**CAI Lab Session 1**

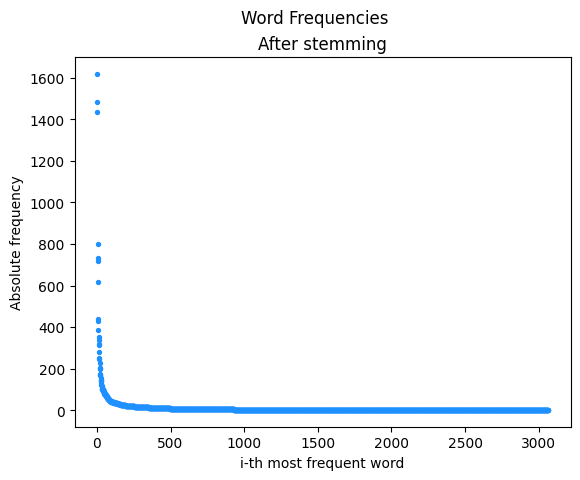
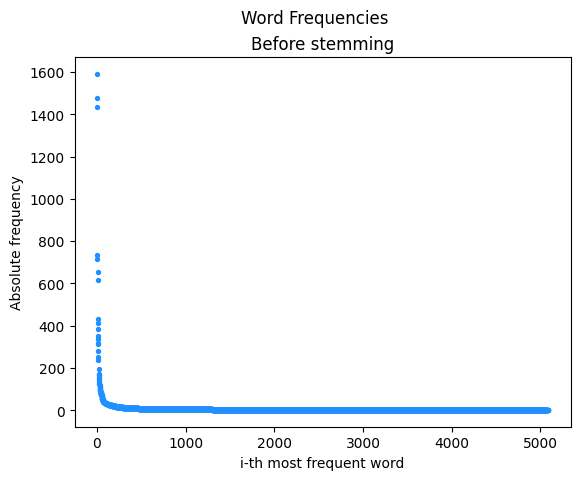
Power-laws

Gerard Comas & Laura García

To carry out the first 3 exercises, we have taken the first lines of *Don Quijote de la Mancha*. After tokenizing the text, we filtered it by taking out only the punctuation signs and lowercase-folding the resulting words. As suggested in class, aiming to obtain better graphical results, we have not filtered stopwords.

**Exercise 1**

The first lab session exercise asks us to plot the frequency of the words appearing in the chosen extract, ordered by said frequency. We want to see if we can express the frequency of the -th most frequent word as a function of . Throughout the first 3 exercises, we’ll plot two cases: before stemming the filtered text and after stemming. Afterwards, we’ll compare both results.

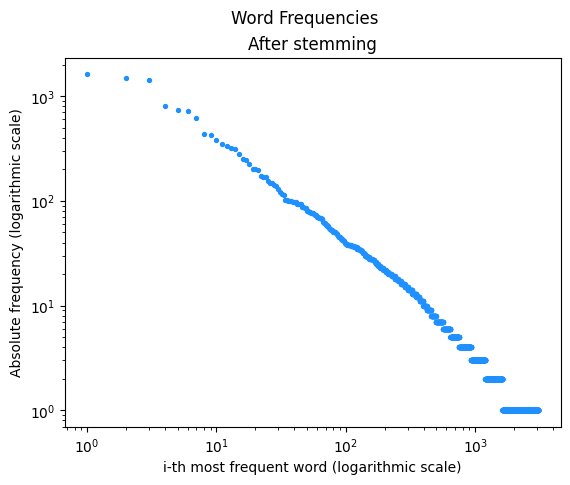
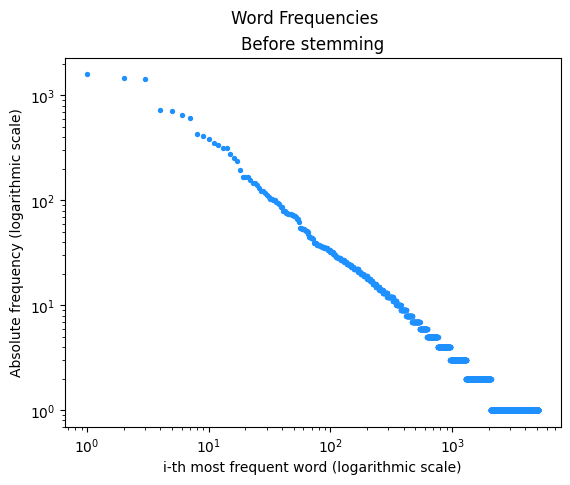


