Distributions Associated with Bose-Einstein Statistics."

A more recent in-depth analysis by Steven T. Piantadosi **(2015)** brought a new insightful perspective, shifting focus from the approach of merely deriving the law by a set of postulated statistical assumptions, to more rigorous analyses (using modern language analysis techniques) of Zipf's proposed explanations. Other than the deeper insight provided by such a detailed and scientifically rigorous case-by-case study, Piantadosi's second reason for moving away from the previous derivation approaches, was based on the fact that the law could be derived from a large number of largely disparate starting-points (as stated above). Thus, the ability of any one-single proposed theory to explain the occurrence of Zipf's law provided at best, very weak statistical evidence for the validity of that proposed theory.
Piantadosi also talked extensively on fundamental issues he termed 'statistical sins' in Zipf's handling of errors, and the failure pervasive in the Zipfian literature study, to rightly address the issue of correlated errors. Although these do not affect Zipf's overall conclusion, Piantadosi argues they contribute to falsely representing frequency-rank correlations as much simpler than they truly are. His $\mathbf{R}^2$ residuals plot does indeed confirm that there are much more intricate patterns at play in natural language which can only be loosely approximated by Zipf's law.

Although many power-law variants have been proven to be statistically compatible with language. Zipfian analysis, however, will be the sole focus of this paper.

1.5 - Project Focus & Scope

The complexity of identifying the specific causes of Zipf's law in nature and Natural Language meant that within the 3-month scope of our project it would be unrealistic to attempt tackling any problem on that scale of magnitude. Thus, we chose a more specific and quantifiable problem as our subject of focus.

We decided on looking to carry out analysis on a language which had not previously been analysed against Zipf's law. We realised, the most likely languages fitting this criterion would be Nigerian languages which only recently have had their online text corpora grow to sizes which could yield statistically reliable results when analysed. The languages chosen were Yorùbá and Ìgbò for their prevalence as 2 of the 3 most commonly spoken languages in Nigeria, the 3rd being Hausa, which we also considered studying, but were unable to do extensive research in, due to our time constraints.

Moreover, we decided starting with Zipf analysis on Bible translations across these 3 languages; comparing results of Yorùbá and Ìgbò against that of English. We expected that the uniformity of language-context, meaning and structure, which is a feature generally known to hold across bible-translations, would provide us with even more understanding of where language-specific differences occur. Thus, we hypothesized that all 3 translations should yield similar, if not identical, Zipfian curve-fits, within estimated boundaries of uncertainty; differing only on language intrinsic differences and specific instances of creative word-choice by the translators. In this way, we hoped to account for any words which did not have a direct one-to-one word mapping. Our assumption was that it should be possible to account for the overall effect of these differences with more detailed analysis of the text. Analysis including comparisons of the literal versus contextual word meanings, which were carried out using tools such as N-grams (described later), Google-translate, total word-count and vocabulary size as well as assistance of a few native language speakers.

Structure of the Languages & History

African language experts place the Two languages within the **Niger-Congo** group of african language roots which are heavily tonal. This tonality is depicted by markings, known as diacritics, on the written text of both languages. Depending on the diacritics present, words with similar *'Latin-alphabet'* make-up can have largely differing meanings. For example; Ogùn: (Medicine), Ògùn: (Charm), Ogun: (War), Ògùn: (A State), Ògún: (God of iron) all have different meanings.

2. METHODS

2.1 - Key Concepts

Here we introduce the background for the concepts and tools used in this project.

Natural Language Processing; abbreviated NLP, is a mixed-field of artificial intelligence, computer science and linguistics which involves 'training' computers to read, analyse and interpret human language inferences, such contextual nuance or sentiment, on large bodies of text. The NLP concepts used in this report, along with their definitions, are listed below.
Corpus: a corpus (plural: corpora) is likely the most frequent NLP term and represents a large body of text formed either Naturally through regular means of communication and information dissemination, or compiled artificially, for the specific purpose of data-analysis.
Token: a token is a segmented piece of data from a larger Corpus, usually representing a word, but not restricted to words. It can also represent linguistic items such as punctuations or mathematical symbols.
Type: a type in NLP refers to each single-unique element (in some instances a word) found in a larger corpus, regardless of its frequency of occurrence in the larger body of text . In linguistics, the total set of word-types in a Corpus represents its complete vocabulary.
Normalisation: In NLP normalisation represents the lowering of capital letters in words like Nouns, the pronoun 'I', or any words that begin a sentence. The purpose of this is to permit the encoding system to read them as the same as their identical counterparts with lower-case letters. This is necessary since computer Encoding counts capital and lower-case letters as different characters, and by consequence, the two forms of the same word as different tokens.
Tokenization: The process (usually computational) of breaking a larger corpus down to its separate tokens.
N-grams: These are common word-groups which show up frequently in the text. Grouping all adjacent words within a corpus as tokens and performing frequency counts on these token-groups is a method used in NLP to identify frequently occurring word-groups. The N in N-gram represents the number of words being grouped together. A uni-gram represents a single word token, a di-gram a word group of 2 and tri-grams word-groups of 3.
Bag-of-words: This is an NLP technique for presenting the frequency of occurrence of words (or tokens) within a corpus. It utilises two major elements: The vocabulary of known word-types and the frequency measure of each of those types.

