

LING/C SC/PSYC 438/538

Lecture 12

Sandiway Fong

Today's Topics

- Homework 8 (*previewed on Tuesday, due Sunday midnight*)
- More on Perl regex:
 - Perl code inside a regex!
 - Character and word frequency counting
- Zipf's Law
 - Brown Corpus case study

Pandora Papers



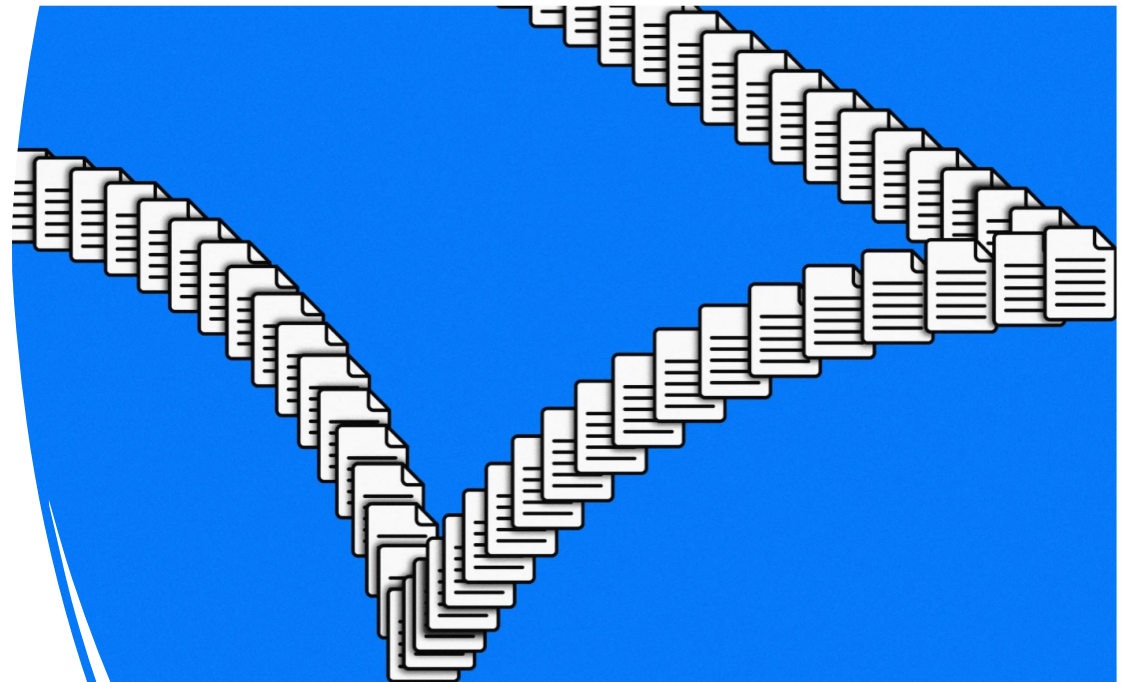
- <https://www.icij.org/investigations/pandora-papers/>
- An investigation by more than 600 journalists from 150 news outlets has unearthed offshore dealings of 35 current and former world leaders and more than 300 other current and former public officials and politicians around the world.
- The offshore system continues to thrive despite decades of legislation, investigations and international agreements aimed at combating money laundering and tax dodging.
- South Dakota and more than a dozen U.S. states have become leaders in the business of selling financial secrecy — even as the U.S. blames smaller nations for enabling tax avoidance and dirty money flows.
- The Pandora Papers unmask the hidden owners of offshore companies, secret bank accounts, private jets, yachts, mansions and artworks by Picasso, Banksy and other masters.
- The biggest leak in ICIJ history contains 2.94 terabytes of confidential information from 14 offshore service providers

the secrets of the Pandora Papers

ra Papers leak includes a colossal 2.94 terabytes of data. Unravelling the conten

Wired Article

- <https://www.wired.co.uk/article/pandora-papers-leak>
- In addition, the Pandora Papers included a broader range of file types and formatting that the machine learning systems the ICIJ previously used had to learn about to be able to parse and identify in order to be able to sort. “It’s now able to read very specific financial documents and very specific PDFs.”



