
Table of Contents

CS 540 - Spring 2018 - HW5	1
(a) Brief explanation of method	1
(b)(i) Total words across n essays	1
b(ii) Total distinct (unique) words across n essays	2
(c) 20 most frequent words and their counts	2
(d) 20 least frequent words and their counts	3
(e) Plot rank r vs count c	3
(f) Plot log(r) vs log(c)	4
(g) Explain the shapes of the curves	5
(h) Two potential issues with this method of preprocessing	5

CS 540 - Spring 2018 - HW5

Question 2 by Roumen Guha

```
clc
clear
close all

addpath('C:\Users\roume\OneDrive\Documents\Classes\4 - Senior
\Spring 2018\CS 540 - Intro to Artificial Intelligence\Assignments
\HW5\WARC201709')
source_dir = 'C:\Users\roume\OneDrive\Documents\Classes\4 - Senior
\Spring 2018\CS 540 - Intro to Artificial Intelligence\Assignments
\HW5\WARC201709'; %uigetdir([]);

d = dir([source_dir, '\*.txt']);
n = length(d(not([d.isdir])));
```

(a) Brief explanation of method

I implement MATLAB for this task (and live to regret it). We take in the strings from the filepath and preprocess them by first converting them to lowercase, then removing punctuation, then splitting them by "whitespace".

(b)(i) Total words across n essays

```
total_words = 0;

for i = 1:n
    txt{i} =
        cellstr(strvcat(strsplit(erasePunctuation(lower(fileread(d(i).name))))));
    total_words = total_words + numel(txt{i});
end

total_words
```

```
total_words =  
  
205029
```

b(ii) Total distinct (unique) words across n essays

```
total_txt = cat(2, txt{:});  
[total_distinct_txt, b, c] = unique(total_txt);  
total_distinct_txt_count = hist(c, length(total_distinct_txt));  
  
total_distinct_words = numel(total_distinct_txt);  
  
total_distinct_words  
  
total_distinct_words =  
  
11236
```

(c) 20 most frequent words and their counts

```
[decreasing_val decreasing_ind] =  
sort(total_distinct_txt_count, 'descend');  
  
total_distinct_txt{decreasing_ind(1:20)};  
twenty_most_frequent_counts = decreasing_val(1:20);  
  
for i = 1:20  
    fprintf('%s: ', total_distinct_txt{decreasing_ind(i)});  
    fprintf('%d \n', decreasing_val(i));  
end  
  
fprintf('\n\n');  
  
the: 11816  
to: 6782  
of: 6152  
and: 5252  
in: 4411  
a: 3892  
that: 3431  
is: 3373  
ai: 3105  
be: 2696  
will: 2056  
it: 1952  
for: 1857
```

```
as: 1699
are: 1632
on: 1575
not: 1550
this: 1437
with: 1359
intelligence: 1181
```

(d) 20 least frequent words and their counts

```
[increasing_val increasing_ind] =
    sort(total_distinct_txt_count, 'ascend');

total_distinct_txt{increasing_ind(1:20)};
twenty_least_frequent_counts = increasing_val(1:20);

for i = 1:20
    fprintf('%s: ', total_distinct_txt{increasing_ind(i)});
    fprintf('%d \n', increasing_val(i));
end

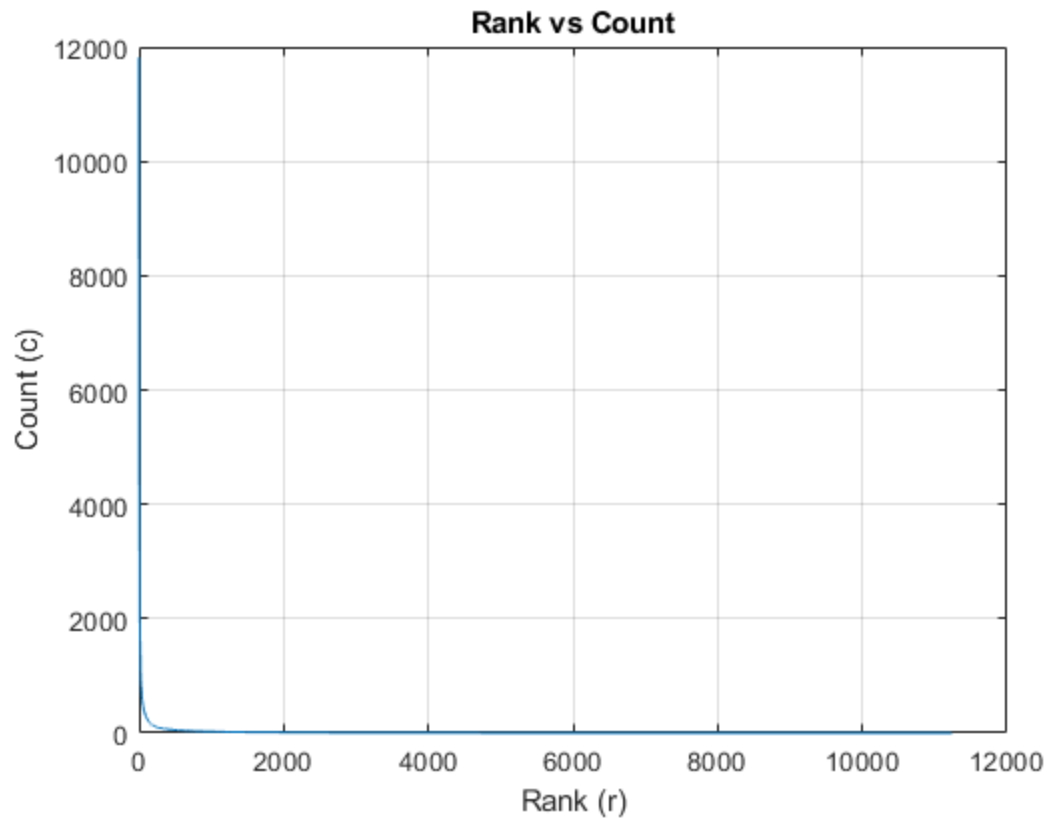
fprintf('\n\n');
```

```
01: 1
02: 1
03: 1
05: 1
09092017: 1
09132017: 1
10000: 1
100000: 1
1000mile: 1
101007s1067701697452: 1
101016japergo201509009: 1
1020: 1
104: 1
107: 1
111: 1
1119: 1
11717: 1
11th: 1
126: 1
12751289: 1
```