The term “bag” of words, stems from the fact that all information about the order and structure of words in their original text source is completely discarded.
Bootstrap: This is a statistical technique for estimating the properties (moments **[p**) such as variance or bias of an estimator (e.g. a samples mean) when dealing with static (non-variable data). Static data is simply data that stays the same after measurement and thus cannot yield a measure of variance with repeat measurements. The bootstrap method works using random **sampling with replacement**, that is; by selecting random samples from the original data-source to place into the new sample and each-time placing them back in the source before the next random selection; thus keeping the probability of selecting each variable from the source constant throughout the process. A sample is complete and is called a Bootstrap when it has equal sample-size as the original source. In the Bootstrap technique multiple bootstraps are taken and variance is estimated using the estimators calculated from each bootstrap.

2.2 - Tools Used

Python: For computational tasks, we used the Python programming language within Spyder, a compatible IDE (integrated development environment) run on the Anaconda platform **[p]**. These were chosen for their convenience and plethora of integrated data-science packages.
NLTK: NLTK, long-name: 'Natural Language Toolkit', is a world-standard suite of Python libraries used by professional data-analysts, text-based machine-learning experts and in the linguistics department of world-class universities. Developed originally by Steven Bird and Edward Loper **(2016)**, its core use is for Symbolic and statistical NLP. It is openware accompanied by a step-by-step online book-guide to learning NLP. It also contains extensive Corpus libraries including a large portion of 'The Gutenberg Project'.
Its accessibility and plethora of integrated modules were two of the many major reasons it was selected as an obvious addition to our NLP toolkit.
Web Scrapers: These are tools used for extracting data from web pages. Web scrapers access data either through Hypertext Transfer Protocol (http) or through a web browser (as an extension).
Pandas: An opensource library which provides high-performance, easy-to-use data structures and analysis tools for the Python programming language. We chose to use this library due to its ability to produce DataFrames (easily manipulated data tables) from regular python generated lists (or bags-of-words). These data-frames within Pandas could with ease perform functions pertinent to our task of Zipfian analysis, such as; counting word frequencies, creating quick frequency-rank columns and combining bags of words and Dataframes using similar words to make quick and easy comparisons.

2.3 - Approach

This segment details the step-by-step approach used in tackling the problem laid out in the introduction section.

2.3a - Extraction, Collation & Cleaning of Corpora

Though all of the data analysed was found in large quantities online it was not exactly trivial to extract and collate the necessary corpora. Web scrapers were needed to automate the process and extensive research was done to determine which were best suited. I initially assumed that a written-python code might be the most efficient tool, but after downloading and trialing 2 codes; with one failing, and the second working incredibly slowly, I concluded that due to time-constraints it would not be the appropriate tool for the task.

I also came across the WebScraper from apify.com which I could not figure out how to use on the Youversion bible website, but was the perfect tool for scraping BBC news sites. It scraped up to 1000 articles in 40 minutes, in stark contrast with the original Python-Scraping code which did only about a 100 per hour and never successfully ran to completion.

The next tool considered was a Chrome extension web-scraper **[**webscraper.io**]** which was useful for its capacity to follow a set logic and interact with the web-pages' toggle buttons, whilst scraping the pages. It was the most useful tool for scraping the bible translations off of the bible website Youversion. No other scraper worked as efficiently for this purpose.

All web scrapers saved text automatically in csv file format. These texts were easily copied to raw text files and saved, for ease of reader convenience and also easier accessibility by our python programme when converting the text-files into bags-of-words.
The entire process of collating corpora; by extraction and preliminary formatting took 2 man-weeks.

