LING439/539 - Statistical NLP
Chapter 6. Statistical inference: n-gram models over sparse data (continued)

Tuesday, September 13 2016

Good-Turing discounting (Recap)

N_c = the number of N-grams that occur c times
$\rightarrow$ frequency of frequency c.

- N_0: the number of bigrams with count 0.
- N_1: the number of bigrams with count 1 (hapax).
- ...

The Good-Turing intuition is to estimate the probability of things that occur c times in the training corpus by the MLE probability of things that occur $c+1$ times in the corpus.

Smoothed (or adjusted) count $\quad c^* = (c+1)\dfrac{N_{c+1}}{N_c}$

—

The probability estimate in Good-Turing estimation is of the form

$$P_{GT} = \frac{c^*}{N} \quad (c > 0) \qquad \text{or} \qquad P_{GT} = \frac{N_1}{N} \quad (c = 0)$$

- N: the total number of word tokens

- $N_1 = 3$
- $N_2 = 1$
- $N = 18$

If $C(w_1...w_n) = 1$:

c	1
MLE	$P = \frac{1}{N} = \frac{1}{18}$
c^*	$c^*(w_1...w_n) = (c+1) \times \frac{N_2}{N_1} = (1+1) \times \frac{1}{3} = \frac{2}{3}$
GT	$P_{GT}(w_1...w_n) = \frac{c^*}{N} = \frac{\frac{2}{3}}{18} = \frac{1}{27}$

- $N_1 = 3$
- $N_2 = 1$
- $N = 18$

If $C(w_1...w_n) = 0$:

c	0
MLE	$P = \frac{0}{N} = 0$
c^*	-
GT	$P_{GT}(w_1...w_n) = \frac{N_1}{N} = \frac{3}{18}$

Good-Turing Discounting (continued)

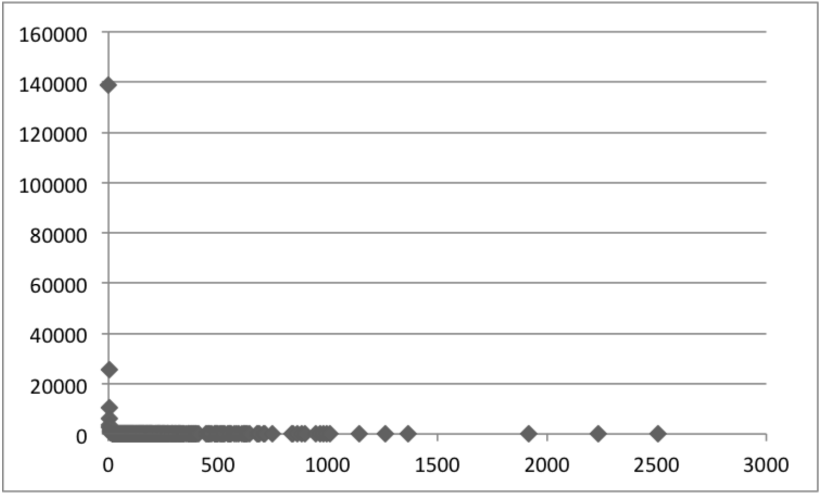
Good-Turing discounting is **undefined** when $N_{c+1} = 0$:

Simple Good-Turing

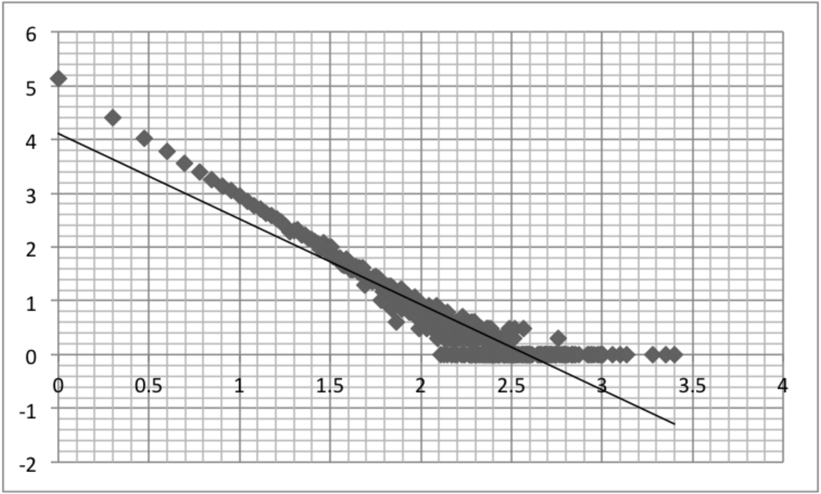
$\log(N_c) = a + b \log(c)$

How to calculate a and b?