At first sight, we can observe that the number of words after stemming has reduced significantly, and the frequencies are considerably higher, which makes sense for it is likely that many words have the same root. This is the main difference between both cases. Aside from this, there are not many more differences: both graphs have the same general properties: they decrease rapidly with . We could expect both of them to be modeled by power-laws (due to this exponential decrease of the frequency), though we can’t fully assume they do.

We’ll need to plot the data in logarithmic scales (as done in the next exercise) to be able to confirm this hypothesis.

**Exercise 2**

If we plot these results but with logarithmic scales for the and axis, we obtain two other graphs that could be way more useful if we want to fit a power-law to these data.



For this exercise, the results look way more promising, slightly better before stemming. We can now observe a straight line with a negative slope, which is something we were expecting. That’s why we could say that, after looking at the results in a log-log scale, they may be approximated with a power-law. Again, the slope is overall constant before stemming, though it looks like it varies a bit after stemming the filtered words.

Thus, our previous hypothesis has been confirmed: we can approximate the frequency of the -th most frequent word with .

**Exercise 3**

In this exercise, we’ll want to fit the parameters of Zipf’s law to the data we have extracted. Zipf’s law states, usually with (above and below ):

We’ll be focusing on a case where we assume and, change the notation a bit to get:

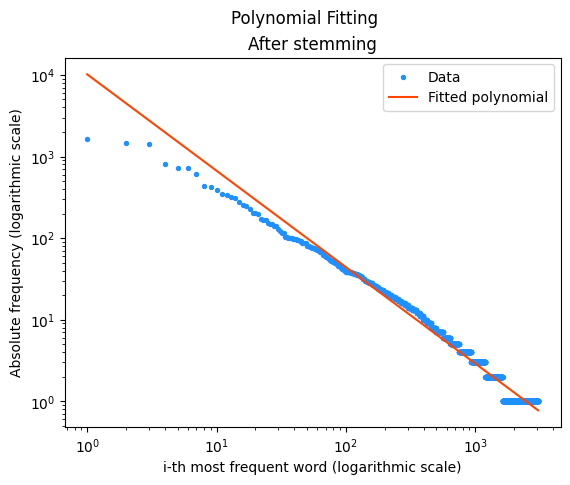
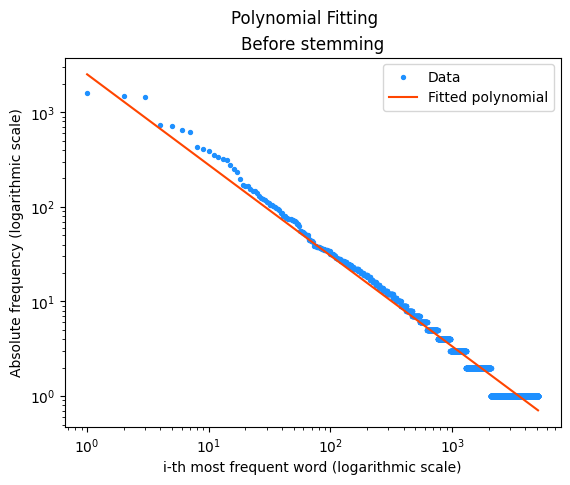
Now can be taken with the same value as before and the opposite sign; and we only need to focus on fitting two parameters.

To find the values of the parameters of the power-law followed by the data, we have used the polyfit method from the *numpy.polynomial* python library to find a least-squares solution. With it, we have found and in , which is equivalent to Zipf’s law. These values correspond to the slope and the -intercept of the two graphs above, respectively. Then we computed .

We have obtained the results shown next:

|  |  |  |
| --- | --- | --- |
| Before stemming |  |  |
| After stemming |  |  |

Note that in both cases, we get , which is expected (in the first formulation of Zipf’s law, is really the exponent of the denominator and ought to be positive). If we fit these polynomials and plot them against the real data, we have the next graphs:



We can see that, as suspected (since the slope was not constant through all the plot for the case after stemming), the fit looks clearly better when we haven’t stemmed the words of the text. It is in this case that the fit looks almost perfect. However, after having stemmed the words, the fit does not perform nearly as well. It looks like the difference between the values obtained for tries to compensate for this slight curvature, which ends up diminishing the fitting capability of the approximation.