2.2b - Producing Primitive code

The primitive code for analysis was produced by my project partner. This was our prototype used for testing proof of concept. Using the code an initial Zipfian test was run on a text-file of *'A Christmas Carol'* by Charles Dickens; downloaded from 'The Gutenberg Project' online **[p].** The code was found to work well, producing the expected negatively-sloped, roughly straight-line graph.

Code-Logic

- **Opening text file:** The code began with a first function which opened and read the file using Python inbuilt 'open()' function and 'read()' method. This transformed it from a raw text file into a long python 'string'.
- **Cleaning text:** Next the text was cleaned of all punctuations, using Python's translate function; maketrans() which took, as an argument, a list of characters to be replaced; `chars_to_remove = "!\"#$%&'()*+,-./:;<=>?@[\]^_`{|}~0123456789"`, and replaced all instances of these within the text with instances of whitespace.
- **Normalisation of text:** The cleaned-out text was then normalised using Python's inbuilt normalisation method '*lower()*'. This means all occurrences of capital letters in words were replaced with they're lower case equivalent.
- **Tokenization:** The new string of cleaned-out and normalised text was then tokenized using the Python in-built *'split()'* method which split the text at every instance of whitespace.
- **Frequency Counts:** Each word-frequency was then counted using the *'counter()'* method from an imported module named *'Collections'*. The data was then restricted to most common *n* words using the *'most_common(n)'* method for the *'Collections'* model.
- **Tabulation and Plots:** The data was tabulated using a function defined by my partner, which contained a column for the words, their measured frequency, their rank, their expected Zipf frequency using the basic ***1/r*** Zipf inverse proportionality and the percentage difference between the measured frequency and anticipated 'ideal'. A plot of the measured frequency against word-rank was then done; both on a normal scale and on a log(Frequency) vs log(Rank) plot. The normal scale plot showed a long-tail distribution whilst the log-log plot demonstrated and approximately straight line.

Weaknesses of Code

- The code did not have a function for counting and showing the total vocabulary-size (number of word-types) and total word-length (number of tokens) within the text document corpus.
- The number of words to be tabulated, analysed and plotted had to be manually input which could serve as a problem since we had no prior knowledge of the vocabulary size (number of types) of the corpus. This yielded an error whenever we accidentally input a number larger than the corpus-size.
- The corpus developed from the book, as anticipated, did not follow the most basic *1/r* Zipfian representation and so the code had to be altered to accommodate for the possibility of an *r* exponent parameter **alpha [equation (4)]**, whose value was to be determined by the corpus.
- The parameter **alpha** was to be estimated using the gradient parameter of the Log-log plot, but this could have only been done with a best-fitting algorithm. The primitive code, however, did not include a best-fitting algorithm.
- To produce a statistical measure for the validity of the hypothesized fit a measure of uncertainty was required. As the primitive code had no measure of uncertainty and so could not be scientifically tested.
- The primitive code had no considerations for data-point weighting since it had no best-fit function or measure of uncertainties. However, the inclusion of data-point uncertainty measures would mean that each data-point would need to be weighted according to the uncertainties each data-point contributes towards the overall fit; with data-points of higher uncertainty weighing less and vice-versa.
- There was no Goodness-of-fit method for testing the level of confidence in the validity of the fit.

2.2b - Code Developments

A new upgrade of the code with the same key primary features as the primitive code, but with newly added useful features was developed by myself.

Token counter: This was added to count the total word-length of the document corpus was added using Python's in-built *'len()'* function the corpus fed into it. In addition a Measure of the Vocabulary-Size of the corpus was also added using the 'len()' and 'set()' in-built functions with the corpus fed as the argument as shown: '*len(set(corpus))*'.
Frequency Cut-off: The new code was modified to have a cut-off for the number of words analysed based on measured *frequency of occurrence*, as opposed to the number-guessing done in the primitive code. For example; setting the cutt-off frequency to 10 would exclude from the Zipf-test all words within the corpus occurring just 10 times and below.
Pandas Library: The Pandas library was utilised to convert the lists into Dataframes for more efficient and flexible management.
Best-Fitting: It was implemented within the code using the Linear-regression fit **[p]** function: polyfit from matplotlib.pyplot and nonlinear regression fit function curve_fit from scipy.optimize for non-linear fitting.
Bootstrap Uncertainties: As explained earlier **in section 2.1** the Bootstrap technique was used for calculating an unbiased estimate of the variance of both the data-points, and the measured parameters for gradient and the Mandelbrot correction *b*. Thus there were 2 methods for showing uncertainties used.

