



probability assigned to the unseen objects. These probabilities for the observed objects have not been easy to estimate, and may have deterred many from using Good-Turing methods. It is a major point of this paper that a simple method for treating the observed objects gives better accuracy than several other methods. In particular, the paper will compare the simple Good-Turing (SGT) method proposed to adding one half to each observation (ELE method), and to two way cross validation.

The paper aims to explain what the Good-Turing method is about in an intuitive fashion, and to show a simple method for treating the observed objects. Four examples will be used to illustrate the various points.

## **2. What Good-Turing Methods are About**

Consider the following example taken from some ongoing prosody work with Joan Bachenko. An input









**Figure 2**  
**The Prosody data need the averaging transform**

Nr, frequency of frequency

r, frequency





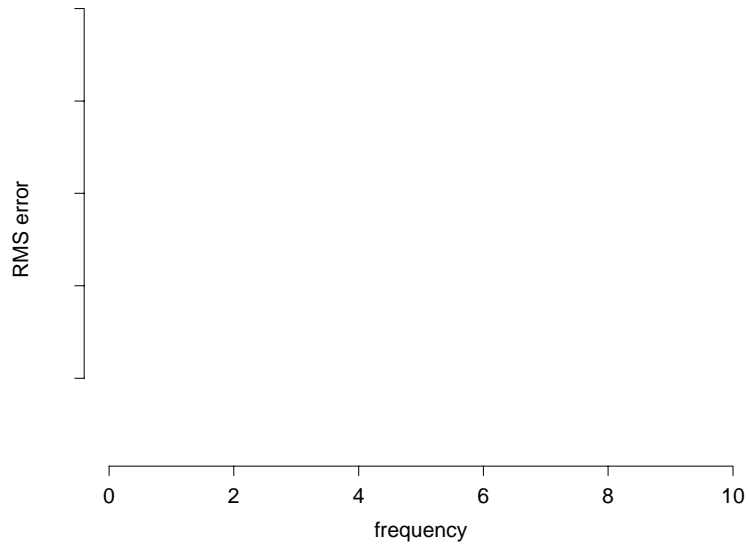
which will generate data of the sort that one wants to apply a method to, generating data by using the model and a (pseudo-)random number generator, applying the methods to be tested to the data, and comparing the results of the methods to the truth known from constructing the model. The utility of the results is related most closely to the validity of the model used to generate the artificial data.

For this study, we constructed twenty texts each having 100,000 tokens. The tokens drawn randomly from a vocabulary of size  $V$  types, with the probability of the  $i$