Homework 8

- Background:
 - *"The Pandora Papers is a leak exposing the secret wealth and dealings of world leaders, politicians and billionaires."*
- Text file (utf-8): `pandora.txt`
- `wc pandora.txt`
- `355 6239 39968 pandora.txt`
- Let's datamine this for *named entities* using Perl regex and see who/what we find!

Homework 8

- What are named entities (NE)?
 - *person, organization, place name, time expression, monetary value, etc.*
- Recall previous lecture slide, we have:
 - `perl -le 'open $f, "pandora.txt"; while (<$f>) {while (/regex/g) {print $&}}'`

Homework 8

- **Question 1a:**
 - in English, names typically begin with an Upper case letter. Other characters may be lower/upper case or include a hyphen/dash (-), e.g. *ABC-CDE*.
 - Write a regex and find all the matching **words** in the article. How many are there?
- **Question 1b:** last lecture we mentioned use of
 - `open qw(:std :utf8);`
 - Find the differences in the words reported when running your code with this declaration.
 - **Hint:** you may want to think about `[A-Za-z-]` vs `[\w-]`
- **Question 1c:**
 - do all name words begin with an Upper case letter? Find two that don't.

Homework 8

- **Question 2:**

- abbreviations/acronyms often consist of words, #letters ≥ 2 , containing only Upper case letters, possibly with periods separating them,
- e.g. TV, US, U.S., TASS.
- Write a regex for this. How many are there?

- **Question 3:**

- many named entities are n -grams, $n \geq 2$, a sequence of words:
 - e.g. Al Mawarid Bank, British Prime Minister Tony Blair
- each beginning with an Upper case letter, **optionally** beginning with a title with leading capitalization:
 - e.g. Mr, Mrs, Ms, Mx, Dr, (Prime) Minister, President or King/Queen (of).
 - e.g. King of Jordan
- Write a regex and find all the matching sequences ($\#words \geq 2$). Print them. How many are there?

Homework 8

Following is optional for 438, mandatory for 538.

- **Question 4:**

- using the Perl hash table described in a previous lecture, re-do **Question 3** and collect together mentions of named entities, e.g. Baker McKenzie occurs multiple times. Then print names and number of occurrences in tabular form, e.g.

Homework 8

- Usual rules:
 - due Sunday midnight
 - email to me
 - 438/538 Homework 8 *YOUR NAME*
 - one PDF file please
 - you can attach code separately if you like, but code should ALSO appear in PDF file!

s/regex/replace/e

Substitution-specific modifiers described in "[s/PATTERN/REPLACEMENT/msixpodualngcer](#)" in `perlop` are:

```
e - evaluate the right-hand side as an expression
ee - evaluate the right side as a string then eval the result
o - pretend to optimize your code, but actually introduce bugs
r - perform non-destructive substitution and return the new value
```

A `/e` will cause the replacement portion to be treated as a full-fledged Perl expression and evaluated right then and there. It is, however, syntax checked at compile-time. A second `e` modifier will cause the replacement portion to be evaluated before being run as a Perl expression.

s/regex/replace/e

- # Add one to the value of any numbers in the string
- `s/(\d+)/1 + $1/eg;`
- One-liner:

```
perl -le '$_ = qq/@ARGV/; s/(\d+)/1 + $1/eg; print' 123  
234
```

```
124 235
```

s/regex/replace/ee

- The period is a string concatenation operator (cf. Python +):

```
perl -le '$_=shift; s/((\d+)(\d+))/\d2.\d1/;print' '3*4'  
3.3*4
```

```
perl -le '$_=shift; s/((\d+)(\d+))/\d2.\d1/e;print' '3*4'  
33*4
```

```
perl -le '$_=shift; s/((\d+)(\d+))/\d2.\d1/ee;print' '3*4'  
132
```

Character Frequency Counting

- Sample code is rather interesting (e flag):