Weighting: Using the natural log of the uncertainty bars created using the Bootstrap method each data-points weighting was calculated using an inverse uncertainty to weighting relation.
KS-Test: For testing the goodness of Zipfian of fit against the data points, the Kolmogorov-Smirnov Hypothesis testing method was used. **This** technique measures a *Distance* **[p]** between the Hypothesised distribution and measured sample distribution as a means of determining the goodness-of-fit. The test works by producing a ***p-value*** probability of identical fit. For p-values (probabilities) less than 0.05 (5%) the proposed (null) hypothesis can be rejected.

2.2c - Reasons and Logic

Weighting: Before applying the appropriate data-points weighting, I observed that the fit was skewed towards lower frequency numbers and thus, by eye, it did not seem to be a very good fit for Zipf's law. On looking through the plotted Zipf table and at the normal scale Zipf plot; it became apparent that this happened because the ratio of data-points in the plot was heavily skewed towards words with lower frequency; meaning that there was a much larger number of unique low-frequency words (or types) than those with higher frequency. For example; the word 'the' which is highest ranked-1 in English and accounts for only about 7% of all word-usage. Since the low-frequency words had much lower frequencies of occurrence the bootstrap method naturally produced higher uncertainties for them, with some having percentage uncertainties as high as 100%. Since weightings and uncertainty are inversely proportional, this, in-turn, meant they had much lower weightings; which balanced out the skew, resulting in a much better fit.
Bootstraps: The Bootstrap samples were initially created using a for-loop running through the entire size of the corpus. The for loop at each iteration generated a random index using the Python function randint(), which in turn selected a word from within the corpus that had the randomly chosen index. In this way a sample corpus or Bootstrap with length equal to the original corpus was created. The process was repeated multiple times to create multiple bootstraps for each Zipfian analysis. The number of Bootstraps shown alongside each result.
KS-Test: For implementation of the KS-test I used a function imported from scipy.stats named kstest. When the test was done using the entire vocab of the corpus the ks-test yielded p-values of magnitudes about **10^-50**, which were much less than the **0.05 acceptance** value, signifying a statistical rejection of the initial (null) hypothesis of a Zipfian fit. This could be accounted for by the high uncertainty of the lower-ranked words, which as stated earlier had a higher number of unique data-points.
By setting a cut-off frequency of **56** for the Igbo Bible, with 10 Bootstraps and running the code 5 times, an average p-value of 0.0598 was gotten. This was reasonably over the 0.05 rejection threshold; signifying insufficient evidence to reject the null-hypothesis of the general Zipf Law equation (4). Repeating the process with the Yoruba Bible and using a cutoff frequency of **70** yielded an average p-value of 0.0505 which was also above the rejection threshold. The cutt-off frequency for the English King-James bible which provided us with a ***p-value*** of **0.051 > 0.05** was **109**. For all 3 cut-off frequencies rough repeat p-value tests were done for the cut-off frequencies around them to ensure that these yielded similar values. This was indeed the case.

2.2b - Further Code Developments

The initial further developments were motivated by my partner, to improve the efficiency and functionality of the previously developed code. We then both worked together on all final additions and modifications.

Discovered Issues

Bootstraps: My project partner rightly identified an error in our original use of bootstraps to estimate the uncertainty of each of our data-points. It went as follows; rather than identifying each unique word across all bootstraps, the positional rank was used, with the assumption that the frequency order of the words remains roughly constant across all bootstraps, for the higher frequency words. This in fact was not valid as bootstraps are a randomised process, and thus frequency order could change across bootstraps; meaning that certain calculated uncertainties would include values from other words' data-points within their own estimation, and some words may never even be recorded in some bootstraps, thus rendering the process invalid.

Runtime: We noted also, that when running the code with bootstraps significantly above 10, the code's runtime increased exponentially rather than linearly. We noted this must have been a result of one of the iteration methods used within the code and discovered that it indeed was caused by the for-loops.

Solutions

Bootstraps: To fix the initial issue of non word-accurate uncertainty estimations, my partner discovered that the merge function within the Pandas library, for merging data frames could be used. In our context it was used to merge all data frames 'on' the column 'word' against the original measured data as the reference. This identifies the unique words and keeps them in identical order as the original regardless of their individual frequency-rank distributions. The pandas merge function had a useful added feature of excluding any words which occurred zero times across any of the bootstraps, which we utilised, since zero-occurrence does not give a valid measure of variance.

