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.

Step 1: Zipf's Law

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

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

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). 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 unique words in the dataset and rank them. There is no need to remove punctuation from the set of words.
2. Plot a frequency vs. rank graph for the 10,000 highest-ranked words. You can use the function `chart_plot(data, title, xlabel, ylabel)` in `utils/sentiment_detection/plots.py` to generate the plot, which uses the `matplotlib` package.
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 *on the same plot* (i.e. draw it as an additional graph on top of the graph from 2. above). In which region of the graph do the words occur?
4. Plot the main graph on the log-log scale. Why do we do this? You will have to convert the values yourselves before plotting.
5. Use the least-squares algorithm to fit a line to the log-log graph. This is also provided in `utils/sentiment_detection/plots.py`. Then, create

a function which given a rank can output an expected frequency. What frequencies does your estimate predict for the 10 words you wrote down for Task 1? How do they compare with actual frequencies? Write down the predicted and expected to show to a demonstrator when you get ticked.

6. Plot the line of the least-squares algorithm on the same plot as the main graph on the log-log scale.

7. Use the best-fit line to estimate the Zipf's law parameters k and α.

Step 2: Heaps' Law

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

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

where k and $\beta < 1$ are 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.

8. 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).

9. 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)?