This example counts character frequencies in a line:

```
1. $x = "Bill the cat";
2. $x =~ s/(.)/$chars{$1}++;$1/eg; # final $1 replaces char with itself
3. print "frequency of '$_' is $chars{$_}\n"
4.   foreach (sort {$chars{$b} <=> $chars{$a}} keys %chars);
```

s/regex/replace/e

- Generally, in regex: (?{ Perl code })

- Slightly modified but easier to read:

```
1 $x= "This is a slightly simplified version of a rather complicated piece of Perl code.";
2 $x =~ s/(.)/$chars{lc($1)}++;$1/eg;
3 @sorted = sort {$chars{$b} <=> $chars{$a}} keys %chars;
4 print $x, "\n";
5 foreach $key (@sorted) {
6     print "freq of $key is $chars{$key}\n"
7 }
```

note: lowercase

Character Frequency Counting

```
$ perl -le '$_ = qq/@ARGV/; s/(.)/${c{$1}}++/eg; foreach $k (sort {${c{$b}} <=> ${c{$a}}} keys %c) {print "$k:${c{$k}} "}' this is a sentence to test this program.
```

```
:7  
t:6  
s:5  
e:4  
i:3  
r:2  
h:2  
a:2  
n:2  
o:2  
g:1  
m:1  
.:1  
p:1  
c:1
```

Steps:

1. `$_ = qq/@ARGV/` put the command line arguments into a string `$_`
2. globally match each character `(.)` and increment a hash table `(%C)` count
3. for each key `$k` in `%C` (sorted in descending order by value), print `key:value`

Character Frequency Counting

```
perl -ne 's/(.)/${lc($1)}++/eg' END {foreach $k (sort {$c{$b} <=> $c{$a}} keys %c) {print "$k:$c{$k} "}}
print "\n"} 12thnight.txt
```

Implicit Loops

Two other command-line options, `-n` processing files a line at a time. If you

```
$ perl -n -e 'some code' file1
```

Then Perl will interpret that as:

```
LINE:
  while (<>) {
    # your code goes here
  }
```

Note:

- `$_` will contain each line of `file1`
- *loop code* `END` *post-loop code*

12thnight.txt

```
1 If music be the food of love, play on;¶
2 Give me excess of it, that, surfeiting,¶
3 The appetite may sicken, and so die.¶
4 That strain again! it had a dying fall:¶
5 O, it came o'er my ear like the sweet sound,¶
6 That breathes upon a bank of violets,¶
7 Stealing and giving odour! Enough; no more:¶
8 'Tis not so sweet now as it was before.¶
9 O spirit of love! how quick and fresh art thou,¶
10 That, notwithstanding thy capacity¶
11 Receiveth as the sea, nought enters there,¶
12 Of what validity and pitch soe'er,¶
13 But falls into abatement and low price,¶
14 Even in a minute: so full of shapes is fancy¶
15 That it alone is high fantastical.¶
```


Character Frequency Counting

12th Night:

```
perl -ne '{s/(.)/$c{lc($1)}++/eg} END {foreach  
$k (sort {$c{$b} <=> $c{$a}} keys %c) {print  
"$k:$c{$k} "}} print "\n"}' 12thnight.txt
```

```
 :99 t:52 e:51 a:45 i:40 o:36 n:33 s:31 h:25  
f:16 l:16 r:16 ,:14 d:13 u:12 c:12 g:11 p:9 w:8  
m:8 v:8 y:8 b:6 k:4 ':3 .:3 !:3 ::3 ;:2 x:1 q:1
```

(?{ Perl code })

```
1. $x = "abcdef";
2. $x =~ /abc(?{print "Hi Mom!";})def/; # matches,
3.                                     # prints 'Hi Mom!'
4. $x =~ /aaa(?{print "Hi Mom!";})def/; # doesn't match,
5.                                     # no 'Hi Mom!'
```

Pay careful attention to the next example:

```
1. $x =~ /abc(?{print "Hi Mom!";})ddd/; # doesn't match,
2.                                     # no 'Hi Mom!'
3.                                     # but why not?
```