Runtime: As a solution to the second issue of runtime, my partner proposed the idea of replacing all for-loops for generating bootstraps and other necessary lists with a much more efficient method utilising numpy arrays, to generate the bootstraps and other lists in one step rather than 1000s. This exponentially reduced our runtime, permitting us to run up to 10,000 bootstraps, despite previous difficulties.

Additions

Bootstrap Histograms: My partner added into the code a function for plotting the bootstrap-histograms for each word of the frequency-rank distribution. These were not shown in this report.

Residuals Plots: Finally, I added a plot of the **R2** Residuals of the linear regression fit of the data. This was done to test what is known as the *heteroscedasticity* **[p]** of the plot. The term simply means the measure of how constant a dependent variable's variability is across all values of the corresponding independent variable; and therefore tests whether a linear regression model is the appropriate fit for the data **[p]**.

Mandelbrot Correction: For increased accuracy I introduced the Mandelbrot correction *'b'* from equation (5) into the curve fit, using the curve_fit nonlinear regression fitting function for nonlinear fitting. The kstest was again used to statistically test the confidence level of a Mandelbrot model.

3. RESULTS

3.1 - Zipf's General Law

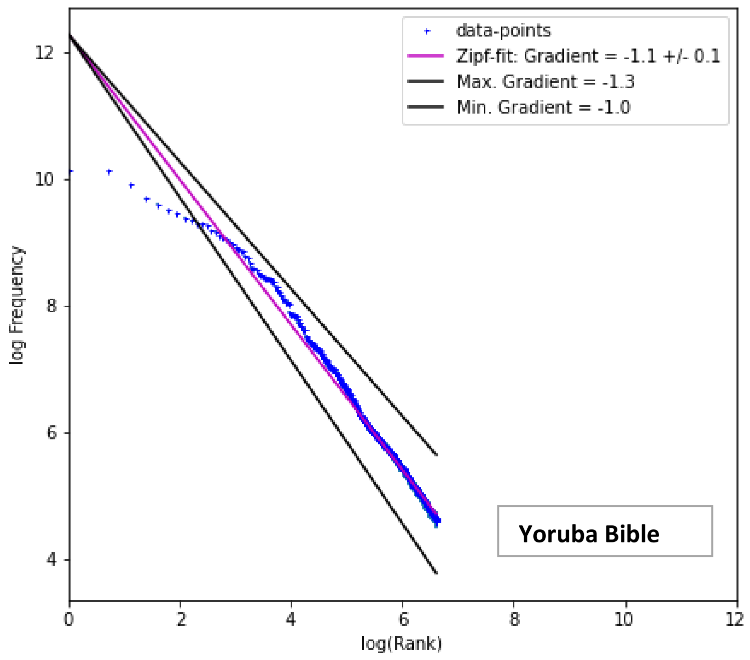


Fig.1 Shows the General Zipfian fit for the Yoruba-Bible**,** after a frequency cut-off of 70. The black lines represent the max. and min. possible gradients estimated using the bootstrap technique.

Yoruba Bible	Gradient	Intercept
Original Value	-1.1	12.3
Bootstrap Mean	-1.2	12.4
Standard-Dev.	+0.1	00.9
Standard-Dev.	+0.1	00.9

