Foundations of Statistical Natural Language Processing
Notes and Exercises

Chris Merck

June 30, 2011

1 Introduction

Notes

- (1.4.1) A word X is a *meronym* of a word Y if X is a part of Y. From Greek *mero* 'part'. Also, opposite *holonym*. Compare *hyponym* and *hypernym* which describe a subset/superset relationship.
- (1.4.5) *nonagentive* = ?
- (1.4.5) *ditransitive* = taking two direct objects

Exercises

1.1 Noncategoricallity

In (1.19a) *fun* is used as a noun, and in (1.19b) we see the superlative form *funnest* showing that *fun* can also be an adjective. Here I propose that the word simply belongs to both the noun and adjective category. In (1.20a) *email* is used as an uncountable noun, just like *mail*, but in (1.20b) the word appears in the plural and therefore is countable. Note however that **mails* is certainly unacceptable. Again, here we can simply assign *email* to both the uncountable and countable categories, on the basis of observation. I see no technically noncategorical issue here either; however, I would propose that the membership of *email* to each category be associated with a sort of weight, indicating how common, acceptable, or old the usage is. Compare to fuzzy sets.

1.2 Dirty Hands

See python2 program called 'DirtyHands.py' posted to blog. The program reproduces the tables 1.1, 1.2, and 1.3 with good agreement, and obtains a resonably straight log-log plot for Zipf's law (see Figure 1).

1.3 Mandelbrot generalizes Zipf

Mandelbrots law is

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

Substitute $\rho = 0$ and $B = 1$ to obtain:

$$f = Pr^{-1} = P/r,$$

which is a statement of Zipf's law.

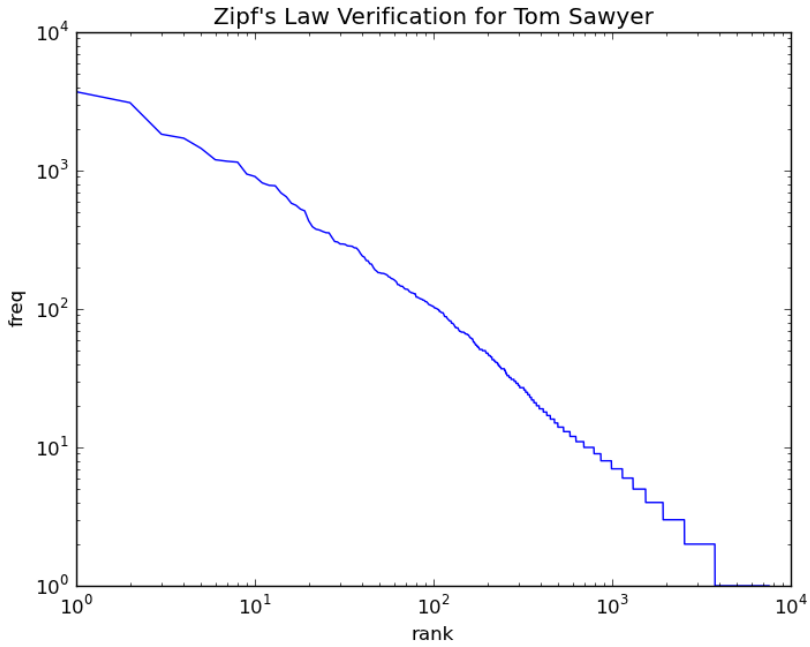

Figure 1: Tom Sawyer shows good agreement with Zipf's law.

1.4 Emperical Zipf on Random Text

See python2 program called 'EmpericalZipf.py' posted to blog.

The results for the 26-character alphabet, even when using a million character random text, are not so good (see Figure 2). However, when I reduce the number of characters in the alphabet to three, we obtain a much closer approximation of a straight line in the freq-vs-rank log-log plot (see Fig 3).

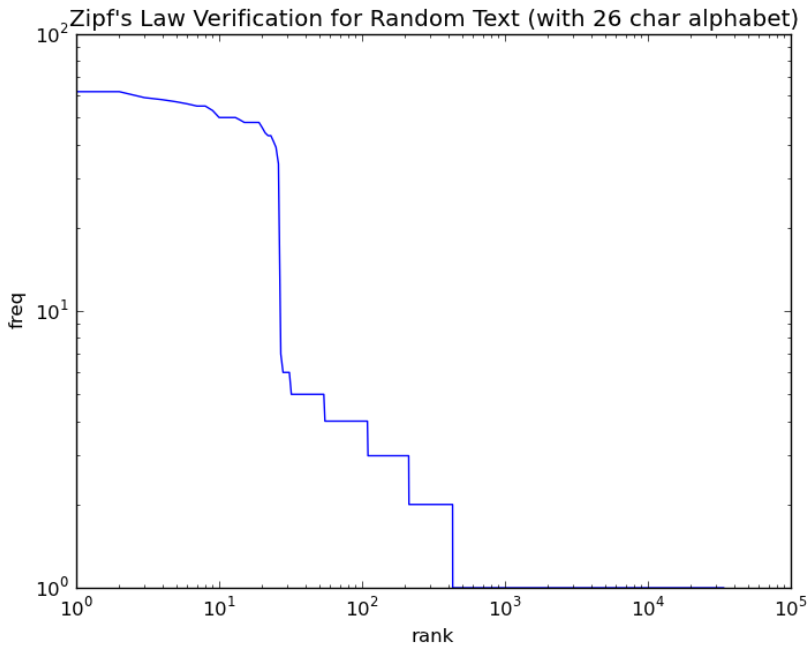

Figure 2: 26-character alphabet random text shows poor Zipf's law performance.