Perl regex
optimization

At first glance, you'd think that it shouldn't print, because obviously the `ddd` isn't going to match the target string. But look at this example:

```
1. $x =~ /abc(?{print "Hi Mom!";})[dD]dd/; # doesn't match,
2.                                     # but _does_ print
```

No Perl regex
optimization

(?{ Perl code })

- One-liner:

```
[~$ perl -le '"abcdef" =~ /abc(?{print "Hi!"})ddd/'  
[~$ perl -le '"abcdef" =~ /abc(?{print "Hi!"})[dD]dd/'  
Hi!  
~$ █
```

Word Frequency Counting

- Words, including punctuation: `\b(\S+?)\b`

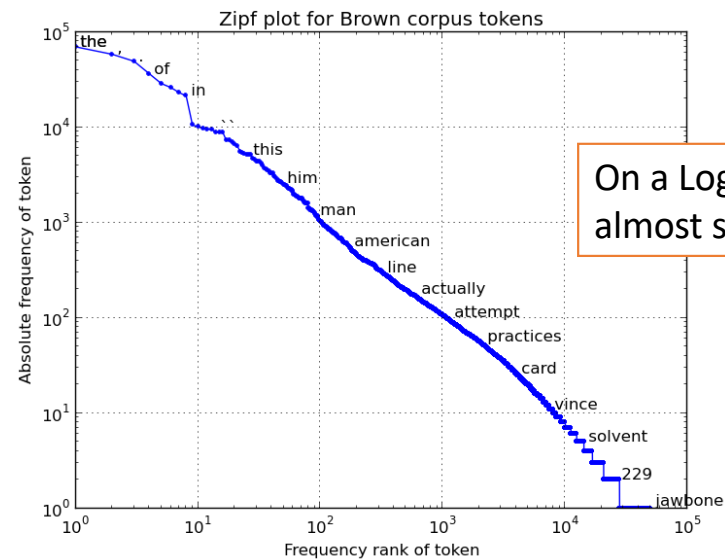
```
$ perl -ne '{s/\b(\S+?)\b/$c{lc($1)}++;/eg} END {foreach $k (sort {$c{$b} <=> $c{$a}} keys %c) {print "$k:$c{$k} "; print "\n"} ' 12thnight.txt
of:6 and:5 that:5 it:5 the:4 o:3 a:3 so:3 love:2 sweet:2 er:2 as:2 ':2 is:2 wa
s:1 ear:1 sea:1 give:1 excess:1 shapes:1 notwithstanding:1 spirit:1 thou:1 sou
nd:1 quick:1 me:1 art:1 die:1 falls:1 there:1 pitch:1 like:1 came:1 before:1 t
is:1 minute:1 no:1 had:1 how:1 soe:1 sicken:1 even:1 fresh:1 in:1 validity:1 b
ank:1 high:1 full:1 odour:1 now:1 enough:1 music:1 into:1 violets:1 again:1 st
rain:1 dying:1 food:1 nought:1 more:1 fall:1 breathes:1 receiveth:1 low:1 fanc
y:1 what:1 be:1 fantastical:1 thy:1 not:1 enters:1 upon:1 giving:1 play:1 alon
e:1 stealing:1 if:1 may:1 but:1 capacity:1 abatement:1 price:1 on:1 surfeiting
:1 appetite:1 my:1
$ █
```

Zipf's Law

- Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table.
- See:
 - <http://demonstrations.wolfram.com/ZipfsLawAppliedToWordAndLetterFrequencies/>
 - Brown Corpus (1,015,945 words): only 135 words are needed to account for half the corpus.