In all the previous exercises, we observe that the results turn out better before stemming the words on the text. This may be due to Zipf’s law itself, which aims to model the frequency of each word individually, not taking into account its meaning or where it comes from.

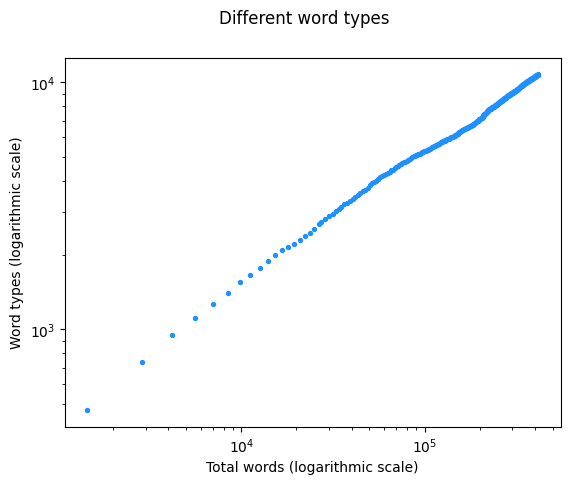
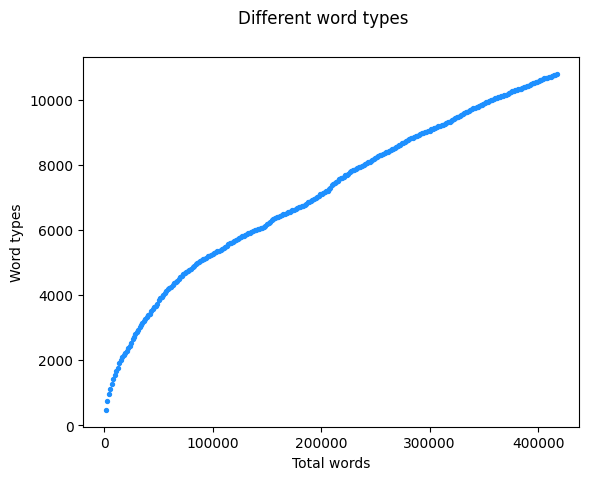
Besides, Zipf’s law tries to capture the natural use of a language (in this case in a text). When stemming the words, we are taking words that are part of the natural use of the language out of the sample. Hence, the set of words being analyzed is not completely natural and using the law for it makes less sense.

**Exercise 4**

For the last exercise, the objective is to check whether Heap’s law applies in the *Don Quijote*. We do this by plotting the number of different words or word types (as in after stemming the words of the text, we’ll only want to count semantically different words) as a function of the text length (total number of words in the chosen fragment). Heap’s law is shown in exponential form next, and also its equivalent formulation in logarithmic form.

In these formulas, is the number of different words and the length of the text. Both and are the constants of Heap’s law, to be determined for each text. In this exercise we’ll try to determine the value of .

First we plot the number of different words against the total length of the fragment (left plot). Even though this shape could allow us to assume there is a power law involved in the modeling of both variables, we also plot the logarithms of the different words in the text () and the total length of the text (). Here we could definitely see a straight line, which confirmed what we thought before: the number of different words in the *Don Quijote* follows Heap’s law.



Now the last step to be carried out is to find the values of the constants and of said law. Just as in the previous exercises, we did a polynomial fit of a first degree polynomial to and . We finally got the next results:

and

Regarding the value of , which was the parameter we were interested in, we observe 2 things that we expected before carrying out the experiment:

1. The longer the text, we’d expect a higher number of different words: is positive so the number of different words increases with .
2. The longer the text already seen, the lesser the chance of finding new terms: is smaller than .

Finally, we plotted the data in logarithmic scales again, also plotting the fit of the resulting polynomial. We observe that the results are pretty exceptional and fit almost perfectly. It is not so perfect for small values of , which makes sense given that the longer the text, the more data and the more likely it will be for any model (based on that data) to better characterize it.

