Functional Approach to Library of Babel for Text Analysis

Dakota Sanders, *Graduate Student, University of Tennessee,* Zhixiu Lu, *Graduate Student, University of Tennessee,* Bradley White, *Undergraduate Student, University of Tennessee,* and Rachel Offutt, *Graduate Student, University of Tennessee*

***Abstract*—The Library of Babel is a permutation of every possible combination of characters as words, on every possible combination of pages, in every possible combination of 410 page books. Currently, any useful data is drowned out by nonsensical words acting as "noise." We are proposing potential objectives, depending on the feasibility of an initial mathematical investigation: either a functional implementation of the Library of Babel or a tool for effective and concise data parsing to make the current Library of Babel more navigable and usable.**

***Index Terms*—digital, archaeology, library, Babel, implementation, algorithms.**

I. Background

THE Library of Babel was first proposed as a thought experiment by Jorge Luis Borges in his short essay titled "La Bibilioteca Total" in 1939. The concept is quite simple: what if there existed a library, with a combination of every possible combination of letters, and every combination of all possible words, and thus containing all works of literature, past, present and future? In his essay, Borges muses that the library would contain detailed records of historical events passed, and even contain events of the future in perfect accuracy. Unfortunately, this library would also contain combinations of letters that are not recognized by any language as words, and thus the infinite knowledge contained in this theoretical library is perpetually drowned out by the noise of nonexistent words and nonsensical mismatches of words.

In 2015, Jonathan Basile was inspired by the works of Borges and committed to creating a digital version of the Library of Babel. In his implementation described by Borges, the Library is organized as the following:

> "The universe (which others call the Library) is composed of an indefinite, perhaps infinite number of hexagonal galleries... The arrangement of the galleries is always the same: Twenty bookshelves, five to each side, line four of the hexagon's six sides... each bookshelf holds thirty-two books identical in format; each book contains four hundred ten pages; each page, forty lines; each line, approximately eighty black letters."

Thus, we have the way to organize the data. For a naming convention in terms of finding a specific section of data, take for example: jeb0110jlb-w2-s4-v16. This value means the book you are reading is the 16th volume (v16) on the fourth shelf (s4) of the second wall (w2) of hexagon jeb0110jlb.

Special thanks to Preston Provins and David Kennard, the teaching assistants of this course.

Special thanks to Dr. Audris Mockus, instructor of this course.

Manuscript received December 9, 2019.

In the current version of the digital Library of Babel, the library claims to contain every possible page of 3200 characters, but not every possible book containing every possible permutation of these pages, due to size and time constraints.

II. Project Details

A. Objective and Motivation

After reading about such a lofty idea and viewing the results of the digital implementation of this concept, there was a question as to the value in such a data-set for the purpose of data science. Our initial proposal was to find a way to implement the Library of Babel, or a subset of it, in a way that would be easily accessible via some public API or other method of data gathering. Once this tool was usable, we planned to analyze the data-set to decide whether any valuable or interesting data would arise from it.

Given a theoretical data-set that is entirely random, the initial topic for analysis revolved around finding some indicator of structure within the randomness. We believed if we could find a level of structure in a given page of random text, then perhaps we could compare that value to other works of literature and see a level of structure in those works, which should be significantly higher. This analysis primarily serves as a thought experiment, and any deeper practical meaning of this analysis can be left to the reader.

B. Data Used for Analysis

Data will be scraped or modeled off of the digital archive of the Library of Babel, https://libraryofbabel.info.

We also used four works of literature to compare the natural language structure to the Library of Babel: Grimm's Fairy Tales, Pride and Prejudice, 500 Most Common Words in the English language, Copyright Law, and an Academic Journal Paper Collection (10 papers).

III. Mathematical Analysis

The current implementation of the Library of Babel does not actually store text files as it projects. The original version of the site did store text files on disk, however, the current version uses a hash function with the key of the hash function being the location of a page to pseudo randomly generate pages dynamically on request. The current library also contains all iterations of 3200 characters on a page and all assemblage of those said pages. Our first question is can we come up

with a workable subset of data that is not as massive at the current goal of the Library of Babel website. The library of Babel currently contains 1-3200 character words using the standard 26 letter English alphabet, al lower cased, including the following characters: a period, space, and comma. If we exclude the comma, we then have 28 characters to work with. The longest technical word in a major English dictionary is "Pneumonoultramicroscopicsilicovolcanoconiosis", sitting at 45 characters.[1] Thus, we do not have to try and generate any words greater than 45 characters. Therefore, we need to consider words comprised 1 to 28 unique characters of size 1 to 45 characters long. We also need to keep in mind that the order of the said letters is important, as the word "cat" is different than the word "tac", where the same letters are used but the words are different. Thus, for a given set of size n, there will be n^k possible strings of length k. The math for potential words then becomes:

$$numberofwordspossible = \sum_{k=1}^{45} 28^k = 1.37 * 10^{65}$$

Keeping with the Library of Babel's web implementation of pages containing 3200 characters, the number of pages possible are then $28^{3200} = 8.04 * 10^{4630}$ pages If we store every potential page, the storage on disk breakdown is as follows: Standard is 1 byte per character, so each page requires 3200 bytes. Therefore, for every potential page, $8.04 * 10^{4630} * 3200 = 2.57 * 10^{4634}$ bytes are needed. This becomes $2.57 * 10^{4622}$ terabytes required to store every page, and not even every grouping of 410 pages. For our implementation, we shall consider the 171,476 words in current use in Webster's Third New International Dictionary, in the current dictionary, plus the period and space, so 171,478 words total. The average length of English words is 4.7, so we rounded to 5. Thus, a page of 3200 character would contain roughly 640 words, so an estimate of the number of possible pages would be: $171,478!/(640!(171-478-640)!$ or 10^{1829} potential pages, which would require $(10^{1829} * 3200)/1 * 10^{12}$ terabytes.

Due to the sheer size of space required, we decided to no longer stay with out original project proposal of implementing a subset of the Library of Babel.

IV. Implementation and Limitations

A. Limitations

Several limitations existed on this current project. For one, our proposal had to be modified due to the sheer data size that would be needed to store the Library of Babel's pages and books. Even the reduced subset was on the order of 10^{12} terabytes. Due to this limitation, our original proposal of creating a usable subset of the English language is not possible given data constraints.

Another limitation on this project comes from the way the website of the Library of Babel is set up. In Python, to scrape a website, you use something called a GET() request, which retrieves the website of a specified URL. However, a problem arises when a long URL is entered, as there are size limitations to the max string length a GET() request can handle. Pages on the website have the hexagon, wall, shelf, book, volume, then page the entry belongs on in the URL, causing many of the URLs to exceed the length of URL that can be handled by the GET() request. With several of the web addresses being non-accessible by Python and only through the website, scraping the Library of Babel becomes quite impossible, unless you want to access these pages directly through a web browser.

These factors caused our team to shift our project from the original proposal to a different idea.

B. Implementation

Given the nature of such a large data-set, it became clear that data storage was entirely infeasible for a project of this scope. Instead, we decided that dynamically generating pages was a valid approach, and that most of our effort should be focused on performing analysis and deriving value from a random data-set. We believe the implementation was secondary to the analysis, as the value provided in such analysis could spur on more discussion and more research.

To achieve these goals, we found an open-source Python library implementing the dynamic generation of Library of Babel pages. Using this library allowed us to focus our efforts on providing meaningful analysis. Our analysis focused on natural language processing primarily, and as such any code written in our implementation strives to extract valuable data from the data-set rather than implement the Library of Babel.

V. Data Analysis

A. Natural Language Processing

The first step of processing the data involves setting a baseline for the English language. We set out to determine whether the Library of Babel conforms to any patterns found within more structured language. Once this baseline was set, we could then run the same tests on the Library of Babel and view our results. After normalizing our results, we used the findings of such research to implement a toy encryption protocol. While we recognize the use of such a protocol is unwise for a large number of security related reasons, we found value in providing an example to the reader that may inspire novel ideas that are more helpful to the general public.

B. Analysis on English Language

To set the English language baseline for the project, we found the most commonly used letters found within the top 10,000 most common words in the English language, which can be seen in figure 1. We base the following analysis on the principle that a document can be considered more complex if it contains a large number of less-common English words.

The frequency of use of a given letter in the English language based on the rarity of the word can be found in figure 2. Of the frequencies found in such analysis, two letters stand out based on their polarizing frequencies, namely e and h as shown in 3. The letter e is found to be positively correlated with more complex texts, and the letter h is negatively correlated. Given these two correlations, the following observation can be made:

$$Complexity = Freq(e) - Freq(h) \quad (1)$$

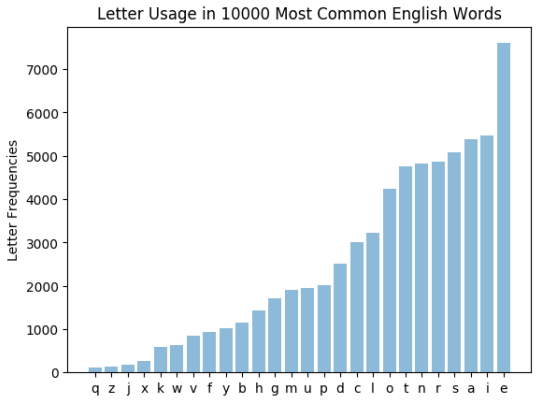

Fig. 1. Letter usage in Common English Words

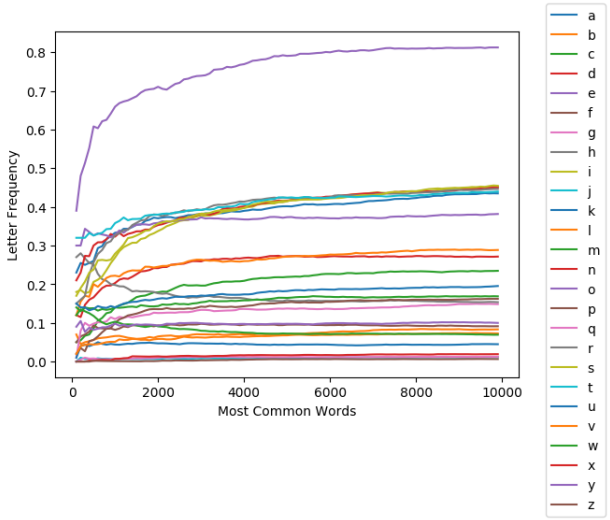

Fig. 2. Letter Frequencies in Common English Words

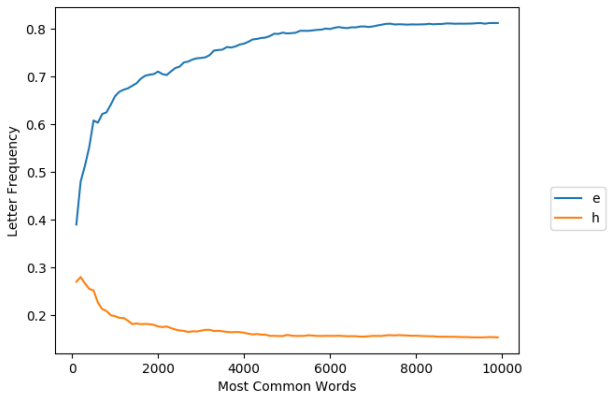

Fig. 3. Letter Frequencies in Two English Letters

To gather a suitable subset of English text to verify these results, we used the following texts:

1) Grimm's Fairy Tales
2) Pride and Prejudice
3) 500 most common english words
4) A selection of US copyright law

	Grim's Fairy Tale	Pride and Prejudice	Copyright Law	500 Common Words	Paper Collection
Freq (e)	0.1286	0.1291	0.1138	0.1447	0.1277
Freq (h)	0.0793	0.0627	0.0401	0.0615	0.0337
Freq (e) - Freq (h)	0.0493	0.0663	0.0737	0.0833	0.0941

Fig. 4. Calculated complexity in given text

	Grim's Fairy Tale	Pride and Prejudice	500 Common Words	Copyright Law	Paper Collection
Freq (e)	0.1286	0.1291	0.1447	0.1138	0.1277
Freq (h)	0.0793	0.0627	0.0615	0.0401	0.0337
Freq (e) - 3.33* Freq (h)	-0.1355	-0.0799	-0.0600	-0.0196	0.0157

Fig. 5. Calculated weighted complexity in given text

5) A collection of articles from an academic journal

These texts were chosen due to their perceived complexities, namely that works of literature like Grimm's Fairy Tales should be significantly less complex than US copyright law or an academic journal.

Figure 4 shows the complexity values for each of these texts. Notably, copyright law was computed to be less complex than the 500 most common English words. Because of this seemingly odd data point, we reevaluated what we were measuring.

Figure 5 shows the newly calculated results. We recognized that we were actually calculating the relative frequency of the letters for a given word, and the letter e has a greater rise from approximately 0.4 to 0.8, while the letter h only decreased from 0.27 to 0.15. In order to account for this discrepancy, we decided to weight the letter h based on the following formula:

$$Weighting\ Factor = \frac{(0.8 - 0.4)}{(0.27 - 0.15)} = 3.33 \qquad (2)$$

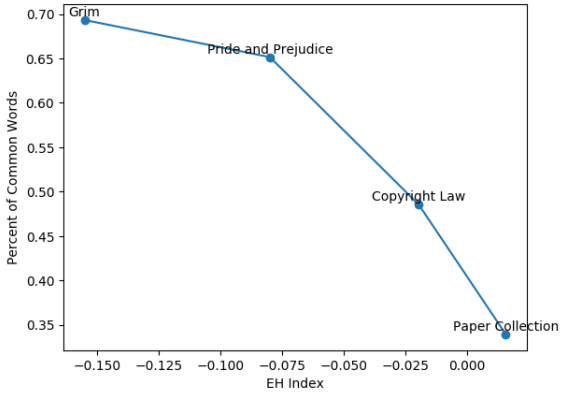

Fig. 6. Complexity value based on piece of literature

Figure 6 shows the results of these complexity calculations for each piece of literature. As was expected, the highly technical work has a much higher complexity index, and the works of fiction have a very low complexity index.

