CAIM Lab, Session 1: ElasticSearch and Zipf's and Heap's laws

Ana Mestre Borges

Carlos Roldán Montaner

Zipf's law report

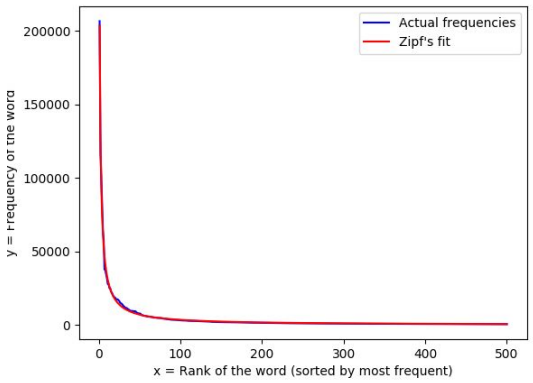
Our goal was to find out whether or not the rank-frequency distribution of words within a large set of texts seemed to follow a power law. In particular, we wanted to see if it followed Zipf's law $f = \frac{c}{(rank + b)^a}$ and if it did, to adjust the parameters so that the formula described the data in the best way possible.

We indexed the novels corpus to extract all the words and their respective frequencies. Then we filtered all the urls, numbers, etc. When we had only proper words, we began our study.

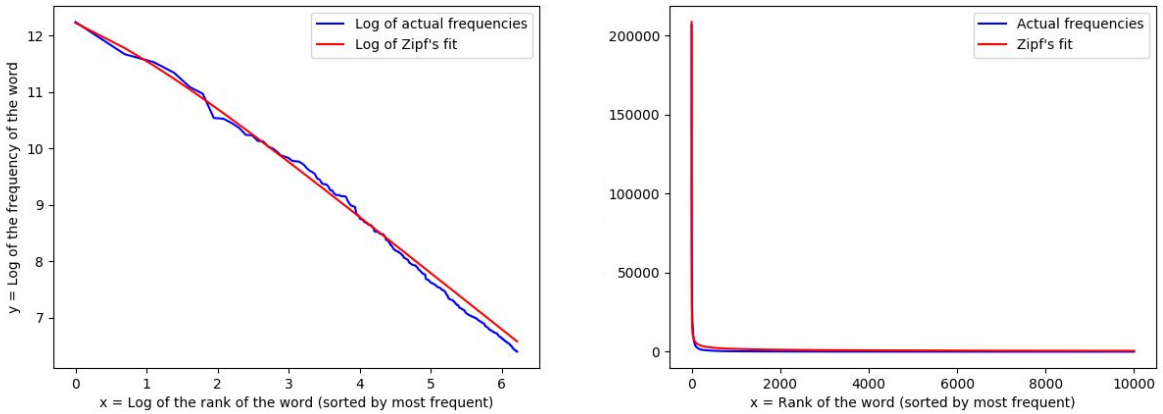
We started making experiments with the 500 most frequent words. After plotting the rank-frequency distribution we obtained a decreasing curve. It showed really high frequency values for the first ranks, while all the lower ranked words had similar low frequency values. We tried to adjust the parameters of the Zipf's law to get a matching curve. To do so, we used the **curve_fit** method on the **scipy** Python library.

Our first attempts failed, since we could not obtain a similar looking curve. We tried plotting the log-log distribution and matching that instead, to no avail. Then we started giving manual values to the **a** parameters and letting the library method do the adjusting for the **b** and **c** parameters.

With a value of **a** = 1 we instantly got an almost perfect fit (**b** = 0, **c** = 361713). We also checked whether the log-log plots matched, and as expected, we also obtained a really good fit using the same parameters, as we can see in the following plots:



With these experiments, we concluded that the rank-frequency distribution does follow Zipf's law. However, we did some more testing and realized that the parameters that we thought were almost optimal produced very poor results when we increased the number of words. For instance, using the same parameters and 10000 words instead, we obtained a curve that didn't match at all. This leads us to the conclusion that the parameters not only depend on the words, but also on the amount of them. However, the power law still holds, since we managed to obtain another good fit for 10000 words by changing the **a** parameter to 0.5 (**b** = 0, **c** = 56163), as we can see in the following plot:



Heap's law report

The Heap's law is an empirical law which describes the number of different words in a document. It is formulated like this:

$$d = K x N^{\beta}$$

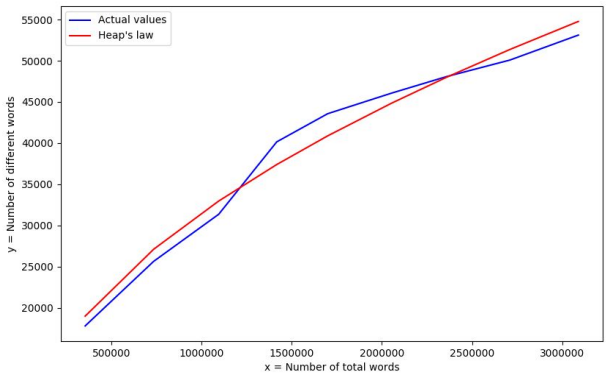
d represents the number of different words

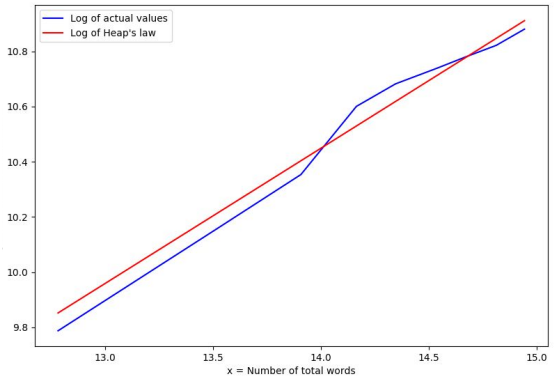
N represents the size of the document (the number of words)

K and β are free parameters

Our goal was to find K and β in order to prove if it follows the Heap's law. As we have to analyze different sizes of documents, the easiest way was to remove each time some of the novels of the collection we are given. To make it easier and clearer, we chose the biggest size of a document, that was the size we removed at each iteration (2MB). Therefore, we had 9 different indexes for our study. For each index, first we had to clean a bit the documents, so we deleted all the url's, numbers... and second, counted the number of different words and the total number of words. Following the same methodology used before, the **curve_fit** method on the **scipy** Python library, we obtained the following data and plot:

K = 36, β = 0





As we wanted to be sure of the results obtained, we executed the same code but this time drawing the log-log-plot. As we can see in both plots, we can conclude that it does follow the Heap's law.