Frequency of frequency data



Frequency of frequency data (log scale)

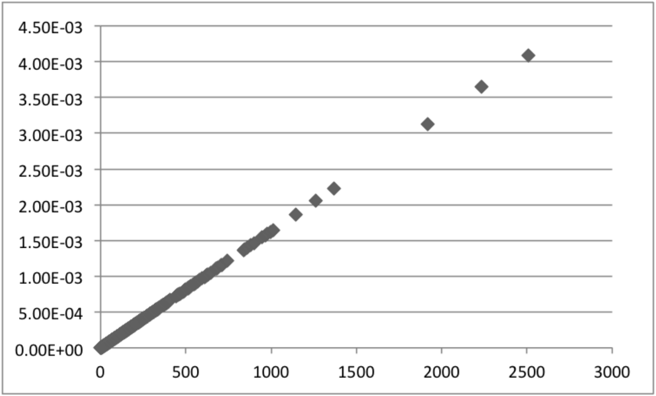


Using a linear regression, we can obtain a and b. See Sampson's C program for simple Good-Turing.

- Sampson's C program: `http://nlp.stanford.edu/fsnlp/statest/SGT.c`
- Gale and Sampson's (1995)

W. Gale and G. Sampson (1995). Good–Turing frequency estimation without tears. *Journal of Quantitative Linguistics*, **2**:217–237.
Available at `http://www.grsampson.net/AGtf1.html`

$P_{GT}(\cdot)$



Results for $P_{GT}(\cdot)$ from Sampson's SGT program.

For Katz, $c* = c$ for $c > k$ where $k = 5$ (or $k = 10$ would be generally ok).

Katz backoff

If the N-gram has zeor counts, we approximate it by backing off to the $(N-1)$-gram. We continue backing off until we reach a history that has some "counts":

$$P_{katz}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & \text{if } C(w_{n-N+1}^{n}) > 0 \\ \alpha(w_{n-N+1}^{n-1}) P_{katz}(w_n|w_{n-N+2}^{n-1}), & \text{otherwise.} \end{cases}$$

Interpolation

In simple linear interpolation, we combine different order N-grams by linearly interpolating all the models.

$$
\begin{aligned}
\hat{P}(w_n|w_{n-2}w_{n-1}) \quad = \quad & \lambda_3 P(w_n|w_{n-2}w_{n-1}) \\
& +\lambda_2 P(w_n|w_{n-1}) \\
& +\lambda_1 P(w_n)
\end{aligned}
$$

where $\sum_i \lambda_i = 1$

Algorithm for simple linear interpolation

function interpolation(*corpus*) **return** $\lambda_1, \lambda_2, \lambda_3$

$\lambda_1 \leftarrow 0$

$\lambda_2 \leftarrow 0$

$\lambda_3 \leftarrow 0$

foreach trimgram t_1, t_2, t_3

if $\frac{C(t_1,t_2,t_3)}{C(t_1,t_2)} > 0$: increase λ_3

else if $\frac{C(t_2,t_3)}{C(t_2)} > 0$: increase λ_2

else if $\frac{C(t_3)}{N} > 0$: increase λ_1

end

end

normalize $\lambda_1, \lambda_2, \lambda_3$

return $\lambda_1, \lambda_2, \lambda_3$

For the interpolation, we should learn λs from a **held-out** corpus.

Why ?

Toolkits and data formats

Well-known toolkits for LMs:

- SRILM `http://www.speech.sri.com/projects/srilm/download.html`
- IRSTLM `https://hlt-mt.fbk.eu/technologies/irstlm`
- KenLM `http://kheafield.com/code/kenlm/estimation/`

```
\data\
ngram 1=39864
ngram 2=281348
ngram 3=46198

\1-grams:
-5.516118       $.027       -0.3422345
-5.215087       $.03       -0.2032839
-5.817147       $.054/mbf       -0.1140985
...

-0.3811322 his zest for
-0.3811322 a zinc mine
-0.5572235 of zinc and

\end\
```