No. tokens	No. types	KStest (p-vaue)
819174	23912	0.034

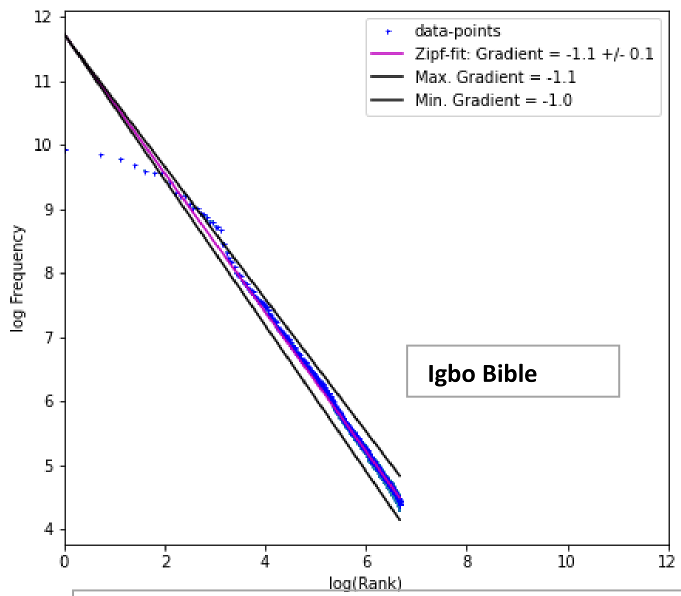
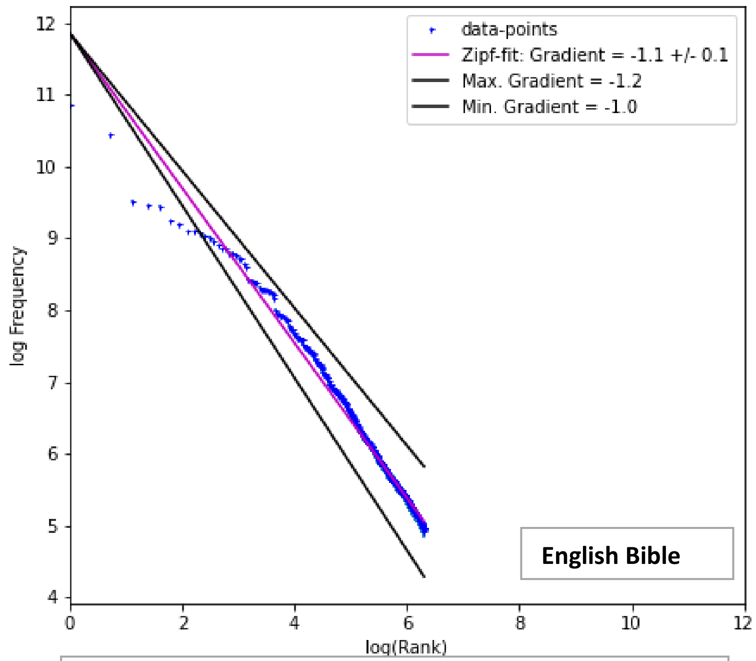


Fig.2 Shows the General Zipfian fit for the Igbo-Bible**,** after a frequency cut-off of 56. The black lines represent the max. and min. possible gradients estimated using the bootstrap technique.

Igbo Bible	Gradient	Intercept
Original Value	-1.1	11.7
Bootstrap Mean	-1.1	11.9
Standard-Dev.	+0.1	00.3

No. tokens	No. types	KStest (p-vaue)
714675	29989	0.213

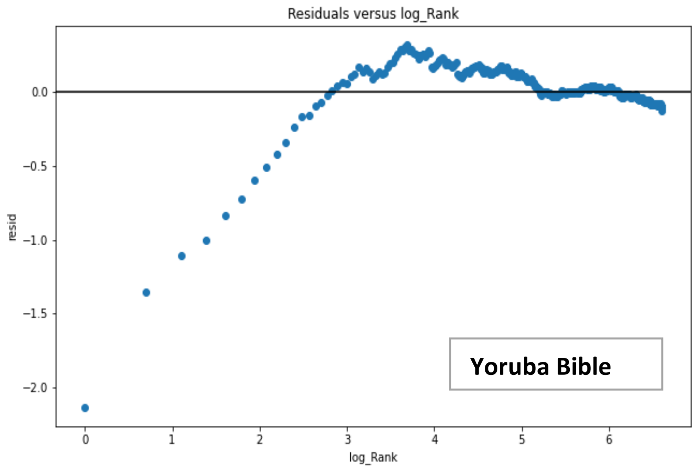


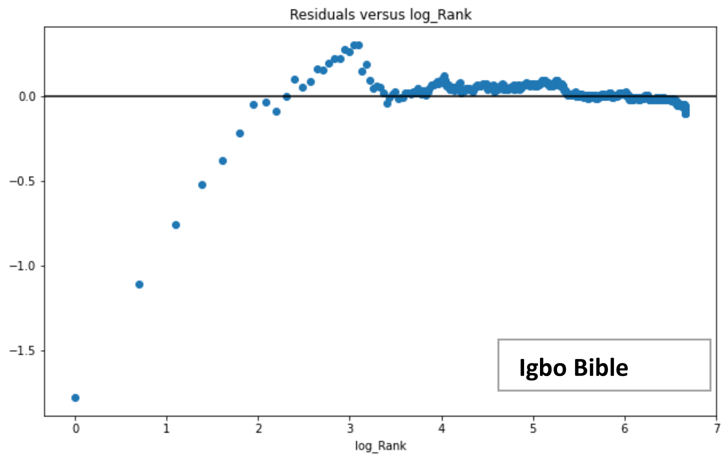
KJV Bible	Gradient	Intercept
Original Value	-1.1	11.9
Bootstrap Mean	-0.1	12.0
Standard-Dev.	+0.1	00.5