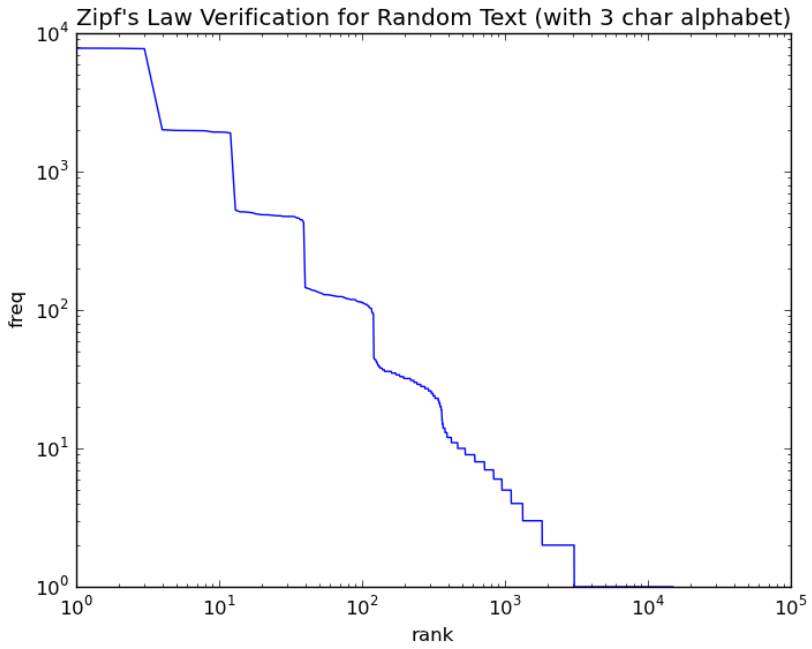

Figure 3: Testing Zipf's law with 3-charicter alphabet looks much better.

1.5 Better Collocation Identification

To improve the identification of common collocations within a corpus, I propose modifying the score function to take into account the frequencies of the constituant words. Let f_i denote the frequency of the ith word and b_i denote the frequency of the ith bigram. In the context of the ith bigram, f_i and f_{i+1} are the frequencies of the constituant words. I propose dividing b_i by one of the following denominators:

$$f_i f_j,\ f_i + f_j,\ \sqrt{f_i, f_j},\ \max(f_i, f_j).$$

1.6 Applied Improved Collocation ID

See the python2 program called 'Collocations.py'.

I found that the denominators I proposed above do not work well as given. They so heavily weight bigrams containing rare words that the only results I get when running them on Tom Sawyer are bigrams with a frequency of one.

However, when the denominator is raised to a fractional power, we do obtain interesting results. Below are given the results. These are the top ten most 'interesting' collocations using several different denominators, where the exponent has been chosen large enough to reject 'of the' and similarly uninteresting bigrams from the top ten.

$(f_1 + f_2)^{.8}$	$(f_1 * f_2)^{.4}$	$max(f_1, f_2)^{.75}$	$min(f_1, f_2)^{.95}$
aunt polly	aunt polly	robin hood	aunt polly
robin hood	robin hood	aunt polly	sort of
st petersburg	st petersburg	st petersburg	a couple
bob tanner	bob tanner	bob tanner	couple of
united states	united states	united states	aunt polly's
ole missis	ole missis	ole missis	the widow's
corn pone	injun joe	corn pone	i dono
per cent	corn pone	per cent	cardiff hill
uncle jake	per cent	uncle jake	his aunt's
unavailing regrets	uncle jake	unavailing regrets	i am

Note the nearly identical behavior of the first three formulae.

1.7 Concordancing Program

See python2 program called 'kwic.py' posted to blog.

Example output:

```
[cmerck@caladan ch1]$ python2 -i kwic.py show
1   -- " "And besides, if you will I'll  show  you my sore toe." Jim was only huma
2    know she was present, and began to "show  off" in all sorts of absurd boyish
3    downhearted, but tried hard not to  show  it. He had a secret which he was no
4   bring it out. He said, with a great  show  of cheerfulness: "I bet there's bee
5   here was no other way; so with such  show  of cheerfulness as they could muste
6   all the boys' kites and things, and  show  'em where the good fishin' places w
7   ng. For a little while, hope made a  show  of reviving -- not with any reason
8   sensation in his throat, and made a  show  of being confident of finding the s
9   feet high. Tom whispered: "Now I'll  show  you something, Huck." He held his c
10  wever, the widow made a pretty fair  show  of astonishment, and heaped so many
11  you needn't smile -- I reckon I can  show  you. You just wait a minute." Tom r
```

References

[1] Manning and Schuetze, *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge (1999).