C. Analysis of Library of Babel

In order to validate the results of the Library of Babel implementation we used, we performed a letter frequency analysis of 2 distinct pages of the Library, as well as a

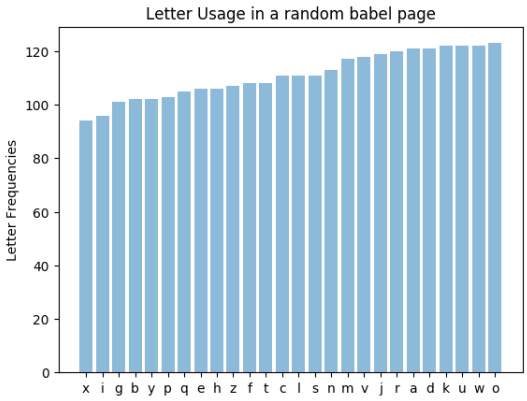

Fig. 7. Letter frequency of a random page of the LoB

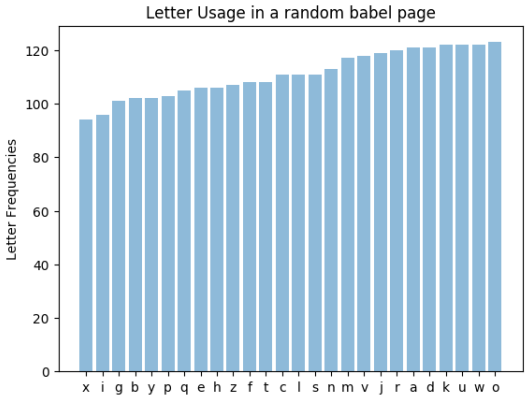

Fig. 8. Letter frequency of a random page of the LoB

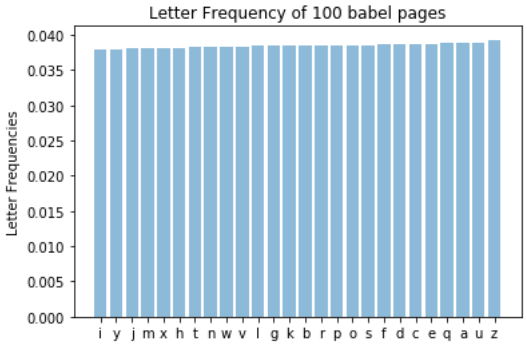

Fig. 9. Letter frequency of 100 pages of the LoB

collection of 100 separate pages. These results can be found in figures 7, 8, and 9. As can be seen in these figures, the Library appears to be truly randomly distributed.

Because we devised a protocol to observe the randomness found in the text, we could apply that method to the library. As the figures show, however, there is no structure to be found in the library as a whole, although the metric may be interesting if run on individual pages. The application of such a metric is left to the reader.

D. Toy Example

As an illustration, we provided a simple implementation of an encryption protocol. This program takes in any given sentence which contains the legal English characters, and converts this piece of text to it's address in the Library of Babel. By using this index, the exact same text can be retrieved from the library. By using the addresses in the Library of Babel, we created a simple encryption system. Although this encryption is primitive and certainly not suggested in serious cases, it showcases one way to make use of the Library of Babel. Implementation of this simple diary program can be found in the class GitHub repository.

VI. Team Assignments

Dakota had approached the team with the idea of exploring the Library of Babel and finding elements of it that could potentially be re-implemented. The team worked together to plan and research what could be done with the Library and how we planned to incorporate that. During the research, Rachel focused on numerical analysis and determined what mathematical constraints needed to be placed onto the project. With the constraints in place and a direction of where the project would head, the team began coding and solving some of the questions we had proposed. Lu focused on determining the letter frequency for different types of literature, as well as crafting an insecure and experimental form of encryption based on the data we had gathered. The team came together again to collaborate and write our presentation and final report.

VII. Future Work

The Library of Babel is an inherently chaotic data set that we have shown here to have little structure in terms of a modern natural language. Unfortunately, such lack of structure makes text parsing not only difficult, but useless, as there is nothing there to find; therefore, future work on this project may stray from our original goal of text parsing.

We did, however, show that there are some toy problems that can make use of the Library of Babel's lack of structure - like a naive encryption technique to write one page diary entries and store these entries for future visitation with the long sequence of what hexagon, wall, shelf, book, volume, then page the entry belongs on. One future project could be to find more toy uses for the Library of Babel like the naive page encryption to mess around with and use as a fun side project. Unfortunately, without appropriate data storage, creating a usable and manageable subset is not feasible with today's technology, bandwidth limitations, and lack of large, cheap storage systems.