	Word	Instances	% Frequency
1.	The	69970	6.8872
2.	of	36410	3.5839
3.	and	28854	2.8401
4.	to	26154	2.5744
5.	a	23363	2.2996
6.	in	21345	2.1010
7.	that	10594	1.0428

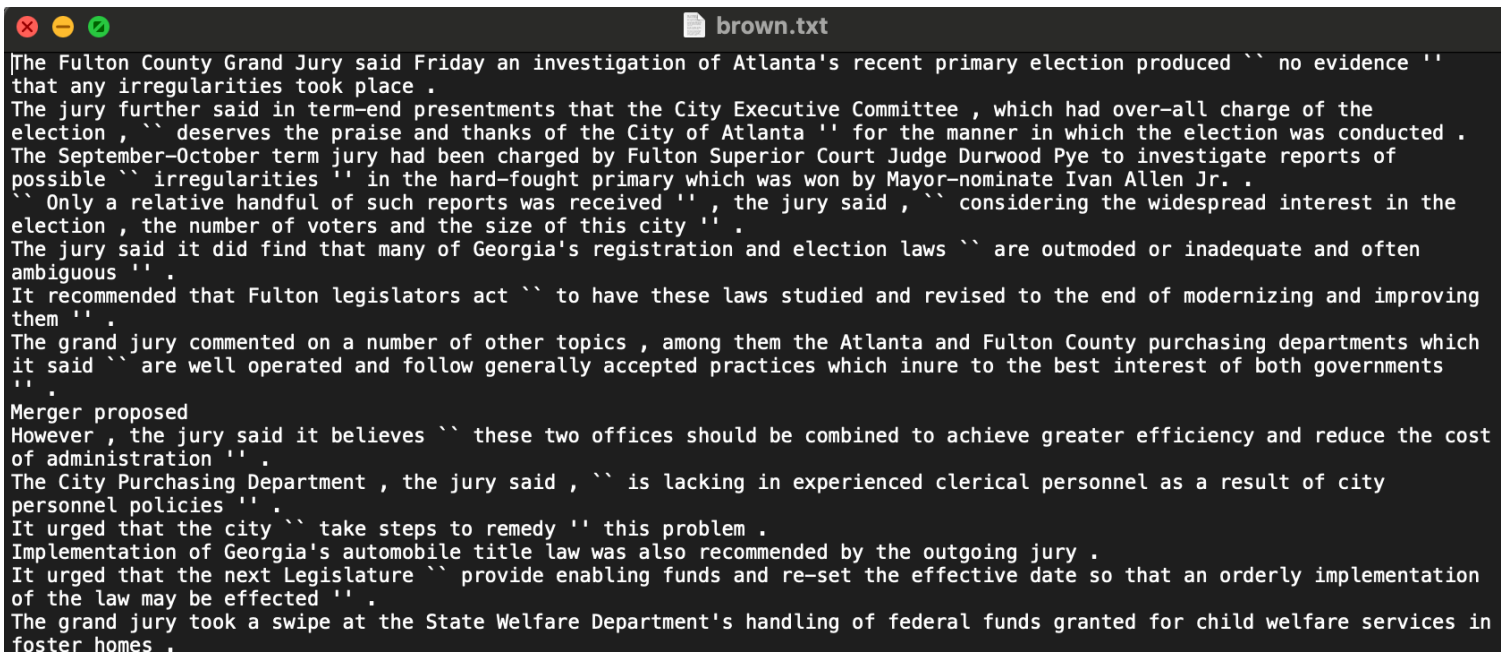
<http://www.learnholistically.it/esp-clil/wfk2.htm>



<https://finnaarupnielsen.wordpress.com/2013/10/22/zipf-plot-for-word-counts-in-brown-corpus/>