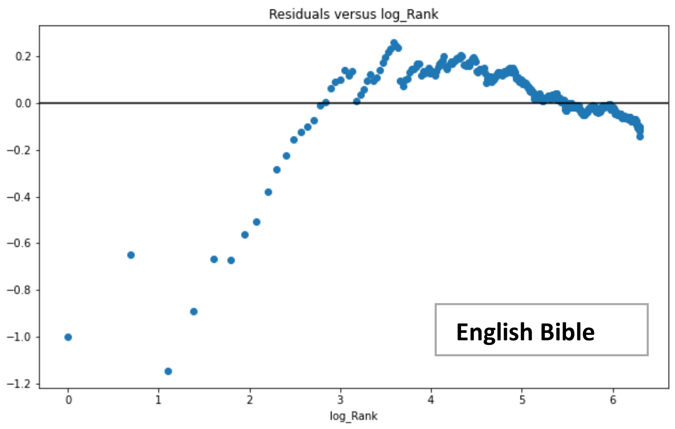
No. tokens	No. types	KStest (p-vaue)
793074	13199	0.11

Fig.3 Shows the General Zipfian fit for the KJV English-Bible**,** after a frequency cut-off of 70. The black lines represent the max. and min. possible gradients estimated using the bootstrap technique.

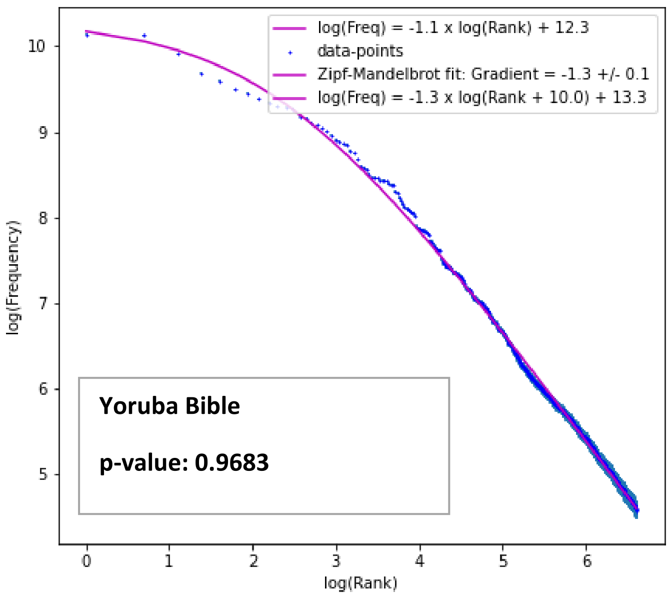
Residuals Plot

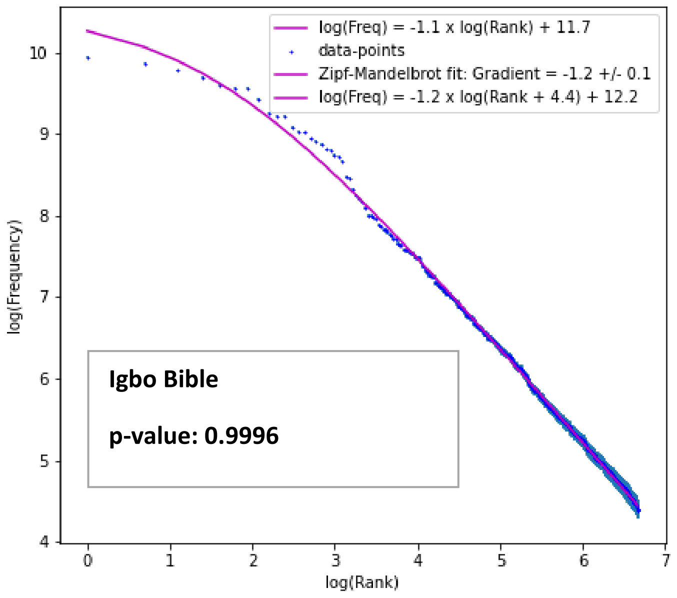


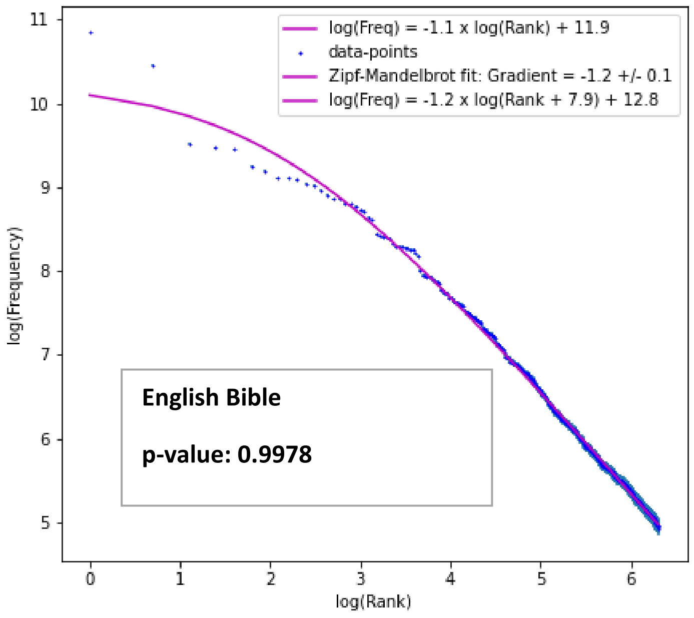




3.2 - Mandelbrot's Correction







4. DISCUSSION OF RESULTS

Using the cut-off frequencies determined in the 'Reasons and Logic' Section of the previous chapter each Corpus was tested, using 100 Bootstraps for uncertainty estimation. The results show that our null-hypothesis of identical Zipfian fit across the 3 bible language translations holds; within the limits of experimental uncertainty. This is because within these limits the Zipfian gradients are roughly equal.

As a test for whether the above Linear-regression model was a good model for the data we did a plot of the Residuals. A good linear model would have a Residuals scatter plot which has no distinct patterns, with points randomly distributed equally above and below the horizontal-axis. Distinct patterns however were observed, suggesting as **Piantadoosi** identified **(2015)** that language can only be loosely approximated by Zipf's general equation.

After the data was fit using the Mandelbrot correction, with p-values almost approaching, it showed it was an even stronger argument for the null-hypothesis that all 3 translations should have identical Zipfian-fit.

5. **CONCLUSION**