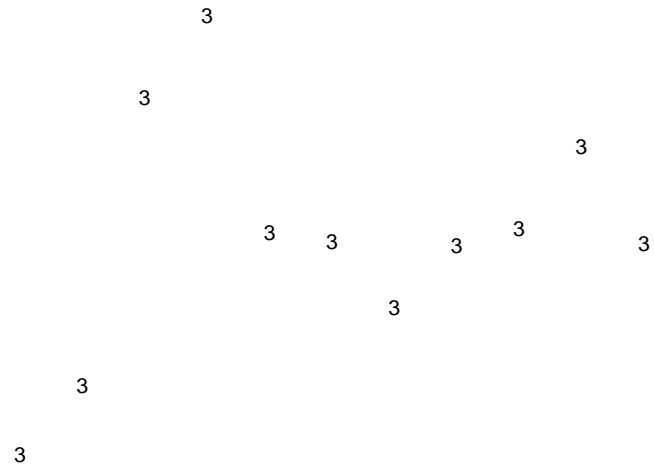




**Figure 5**  
**Two Way Cross Validation is Poor for Small Frequencies**



**Figure 6**  
**Simple Good Turing Errors are Least for Unseen Objects**



**Figure 7**  
**Simple Good-Turing Accuracy is Independent of Vocabulary Size**

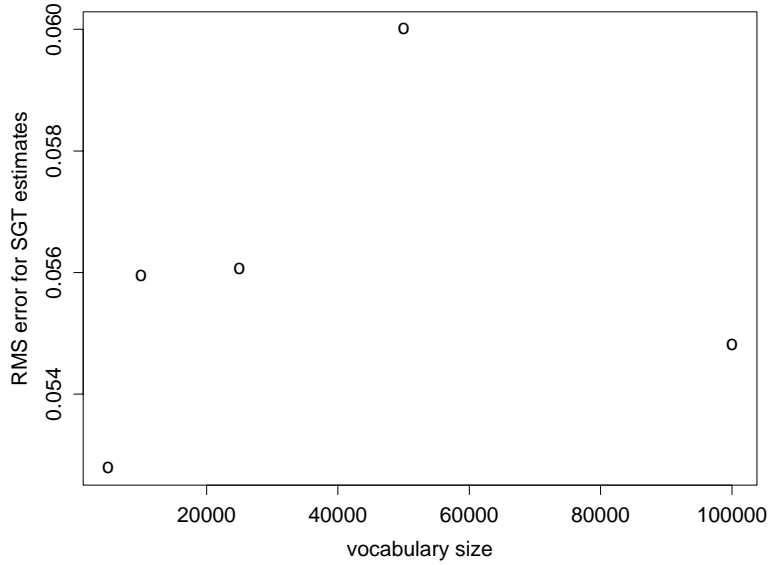
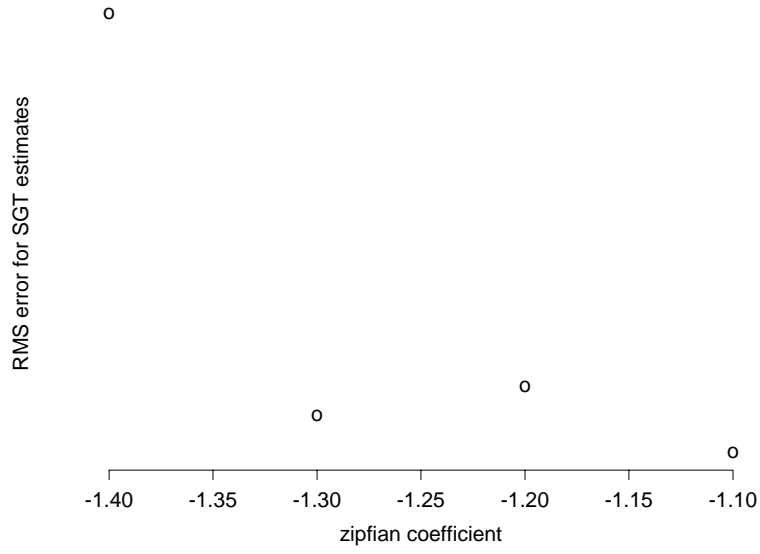


Figure 7. Vocabulary size is plotted horizontally, while the vertical axis shows RMS errors for the Simple Good-Turing estimates. There is no systematic dependence on vocabulary size, and the range of variation is small (note that the range of the vertical axis is a small part of the range in Figure 6.)

The figure shows little dependence on vocabulary size, and over the range from 5000 types to 100000 types for 100000 token texts. Furthermore, the range of RMS errors shown here is much less than that shown for dependence on vocabulary size in the previous figure. The following figure plots RMS errors for the SGT estimates by Zipfian exponent.

**Figure 8**  
**Simple Good-Turing Accuracy is Independent of Zipfian Coefficient**



**Table 6**  
**SGT Estimation is Accurate for Bigram Example**



*References*

1. Becker, R., J. Chambers, and A. Wilks, (1988), *The New S Language*, Wadsworth & Brooks/Cole, Pacific Grove, California.
2. Box, G. and G. Tiao, (1973),

*Appendix I*

This appendix contains the complete set of  $(r, N_r)$  pairs for the pp.5/dy and Chinese plurals examples.

**Pp.5/dy**

$r$	$N_r$
1	120
2	40
3	24
4	13
5	15
6	5
7	11
8	2
9	2
10	1
12	3
14	2
15	1
16	1
17	3
19	1
20	3
21	2
23	3
24	3
25	3
26	2
27	2
28	1
31	2
32	2
33	1
34	2
36	2
41	3
43	1
45	3
46	1
47	1
50	1
71	1
84	1
101	1
105	1
121	1
124	1
146	1
162	1
193	1

199	1
224	1
226	1
254	1
257	1
339	1
421	1
456	1
481	1
483	1
1140	1
1256	1
1322	1
1530	1
2131	1
2395	1
6925	1
7846	1

**Chinese Plurals**

$r$	$N_r$
1	268
2	112
3	70
4	41
5	24
6	14
7	15
8	14
9	8
10	11
11	9
12	6
13	6
14	3
15	7
16	9
17	4
18	4
19	8
20	2
21	4
22	2
23	2
24	3
25	4
26	4
27	4
28	1
29	1



*Appendix II*

```
xN<-sum(xr*xnr)

#make averaging transform
xnrz<-nrzest(xr,xnr)

#get Linear Good-Turing estimate
xf<-lsfit(log(xr),log(xnrz))
xcoef<-xf$coef
xrst<-rstest(xr,xcoef)
xrstrel<-xrst/xr

#get Turing estimate
xrtry<-xr==c(xr[-1]-1,0)
xrstarel<-rep(0,length(xr))
xrstarel[xrtry]<-(xr[xrtry]+1)/xr[xrtry]*c(xnr[-1],0)[xrtry]/xnr[xrtry]

#make switch from Turing to LGT estimates
tursd<-rep(1,length(xr))
for(i in 1:length(xr))if(xrtry[i])
  tursd[i]<-(i+1)/xnr[i]*sqrt(xnr[i+1]*(1+xnr[i+1]/xnr[i]))
xrstcmbrel<-rep(0,length(xr))
useturing<-TRUE
for(r in 1:length(xr)){
  if(!useturing) xrstcmbrel[r]<-xrstrel[r]
  else if(abs(xrstrel-xrstarel)[r]*r/tursd[r] > 1.65)
    xrstcmbrel[r]<-xrstarel[r]
  else {useturing<-FALSE; xrstcmbrel[r]<-xrstrel[r]}
}

#renormalize the probabilities for observed objects
sumpraw<-sum(xrstcmbrel*xr*xnr/xN)
xrstcmbrel<-xrstcmbrel*(1-xnr[1]/xN)/sumpraw
```