Brown Corpus

- https://en.wikipedia.org/wiki/Brown_Corpus
 - "It contains 500 samples of English-language text, totaling roughly one million words, compiled from works published in the United States in 1961."



```

The Fulton County Grand Jury said Friday an investigation of Atlanta's recent primary election produced `` no evidence '' that any irregularities took place .
The jury further said in term-end presentments that the City Executive Committee , which had over-all charge of the election , `` deserves the praise and thanks of the City of Atlanta '' for the manner in which the election was conducted .
The September-October term jury had been charged by Fulton Superior Court Judge Durwood Pye to investigate reports of possible `` irregularities '' in the hard-fought primary which was won by Mayor-nominate Ivan Allen Jr. .
`` Only a relative handful of such reports was received '' , the jury said , `` considering the widespread interest in the election , the number of voters and the size of this city '' .
The jury said it did find that many of Georgia's registration and election laws `` are outmoded or inadequate and often ambiguous '' .
It recommended that Fulton legislators act `` to have these laws studied and revised to the end of modernizing and improving them '' .
The grand jury commented on a number of other topics , among them the Atlanta and Fulton County purchasing departments which it said `` are well operated and follow generally accepted practices which inure to the best interest of both governments '' .
Merger proposed
However , the jury said it believes `` these two offices should be combined to achieve greater efficiency and reduce the cost of administration '' .
The City Purchasing Department , the jury said , `` is lacking in experienced clerical personnel as a result of city personnel policies '' .
It urged that the city `` take steps to remedy '' this problem .
Implementation of Georgia's automobile title law was also recommended by the outgoing jury .
It urged that the next Legislature `` provide enabling funds and re-set the effective date so that an orderly implementation of the law may be effected '' .
The grand jury took a swipe at the State Welfare Department's handling of federal funds granted for child welfare services in foster homes .

```

Brown Corpus

- brown.txt (lines taken from brown.sents() from nltk.corpus)

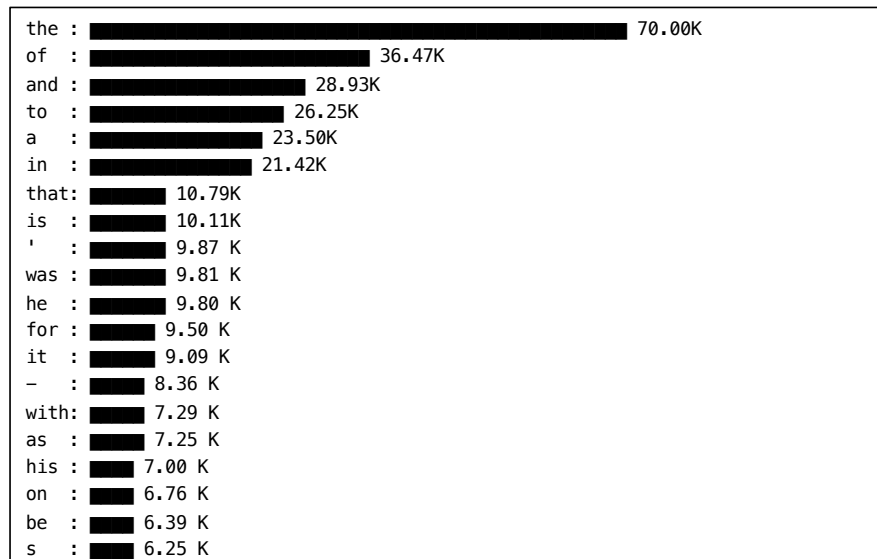
```
perl -ne '{s/\b(\S+)\b/${lc($1)}++;/eg} END {@s = sort {$c{$b} <=>
$c{$a}} keys %c; foreach $k (@s[0..99]) { print "$k:$c{$k} " } }'
brown.txt
```

```
the:70003 of:36473 and:28935 to:26247 a:23502 in:21422 that:10789
is:10109 ':9865 was:9815 he:9801 for:9500 it:9094 -:8355 with:7290
as:7255 his:6999 on:6765 be:6388 s:6250 i:5932 at:5377 by:5346 this:5146
had:5133 not:4620 are:4394 but:4382 from:4371 or:4226 have:3942
they:3763 an:3751 you:3634 which:3561 one:3504 were:3285 all:3099
her:3036 she:2987 we:2844 there:2844 would:2719 their:2670 him:2619
been:2472 has:2437 when:2331 who:2280 will:2251 t:2246 more:2225 no:2219
if:2198 out:2167 so:2033 up:1974 what:1968 said:1961 can:1942 its:1858
about:1817 than:1796 into:1791 them:1790 only:1748 other:1714 time:1695
new:1646 some:1618 could:1602 these:1573 two:1516 may:1402 first:1389
then:1380 do:1375 man:1364 any:1344 like:1343 my:1319 now:1317 over:1307
such:1303 our:1253 .:1209 me:1185 even:1173 most:1162 made:1147
after:1077 also:1069 did:1044 many:1037 before:1016 must:1015 well:1006
af:1005 back:976 through:974
```