We proved that Zipf's law holds across all 3 language translations of the bible. Possible future directions of exploration include Developing Ngrams method of analysis and also extending the scope of our research e.g. looking at BBC translated articles.

References

Zipf, G.K. (1932). Selected studies of the principle of relative frequency in language.

Cambridge (MA): Harvard University Press.

Pareto, Vilfredo (1898). "Cours d'economie politique". *Journal of Political Economy*. **6**. doi:10.1086/250536.

Sergey & Dmitry, 1970 - https://www.researchgate.net/publication/226229954_Heavy-Tailed_and_Long-Tailed_Distributions

Mandelbrot, Benoit B. (5 May 1967). "How long is the coast of Britain? Statistical self-similarity and fractional dimension". Science. New Series. ***156*** *(3775): 636–638. Bibcode:1967Sci...156..636M. doi:10.1126/science.156.3775.636. PMID 17837158. S2CID 15662830.* PDF

Bingham, N.H. (2001) [1994], "Slowly varying function", Encyclopedia of Mathematics, EMS Press

Mandelbrot, B. (1954). Structure formelle des textes et communications: deux études. Word, 10, 1-27.Zipf, G. K. (1949). Human behavior and the principle of least effort. Cambridge (Mass.): Addison-Wesley Press.

M. Petruszewycz, 'L'histoire de la loi d'Estoup-Zipf: documents' Archived 2011-06-05 at the Wayback Machine, *Mathématiques et sciences humaines*, vol. 44 (1973), pp. 41-56.

Glottometrics-zipf.pdf

Rousseau, R. (2000). George Kingsley Zipf: life, ideas and recent developments of his theories. In: R &D Evaluation and Indicators (Jiang Guo-Hua, ed.), Red Flag Publishing House, p. 458-469.