



# Language Modeling

# Advanced: Good Turing Smoothing



## Reminder: Add-1 (Laplace) Smoothing

$$P_{Add-1}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$



## More general formulations: Add-k

$$P_{Add-k}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

$$P_{Add-k}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$



## Unigram prior smoothing

$$P_{Add-k}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

$$P_{UnigramPrior}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + mP(w_i)}{c(w_{i-1}) + m}$$



## Advanced smoothing algorithms

- Intuition used by many smoothing algorithms
  - Good-Turing
  - Kneser-Ney
  - Witten-Bell
- Use the count of things we've **seen once**
  - to help estimate the count of things we've **never seen**



## Notation: $N_c$ = Frequency of frequency $c$

- $N_c$  = the count of things we've seen  $c$  times
- Sam I am I am Sam I do not eat

I      3

sam 2

am    2

do    1

not   1

eat   1

6

$$N_1 = 3$$

$$N_2 = 2$$

$$N_3 = 1$$



# Good-Turing smoothing intuition

- You are fishing (a scenario from Josh Goodman), and caught:
  - 10 carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, 1 eel = 18 fish
- How likely is it that next species is trout?
  - $1/18$
- How likely is it that next species is new (i.e. catfish or bass)
  - Let's use our estimate of things-we-saw-once to estimate the new things.
  - $3/18$  (because  $N_1=3$ )
- Assuming so, how likely is it that next species is trout?
  - Must be less than  $1/18$
  - How to estimate?



## Good Turing calculations

$$P_{GT}^* (\text{things with zero frequency}) = \frac{N_1}{N} \qquad c^* = \frac{(c+1)N_{c+1}}{N_c}$$

- Unseen (bass or catfish)
  - $c = 0$ :
  - MLE  $p = 0/18 = 0$
  - $P_{GT}^* (\text{unseen}) = N_1/N = 3/18$
- Seen once (trout)
  - $c = 1$
  - MLE  $p = 1/18$
  - $C^*(\text{trout}) = 2 * N_2/N_1$   
 $= 2 * 1/3$   
 $= 2/3$
  - $P_{GT}^*(\text{trout}) = 2/3 / 18 = 1/27$





# Ney et al.'s Good Turing Intuition

H. Ney, U. Essen, and R. Kneser, 1995. On the estimation of 'small' probabilities by leaving-one-out.  
IEEE Trans. PAMI. 17:12,1202-1212



Held-out words:

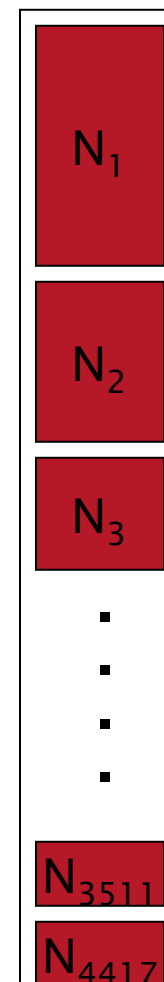


# Ney *et al.* Good Turing Intuition (slide from Dan Klein)

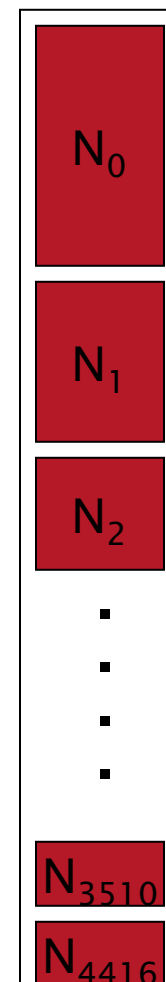
- Intuition from leave-one-out validation
  - Take each of the  $c$  training words out in turn
  - $c$  training sets of size  $c-1$ , held-out of size 1
  - What fraction of held-out words are unseen in training?
    - $N_1/c$
  - What fraction of held-out words are seen  $k$  times in training?
    - $(k+1)N_{k+1}/c$
  - So in the future we expect  $(k+1)N_{k+1}/c$  of the words to be those with training count  $k$
  - There are  $N_k$  words with training count  $k$
  - Each should occur with probability:
    - $(k+1)N_{k+1}/c/N_k$
  - ...or expected count:

$$k^* = \frac{(k+1)N_{k+1}}{N_k}$$

Training



Held out

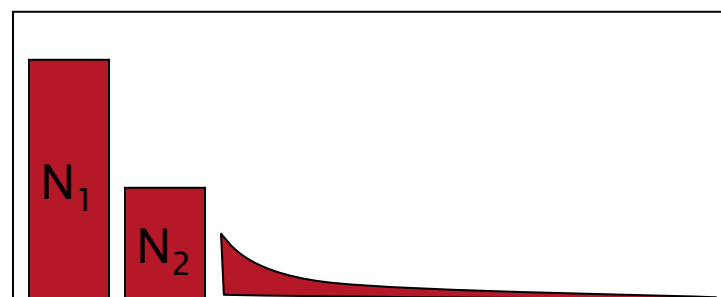
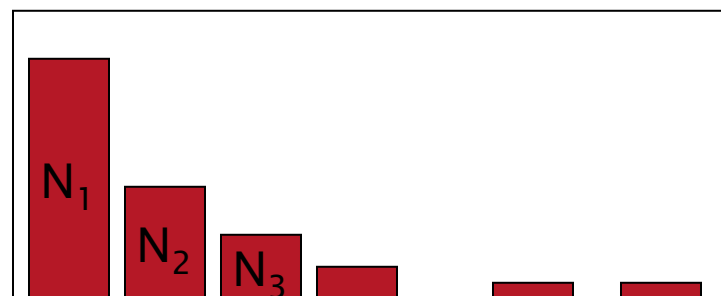




# Good-Turing complications

(slide from Dan Klein)

- Problem: what about “the”? (say  $c=4417$ )
  - For small  $k$ ,  $N_k > N_{k+1}$
  - For large  $k$ , too jumpy, zeros wreck estimates
- Simple Good-Turing [Gale and Sampson]: replace empirical  $N_k$  with a best-fit power law once counts get unreliable





# Resulting Good-Turing numbers

- Numbers from Church and Gale (1991)
- 22 million words of AP Newswire

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

| Count c | Good Turing c* |
|---------|----------------|
| 0       | .0000270       |
| 1       | 0.446          |
| 2       | 1.26           |
| 3       | 2.24           |
| 4       | 3.24           |
| 5       | 4.22           |
| 6       | 5.19           |
| 7       | 6.21           |
| 8       | 7.24           |
| 9       | 8.25           |



# Language Modeling

# Advanced: Good Turing Smoothing