(e) Plot rank r vs count c

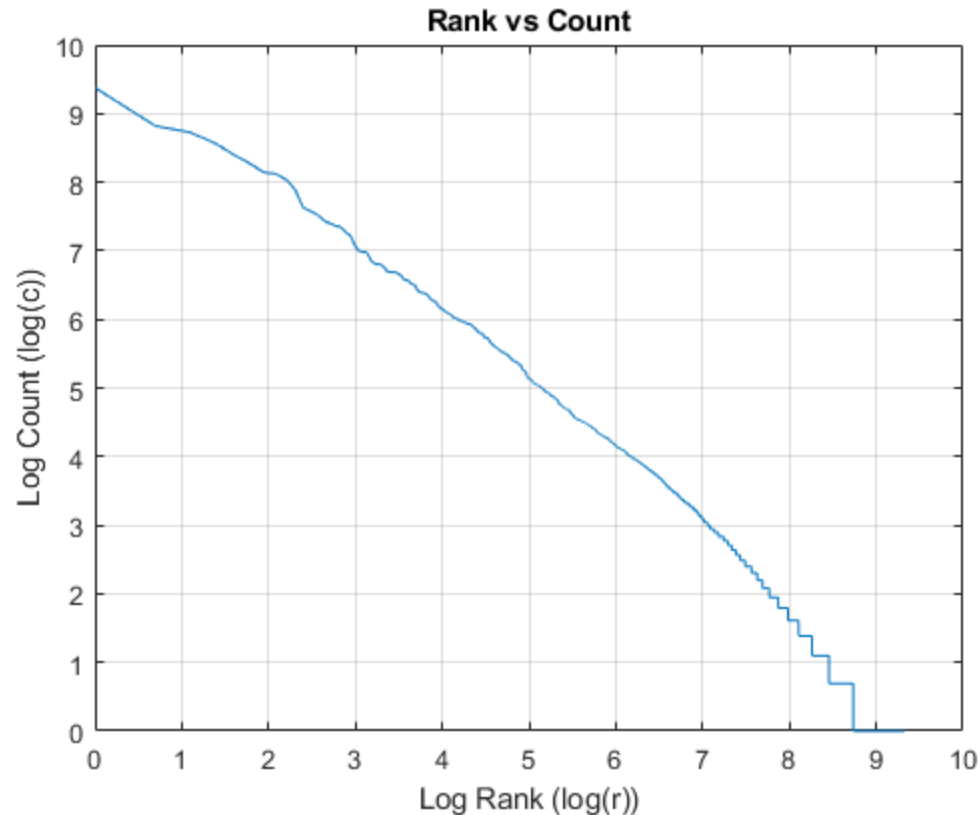
```
figure()
plot(decreasing_val)
```

```
grid
xlabel('Rank (r)');
ylabel('Count (c)');
title('Rank vs Count');
```



(f) Plot $\log(r)$ vs $\log(c)$

```
figure()
x = log(1:length(total_distinct_txt_count));
y = log(decreasing_val);
plot(x, y)
grid
xlabel('Log Rank (log(r))');
ylabel('Log Count (log(c))');
title('Rank vs Count');
```



(g) Explain the shapes of the curves

The plot in part (e) shows how much we humans rely heavily on the same words (at least in English). It is exponentially falling.

```
% The plot in part (f) shows the logarithmic plot, and it brings some  
credance  
% to the fact that the previous plot is exponential, because this plot  
is almost  
% linear.
```

(h) Two potential issues with this method of preprocessing

I didn't account for misspellings, so those words were counted as unique words, instead of being corrected to words that were already in our list. It also did not count aliasing between words with equivalent meanings but different representations, such as between "eleventh" and "11th". Also, simply considering each number as a word would probably just add noise to our data, while giving us little to no new information, so they should probably be removed.

```
% Besides all this, there was also the issue of some instances of a  
word type  
% being lost due to the typeset of my OS, so the counts of some words  
were also
```

% off.

Published with MATLAB® R2017b