

Homework #5: CMPT-825

Anoop Sarkar – anoop@cs.sfu.ca

- (1) For all sub-parts of this question, use the text file called `hw5.txt` from the location provided on the course web page.

Write a program that collects the frequency of each word and prints it out sorted by descending frequency. The first few lines of the output should look like this:

```
3842 ,
3116 the
2310 plant
2244 .
1740 a
...
```

- (2) **Zipf's Law** Modify your code from Q.1 to print out the line numbers for each word and its frequency as shown below:

```
1 3842 ,
2 3116 the
3 2310 plant
4 2244 .
5 1740 a
...
```

Save the entire output of your program to a file called `plotfile`:

The line numbers correspond to the sorted rank r of each word based on its frequency f . Zipf's law states that $f \propto \frac{1}{r}$ or, equivalently, that $f = \frac{k}{r}$, for some constant factor k . In this question, we will use the GNU tool `gnuplot` to plot Zipf's formula against the empirical relationship between a word's rank and its frequency.

At the unix shell, enter the command `gnuplot` and enter the following commands at the `gnuplot` shell. This will produce a plot that compares Zipf's formula with the empirical distribution you have stored in `plotfile`. We plot on the log scale, plotting $\log(r)$ on the x-axis and $\log(f)$ on the y-axis. Since there are 8200 tokens in our file `hw5.txt`, we vary $r = 1 \dots 8200$ and set the value $k = 10000$. The `gnuplot` commands are:

```
set logscale
set xlabel "log(rank)"
set ylabel "log(frequency)"
plot [x=1:8200] 10000/x title "Zipf", 'plotfile' using 1:2 title "Empirical"
```

Now compare Zipf's formula with the Mandelbrot formula

$$f = P(r + \rho)^{-B}$$

or

$$\log(f) = \log(P) - B \cdot \log(r + \rho)$$

This particular command to `gnuplot` uses the parameter settings: $P = 10000$, $\rho = 100$ and $B = 1.15$.

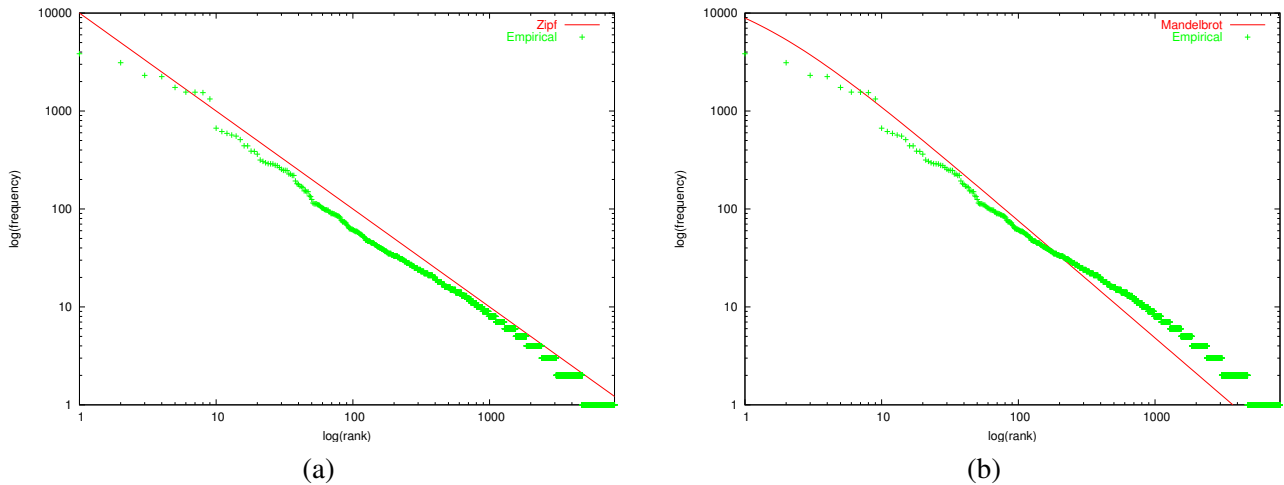


Figure 1: (a) Zipf's formula compared with the empirical distribution of word frequencies (b) Mandelbrot's formula compared with the empirical distribution.

```
set logscale
set xlabel "log(rank)"
set ylabel "log(frequency)"
plot [x=1:8200] [1:] 10000*(x + 100)**(-1.15) title "Mandelbrot", \
    'plotfile' using 1:2 title "Empirical"
```

You can save your output as a postscript file by using the following commands *before* you enter the plot command:

```
set terminal postscript eps color
set output "filename.eps"
```

- Compare your output with Figures 1.1 and 1.2 from the textbook (pages 26–27). Change the parameters P , B and ρ in the Mandelbrot function to match the observed distribution of word frequencies for `hw5.txt`.
- What happens to the Mandelbrot equation when $B = 1$ and $\rho = 0$?
- Ponder the implications of Zipf's law for natural language processing. Let's assume you have an NLP application which uses a dictionary of words. Based on Zipf's law or Mandelbrot's law or the empirical distribution plotted above, approximately how many words taken from a new (previously unseen) text do you expect to match in your dictionary.
- Is Zipf's Law some deep property of language, or is it simply that any random (stochastic) process would have the same property? Confirm or deny this hypothesis using the following experiment: write a program which creates random words out of the (lowercase) English alphabet. Compare the plot for a large enough set of random words with the English data.
- In this question we consider the implications of Zipf's law for natural language processing and the role of morphological processing. For this question you will need an additional file containing a list of English words called `dict.txt`.

Your perl program should take two files (`dict.txt` and `hw5.txt`) as input and print out 4 numbers:
68737 8200 5108 2077

Each number is explained below (the first 3 numbers should match with your implementation):

- the total number of word tokens in `hw5.txt`,

2. the total number of word types in `hw5.txt`,
3. the total number of word types from `hw5.txt` that were **not** observed in `dict.txt` (notice that this number roughly corresponds to the number of single count words predicted by Zipf's law),
4. the total number of word types from `hw5.txt` that were not observed in `dict.txt` after using regular expression substitutions to find the **stem** of each word type in `hw5.txt`.

For example, consider the following two Perl regex substitutions. The first removes the `im` prefix from the word in `$wd` only if the stem begins with `p`, `b` or `m`, and the second removes the `ly` suffix from any word:

```
$wd =~ s/^im([pbm])/$1/;  
$wd =~ s/ly$//;
```

As a result, the word `improperly` which occurs in `hw5.txt` but does not occur in `dict.txt` is converted into the stem `proper` which does occur in `dict.txt`. You should chain together an ordered list of such substitutions to get the lowest number you can for missing matches. Use the many examples of prefixes/suffixes in English from the web to create substitution rules. You should obtain a number that is ≤ 2554 .

You can print out the words that do not match to `STDERR` in order to improve your ad-hoc Perl stemmer and minimize the number of missing matches. You can optionally use a Perl implementation of the Porter stemmer from the web:

<http://www.tartarus.org/~martin/PorterStemmer/perl.txt>