Information Retrieval

ElasticSearch and Zipf's and Heaps' laws

Laura Cebollero Ruiz, Alexandre Rodríguez Garau

3rd October, 2018

1 Introduction

For this Lab session we were asked to work with *ElasticSearch* and learn how this tool works. Among other things, we learned how to index documents in a local database, how to work with those indexes and search for words or query using boolean word vectors.

The last and most important step has been to work with Zipf's and Heaps' laws.

2 Delivery files

- `shuffle_files.py`: Takes random novels and puts them in subfolder in order to index them afterwards.
- `create_indexes.py`: Creates 10 indexes from taking the files from the subfolders.
- `deleteIndexs.py`: Deletes the newly created files from the elasticsearch database.
- `zipfs.py`: File that computes the curve *a, b, c* fitting our data and plots it.
- `heaps.py`: File that computes β and k from a given sample of d and N.

3 Preprocessing

In order to work with what we deem as a clean word corpus, we have performed many steps:

1. Not take into account dates, numbers, words with special characters. Done by adding a new function called `isBadWord` on the `CountWords.py` file.
2. **Stopword removal**. That is, we have removed unimportant words that appear frequently on the English language using a stopword list that we have obtained online [1].
3. Created files containing the frequency of every word ranked from most frequent to less frequent.

Having done this, we have proceeded work with our data.

4 Zipf's and Heaps' laws

Having cleaned our data, now we can proceed onto checking the Zipf's and and Heaps' law.

4.1 Zipf's law

Zipf's law states that, in a corpus of natural language words, the frequency of every word is inversely proportional to their index in the frequency table. Specifically, this frequencies are distributed following this expression —Zipf-Mandelbrot equation:

$$f(i) = \frac{c}{(i+b)^a}$$

[1] Natural Language Toolkit `https://gist.github.com/sebleier/554280`

However, Zipf himself postulated in the 30s that $a = 1$, but later it was found that $a \approx 1$. And so one could approximate Zipf's formula to being when setting $b = 0$:

$$f(i) \approx \frac{c}{i^a}$$

4.1.1 Zipf's law with corpus with stopword removal

To check whether the word count follows Zipf's, we have plotted the frequencies on the y axis and the rank of the words —*i.e.* the position of the word in the sorted list— on the x axis and applied logarithms to scale them both.

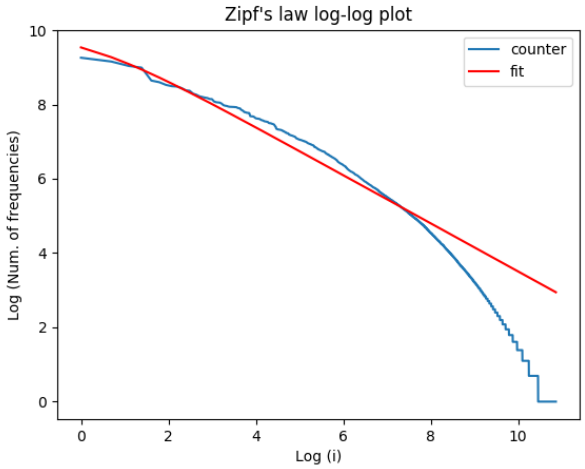

Figure 1: Zipf's law, with stopword removal

As we can see, the number of frequencies per each word follows a decreasing curve as the number of frequencies decreases, and so the regression line does not fit that well with those words at the end of the list.

According to our fitting execution, the best values of the regression to fit this curve are:

- **a: 0.648**
- b: 0 (fixed)
- **c: 21688.48**

4.1.2 Zipf's law with corpus without stopword removal

As extra work, we also wanted to see how the stopword removal affected the curve in Zipf's law as well as the regression line, so we executed it with our clean corpus but **without stopword removal:**

Figure 2: Zipf's law, without stopword removal

In this case we can see that curve is not that prominent so the regression fits more. In this case the obtained values are:

- **a: 1.04**
- b: 0 (fixed)
- **c: 412328.887**

4.2 Heaps' law

Now, we are going to study Heaps' law:

$$d = kN^{\beta}$$

With Heaps' law we can estimate the number of different words that are present in a corpus by first computing it in a subset and then extrapolate the results, which are not linear.

In other words, if we know β and k for our corpus, we can extrapolate the number of different words one document will have if we extend it and make it larger.

So, for our corpus, we want to compute β and k so that we are able to estimate how many different words are in total in our corpus.

To do so, we have created 10 different subsamples from all the novels and indexed them, each with an increasing number of novels s, where

$$s \in S = \{2, 5, 7, 9, 13, 16, 19, 22, 27, 32\}$$

knowing that we have 33 novels in total.

This shuffling and splitting of the corpus has been done with the described files `shuffle_files.py` and `create_indexes.py`. As a reminder, the files taken for each set have been **chosen randomly**.

Once this subsamples have been done, we counted for each set the total number of words (N) and the number of diff. words in that sample/set (d). This way, only k and β are the unknown values on Heaps' equation.

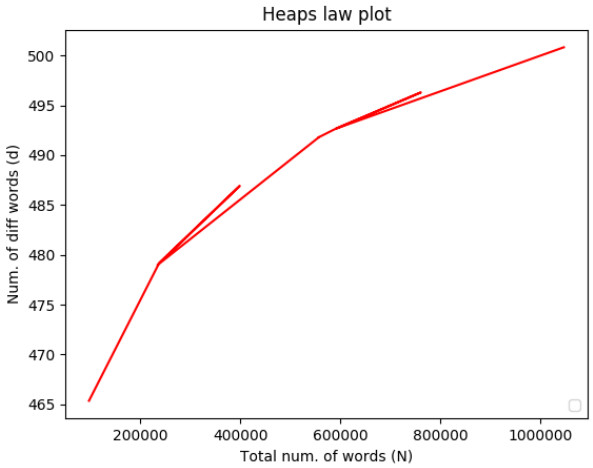

Figure 3: A plot showing how d grows according to N in our subsamples

Finally, we have computed a curve that fits these values N and k in order to estimate a fitting β and k for our corpus. The result has been the following:

- k: 179.66
- β: 0.389

Figure 4: A plot estimating how d grows according to N in logarithmic scale

The value of k is inside our expected range $30 \leq k \leq 100$. [2]

However, if we take a look at β, we can see that it is way lower than 0.5, the value we expected as seen in class for an English corpus.

Nevertheless, **we deem this value correct** and attribute it having a low value to the fact that we applied the **stopword removal** technique to our corpus.

[2] `https://nlp.stanford.edu/IR-book/html/htmledition/heaps-law-estimating-the-number-of-terms-1.html`