	Word	Instances	% Frequency
1.	The	69970	6.8872
2.	of	36410	3.5839
3.	and	28854	2.8401
4.	to	26154	2.5744
5.	a	23363	2.2996
6.	in	21345	2.1010
7.	that	10594	1.0428

Brown Corpus

- Using termgraph:
 - pip3 install termgraph (Python)
 - termgraph [datafile] (two columns: #1 label, #2 value)
- `perl -ne '{s/\b(\S+)\b/$c{lc($1)}++;/eg} END {@s = sort {$c{$b} <=> $c{$a}} keys %c; foreach $k (@s[0..19]) { print "$k $c{$k}\n" } }'` brown.txt | [termgraph](#)



Brown Corpus

from the earlier slide:

- <http://demonstrations.wolfram.com/ZipfsLawAppliedToWordAndLetterFrequencies/>
- Brown Corpus (1,015,945 words): only 135 words are needed to account for half the corpus.

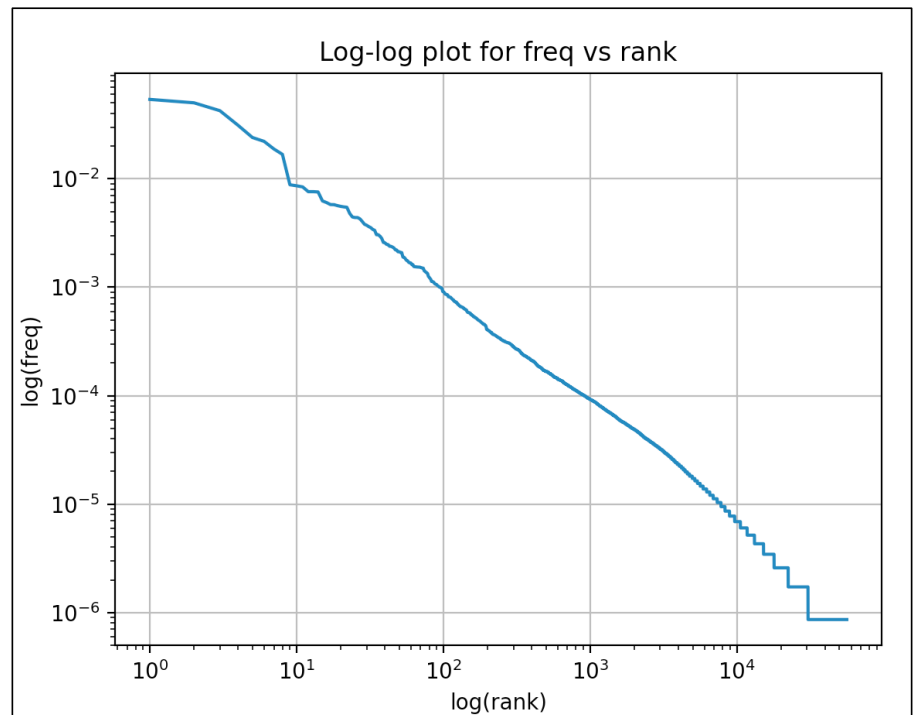
```
perl -ne '{s/\b(\S+)\b/$c{lc($1)}++;$t++;/eg} END {$t /=2; foreach $k (sort  
{$c{$b}<=>$c{$a}} keys %c) {$n++; $t -= $c{$k}; last if ($t<0) }; print "$n\n"}'  
brown.txt  
122
```

<code>last</code>	means terminate the running of foreach loop early
<code>\$t</code>	total number of words, halved by <code>\$t /= 2</code>
<code>\$n</code>	to count the number of different words (<code>\$n++</code>), stop when <code>\$t<0</code>

Brown Corpus

- In nltk (Natural Language Toolkit):

```
python -i zipf.py  
>>> import nltk  
>>> from nltk.corpus import  
brown  
>>> plot(brown.words())
```



Brown Corpus

```
plot([w.lower() for w in brown.words()])
```

