Task 3: Statistical laws of language

This task is not automatically ticked. You should write down your observations in the lab book and keep screenshots of your graphs so that you can show and explain appropriate figures and data to your ticker.

This is the first part of Tick 3. Tick 3 combines this task and Task 4, so don't contact the demonstrator for the tick until you have completed Task 4 as well.

Step 1: Zipf's law

Zipf's law says that there is a reverse exponential relationship between the frequency of a word in a large natural language text f_w, and its relative frequency rank r_w (position wrt frequency in comparison to the other words):

$$f_w \approx \frac{k}{{r_w}^{\alpha}}$$

where k and α are constants that depend on the natural language. This means that given the frequency of the most frequent word, the frequency of the next most frequent word can be predicted (but not which word it will be). There are factors in language that make this observation slightly surprising. Zipf's Law is one example of a power law in nature, and it has been found to be applicable to many phenomena other than word frequencies, for instance global sizes of cities.

How does this apply to Task 2?

Download the large dataset, which contains about 35,000 assorted reviews, with about 11 million words. To which degree does Zipf's law hold in this sample of text? And what are the parameters?

1. Find frequencies of all the tokens in the dataset and rank them.
2. Plot a frequency vs rank graph for the 10,000 highest-ranked tokens. You can use `GraphPlotter.java` to get the plot. **Comment 29/1: Clarification note: we are expecting people to just use the count of the words here for frequency. However if you've done this part of the work already and used a proportion, there's no need to change.**
3. In Task 1 you were asked to choose 10 words which might be good sentiment indicators. What are their frequencies? Plot the Task 1 words on the frequency-rank plot as a separate series (i.e., tell the plotter to draw it as an additional graph in addition to the graph from 2. above). In which region of the graph do they occur?
4. Plot the main graph on the log-log scale. (Why?) Note: some plotting packages offer automatic conversion of datapoints to log scales when

plotting, but not the one we provide. You will have to convert the values yourselves before plotting.

5. Fit a line to the log-log graph, using the least-squares algorithm provided in `BestFit.java`. Weight each word by its frequency to avoid distortion in favour of less common words. Add the line to the plot.
6. Use the best fit line to create a function which given a rank can output an expected frequency. What frequencies does your estimate predict for the Task 1 words? How do they compare with actual frequencies?
7. Use the best-fit line to estimate the Zipf's law parameters k and α.

Step 2: Heaps' law.

Heaps' law relates unique and non-unique words in a text. In computational linguistics, we refer to unique words as **types** and to non-unique words as **tokens**. The law states that for a large text of size n, the number of types u_n in the text approaches:

$$u_n \approx kn^{\beta}$$

with k and $\beta < 1$ constants dependent on the language. Surprisingly, this means that no matter how many new documents we will include and compare to an already existing set, we will continue to find new unseen words.

1. Count how many unique words the system finds in your input files for any given number of tokens. Collect a datapoint every time the total number of tokens you have read in reaches a power of two (2^0, 2^1, 2^2, etc. until you reach the size of the dataset). Also provide a data point for the total number of tokens in all texts (which does not correspond to a power of two). Plot these datapoints on a logarithmic scale, as before.

What does the line look like?

Optional: What kinds of new tokens are collected in the last few documents (i.e., after all "obvious" tokens have already been seen)?