IRRS Lab 1 - Crawling, preprocessing and text statistics

Mathias Pettersen Emmanuel F. Werr

September 27, 2022

1 Introduction

This lab-exercise is an introduction to lingustics and statistics of text. In this lab, we will analyze several text files and compare the statistics of them to the expected, mathematical approximation.

Thus we will extract the text-information, and see if the statistics uphold **Zipf's Law** and **Heap's Law**.

2 Mathematical Preliminaries

Let's first look at the mathematical basis of the two aforementioned laws.

2.1 Zipf's Law

Zipf's law states the following:

$$f(r) = \frac{c}{(r+b)^{\alpha}} \tag{2.1}$$

..where $f(r)$ is the frequency of a word, r is the rank (position when sorted by frequency in a descending order), b and c are constant, and α is a particular constant which is equal to the absolute value of slope the resulting log-plot.

Now what interests us, is verifying whether or not the texts follow this power-law. To do this, we log-transform the formula in order to visualize it better (as we will see later).

This yields the following mathematical results: Let $f(x) = \frac{c}{(x+b)^{\alpha}}$ and $y = f(x)$.

$$\log(y) = \log(c \cdot (x+b)^{-\alpha}) \implies \log(y) = \log(c) - \alpha \log((x+b))$$

As mentioned, α is the slope of the log-plot. We can therefore calculate it like so:

$$\alpha = \left| \frac{\Delta \log(y)}{\Delta \log(x+b)} \right| = \left| \frac{\log(y_2) - \log(y_1)}{\log(x_2+b) - \log(x_1+b)} \right| \tag{2.2}$$

That leaves us with the formula we'll be using:

$$\log(y) = \log(c) - \left| \frac{\log(y_2) - \log(y_1)}{\log(x_2+b) - \log(x_1+b)} \right| \cdot \log(x) \tag{2.3}$$

2.2 Heap's Law

Heap's law is a power-law that describes the number of *distinct* words in a text compared to the *total* number of words. It is described with the following formula:

$$d = k \cdot N^{\beta} \tag{2.4}$$

..where d is the number of distinct words, k and β are constants, and N is the number of total words in the text.

Now this formula is a bit easier to turn into a log-version. Let $e^{k_1} = k$, then

$$d = e^{k_1} \cdot (e^{\log N})^{\beta} = e^{k_1} \cdot e^{\cdot \log N^{\beta}} = e^{k_1} \cdot e^{\beta \cdot \log N} = e^{k_1 + \beta \cdot \log N} \tag{2.5}$$

$$\implies \log(d) = \log(e^{k_1 + \beta \cdot \log N}) = k_1 + \beta \cdot \log N \tag{2.6}$$

3 Analyzing Zipf

We used the above mathematical statements in order to analyze the statistics:

1. We used ElasticSearch to extract the text from all the text-files
2. We removed any non-alphabetic lines in the resulting file (i.e. words that don't belong within a-z, A-Z)
3. We then aggregated the frequencies into a list, and then used this to create an estimate of what the result should look like according to Zipf's law
4. We experimented manually with k and b, where k was simply chosen by visually estimating where the *real* log-plot would cross the y-axis, and b was just done by trial and error

This yielded the following results:

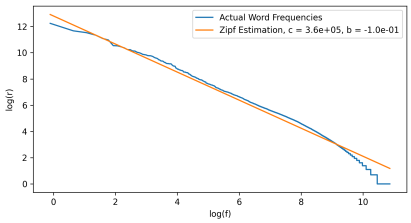

Figure 1: Novels

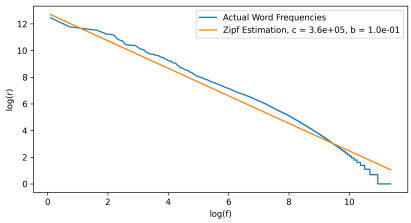

Figure 2: News

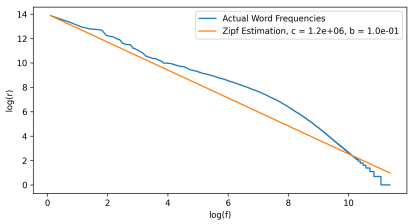

Figure 3: ArXiv

4 Analyzing Heap

This analysis was similar to the one in Zipf, however with one small change:

- Instead of estimating the hyperparameters, we created a function that would calculate the RMSE-loss[1] between the actual number of distinct words d, and the *estimated* number of distinct words according to Heap's law, depending on the choice of hyperparameters β and k
- We also created 4 different indices
  - One with the entire novels-set - size: 17.4MB
  - A subset of the novels - size: 4.28MB
  - Another subset of the novels - size: 943kB
  - A final subset of the novels - size: 12.2MB

This yielded the following result:

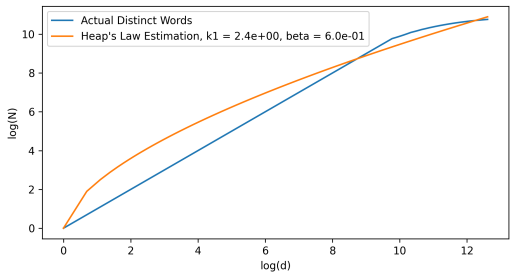

Figure 4: Novels - All

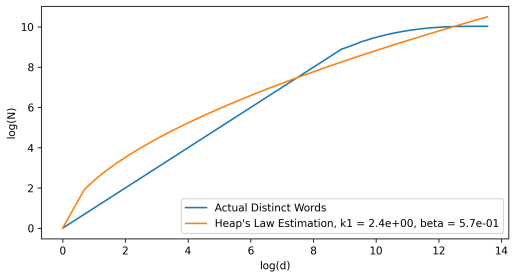

Figure 5: Novels - Subset 1

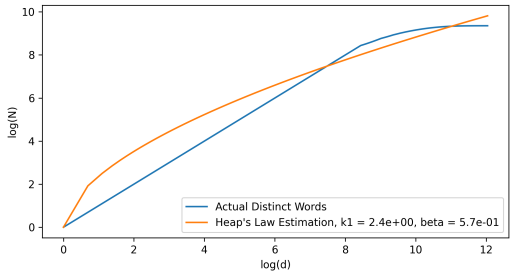

Figure 6: Novels - Subset 2

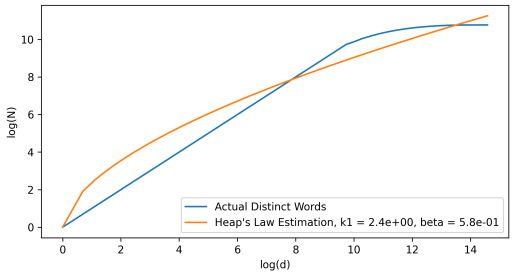

Figure 7: Novels - Subset 3

[1] https://en.wikipedia.org/wiki/Root-mean-square_deviation

5 Discussions

What we noticed in Zipf's law, was that the texts in general held up extremely well according to the law. In the texts in general, it did seem like the frequency of a word was inversely squared proportional to its rank. The trend was obvious and for the most part accurate, but we also saw some changes with the different types of texts. Especially, we saw some slight variation with the ArXiv-texts, which makes sense as scientific papers use a vastly different vocabulary and structure than story-texts, novels, and general news-articles.

In Heap, we noticed some slight differences. Here it was difficult to argue that the formula and power-law matched *exactly*, but it was rather apparent that it followed the same *trend*. What we mean by this, is that getting the equation to fit the actual statistics exactly, was more or less impossible. None of the graphs (normal; not log-log) fit really well to the original line. However in the log-log plots, we saw that the general trend of the approximation matched the actual results quite well. So even though it did not exactly match, it did succeed in extracting the most important characteristic, i.e. the more words you have in a text, the less likely you are to encounter a completely new word.

6 Struggles & Final notes

In the beginning of this lab, we had some struggles using the ElasticSearch-server and setting it up properly. This was partly due to the UPC-computers deleting everything once you log out, so we more or less lost a week of work.

Apart from that, there were some challenges in interpreting the meaning of the different formulas, and how to actually apply them in the code and use them for calculation and approximation. However in the end, we managed to solve it adequately.